*Design Guide: TIDM-02013*

# *7.4-kW EV or HEV Bidirectional Onboard Charger Reference Design With GaN*

## Description

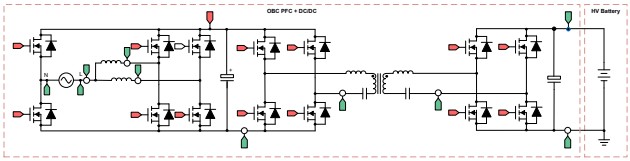
The PMP22650 reference design is a 7.4-kW, bidirectional, onboard charger. The design employs a two-phase totem pole PFC and a full-bridge CLLLC converter with active synchronous rectification. The CLLLC uses both frequency and phase modulation to regulate the output across a wide voltage range. This design uses a single TMS320F280039C microcontroller to control both the PFC and DCDC stages. This design is also supported with a TMS320F28P65x microcontroller. High density is achieved through the use of high-speed GaN switches (LMG3522-Q1). A peak system efficiency of 96.5% was achieved with an open-frame power density of 3.8 kW/L.

This design illustrates control of this power topology using a single C2000™ MCU in closed voltage and closed current-loop mode. The hardware and software available with this design help accelerate your time to market.

## Resources

| | |
|---|---|
| TIDM-02013, PMP22650 | Design Folder |
| TMS320F280039C, TMS320F28P650DK | Product Folder |
| AMC3330-Q1, AMC3302-Q1, UCC21222-Q1 | Product Folder |
| C2000WARE-DIGITAL-POWERSDK | Software Folder |
| TMDSCNCD280039C, TMDSCNCD28P65X | Tool Folder |

Ask our TI E2E™ support experts

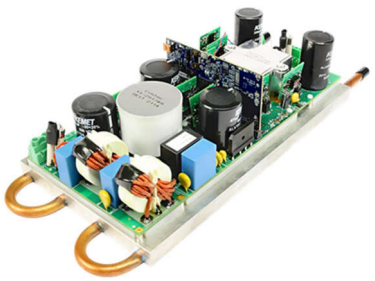


## Features

- Power Max: 7.4 kW, 96.5% peak efficiency
- VAC 90–264V AC: 240V AC TYP
- Vprim: 400 V DC nominal; Vsec: 250–450 V DC
- CLLLC resonant tank with 500-kHz nominal PWM switching (200 kHz–800 kHz range) enables higher power density
- Soft switching with Zero Voltage Switching (ZVS) on the primary; Zero Current Switching (ZCS) and ZVS on the secondary enable higher efficiency
- Active synchronous rectification scheme implementation using Rogowski coil sensor enables higher efficiency
- Software support for TMS320F28003x device with the Control Law Accelerator (CLA), which enables integrated OBC design with AC-DC and DC-DC controlled using a single C2000 MCU
- Software support for TMS320F28P65x device with one CPU (CPU1) controlling AC-DC and DC-DC stages is provided.
- Reduced CPU overhead with the new hardware oversampling feature of TMS320F28P65x

## Applications

- Hybrid, Electric, and Powertrain Systems
- DC Fast Charging Station
- Power Conversion System (PCS)



# 1 CLLLC System Description

Onboard chargers (OBCs) are an essential part of Electric Vehicles (EVs) and Hybrid Electric Vehicles (HEV). An OBC typically consists of an AC-DC [power factor correction (PFC) rectifier stage] and an isolated DC-DC converter, as shown in Figure 1-1. C2000 MCUs are designed to implement advanced digital power control that automotive applications demand; for more information, see C2000 Digital Power and C2000 EV.

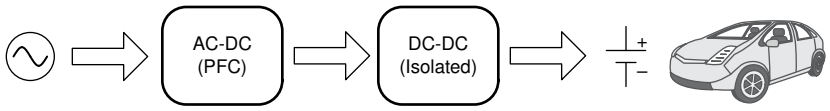



**Figure 1-1. Typical OBC Architecture**

The ability to charge the battery fully overnight is highly desired for most EV Level 1 and Level 2 chargers. With battery capacity increasing, the OBCs need to be designed for even higher power. With the increasing power capacity of the OBC, specifications such as power density and efficiency are even more important, due to limited space and cooling capacity in the car.

The CLLLC (Capacitor-Inductor-Inductor-Inductor-Capacitor)—with its symmetric tank, soft switching characteristics, and ability to switch at higher frequencies—is a good choice for these applications. In this design, control and implementation of a CLLLC topology, as shown in Figure 1-2, is illustrated.

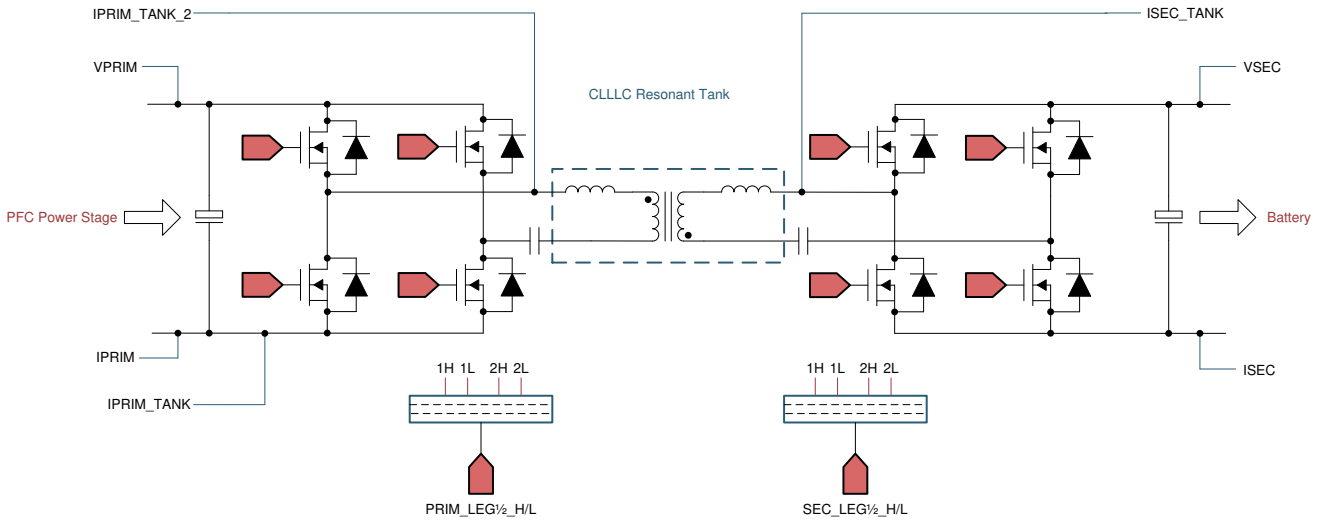



**Figure 1-2. CLLLC Topology for Isolated DC-DC Converter**

The nomenclature for Figure 1-2 is as follows:

| | |
|---|---|
| VPRIM | Primary side voltage (typically comes from a PFC converter) |
| IPRIM | Return current of the primary side, can be used for protection and monitoring. |
| IPRIM_TANK, IPRIM_TANK_2 | Tank current on the primary side, two methods to sense using shunt current sense and other is Rogowski's coil. Only one is needed, used to implement synchronous rectification in the reverse direction for example, secondary to primary. Also used for protection. |
| VSEC | Secondary side voltage (typically, a battery) |
| ISEC | Return current of the secondary side, used to implement the battery current control loop. |
| ISEC_TANK | Tank current on the secondary side, used to implement the synchronous rectification for the forward direction power flow for example, primary to secondary. |
| PRIM_LEG1/2_H/L | PWMs for the primary side full bridge |
| SEC_LEG1/2_H/L | PWMs for the secondary side full bridge |

## 1.1 Key System Specifications

The CLLLC reference design power specifications are listed in Table 1-1.

**Table 1-1. Key System Specifications**

| PARAMETER | SPECIFICATIONS |
|---|---|
| Prim Voltage (Vprim) | 400 V–450 V DC (average) |
| Sec Voltage (Vsec) | 250 V–450 V DC<br>Max |
| Power Rating forward | 7.4 kW |
| Output Current ($I_{OUT}$) | 20<br>A Max |
| Efficiency (CLLLC) | Peak 98% |
| PWM Switching Frequency | 500 kHz Nominal (200 kHz–800 kHz Range) |

**WARNING**

TI intends this reference design to be operated in a ***lab environment only and does not consider it to be a finished product*** for general consumer use. The design is intended to be run at ambient room temperature and is not tested for operation under other ambient temperatures.

TI intends this reference design 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 the design powered when unattended.*

**WARNING**

**High voltage!** There are *accessible high voltages present on the board*. Electric shock is possible. The board operates at voltages and currents that may cause shock, fire, or injury if not properly handled. Use the equipment with necessary caution and appropriate safeguards to avoid injuring yourself or damaging property. For safety, use of isolated test equipment with over-voltage and over-current protection is highly recommended.

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 design or components connected to the design*.

**WARNING**

**Hot surface! Contact may cause burns. Do not touch!**

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.

# 2 CLLLC System Overview

## 2.1 Block Diagram

Figure 2-1 shows the block diagram of the CLLLC topology.

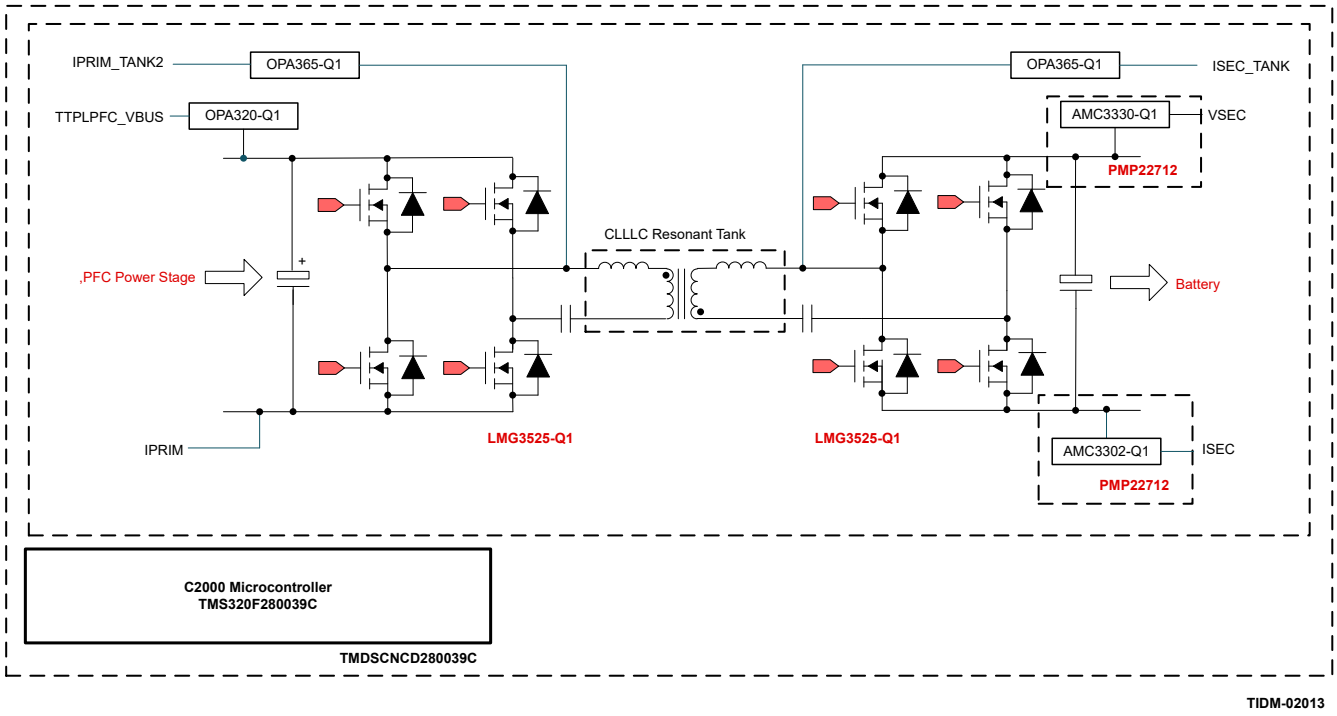



**Figure 2-1. CLLLC Block Diagram**

This reference design uses the following EVMs and PMP designs to achieve operation as documented in this guide:

1. OBC Base Board: PMP22650( with PMP22712 +PMP22773 Daughter cards)
2. F280039C controlCARD Evaluation Module: TMDSCNCD280039C

The following sections discuss details of the hardware, software, and system design.

## 2.2 Design Considerations and System Design Theory

LLC converters are widely popular due to their ability to achieve ZVS at the primary side, and ZCS on the secondary side. A typical LLC Series Resonant Converter (SRC) is shown in Figure 2-2. The primary side of this converter is half bridge; thus, the transformer utilization from a Volt-sec perspective is half. In addition, the current rating for the switches is twice of what is needed, compared to when a full-bridge structure is used.

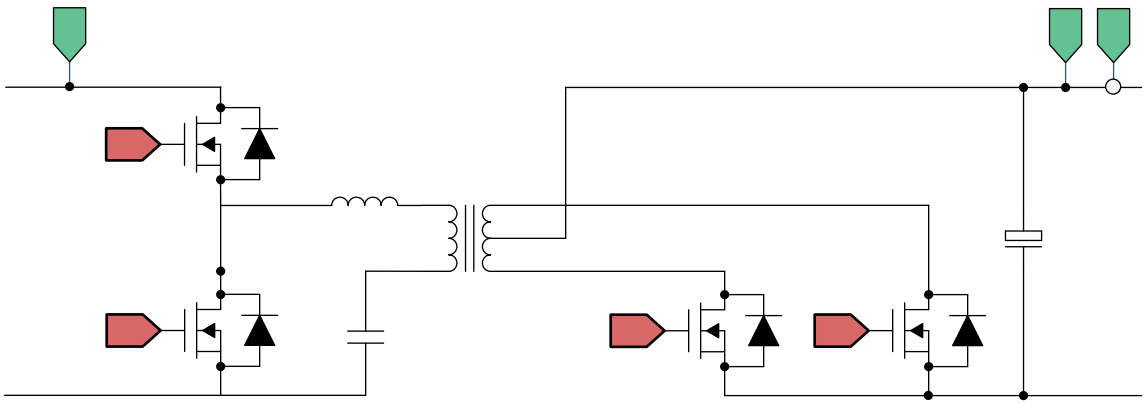


**Figure 2-2. LLC Half-Bridge SRC**

Although half-bridge LLC SRCs are attractive at lower power for cost reasons, for high-power and high-density applications, a full-bridge LLC SRC is desired for the following reasons:

1. A full-bridge LLC converter better utilizes the magnetic core of the transformer on both the secondary side and primary side; therefore, it is able to offer better power density.
2. A full-bridge LLC converter reduces current rating; and therefore, reduces the cost of copper in wires. The converter also enables higher power (compared to half-bridge SRCs) to be achieved with the same copper wires.

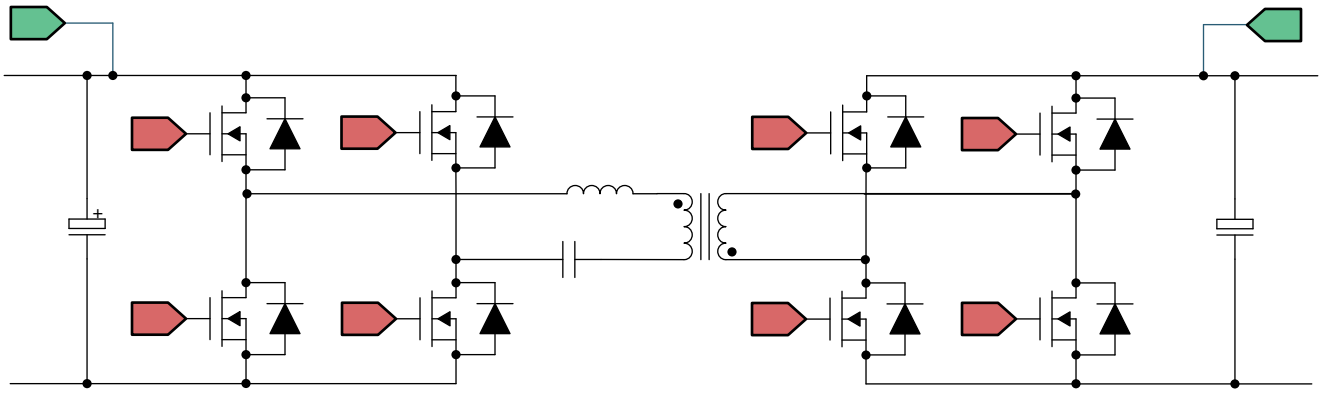


**Figure 2-3. Full-Bridge LLC Converter**

A full-bridge LLC converter, as shown in Figure 2-3, falls under the broad category of Dual Active Bridge (DAB) converters. Under DAB converters, the converter can be classified on the basis of model or operation:

1. A phase-shifted DAB converter is one of the most popular converters historically.
2. Resonant DAB converters have different variants on resonant tanks (LC, LLC, CLLC, CLLLC, and so forth).

Resonant DAB converters are of interest because high efficiency, high power, and high density are achievable with such converters. CLLLC, with its symmetric tank, is capable of bidirectional operation. The problem with using an LLC structure for bidirectional use is that the switching frequency, when operating in the reverse power flow mode, is governed by the transformer winding capacitance and the leakage inductance. This offers little or no control on the gain of the power stage and the switching frequency. Therefore, the CLLLC type of structure is preferred as it offers much better control on the switching frequency and an additional degree of freedom on the gain.

### 2.2.1 Tank Design

In this section, the tank parameter selection for the CLLLC is discussed based on the voltage gain desired, soft switching characteristics, and an appropriate power profile is selected for the charger based on CLLLC.

For additional calculations and information, refer to the following files located inside the software install package at C2000Ware_DigitalPower_SDK_<ver>/solution/tidm_02013/hardware/

#### *2.2.1.1 Voltage Gain*

To understand tank design, first the gains for both battery-charging mode and reverse-power-flow mode must be analyzed with First Harmonic Analysis (FHA) using first harmonic approximation. The simplified diagram of the resonant tank is given in Figure 2-4.

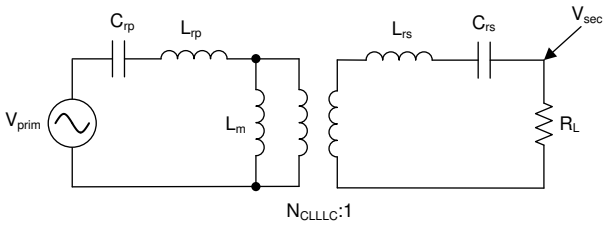



**Figure 2-4. FHA Model for CLLLC Resonant Tank During Battery Charging Mode (BCM)**

The nomenclature for Figure 2-4 is as follows:

| | |
|---|---|
| $V_{prim}$ (TTPLPFC_$V_{BUS}$) | Voltage input at primary side |
| $L_{rp}$ | Primary side resonant inductor |
| $C_{rp}$ | Primary side resonant capacitor |
| $N_{CLLLC}$ | Turns ratio of the transformer |
| $L_m$ | Magnetizing inductor |
| $V_{sec}$ | Voltage output at secondary side |
| $L_{rs}$ | Secondary side resonant inductor |
| $C_{rs}$ | Secondary side resonant capacitor |
| $R_L$ | Effective load seen with FHA on the secondary output |

Note here the effective $R_L$ is accounted for as $R_L = \left(\frac{8}{\pi^2}\right) R_{L\_dc}$ where $R_{L\_dc}$ is the DC resistive load at the output.

Referring to secondary side quantities on the primary side,

- $L_{rs}'$ is equal to $L_{rs} * N_{CLLLC} * N_{CLLLC}$
- $C_{rs}'$ is equal to $C_{rs} / (N_{CLLLC} * N_{CLLLC})$
- $R_L'$ is equal to $R_L * (N_{CLLLC} * N_{CLLLC})$
- $V_{rs}'$ is equal to $V_{rs} * N_{CLLLC}$

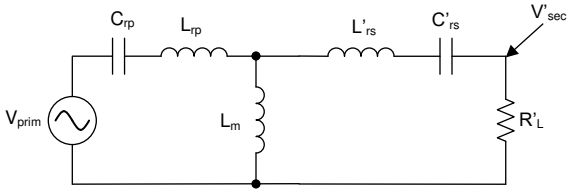



**Figure 2-5. FHA CLLLC With Quantities Referred to Primary Side in BCM**

Using KCL and KVL, the gain equation can be written as Equation 1.

$$\frac{V_{sec}}{V_{prim}} = \frac{\left[Z_m \,||\, \left(Z_{rs}' + R_L'\right)\right] R_L'}{\left(Z_{rp} + \left[Z_m \,||\, \left(Z_{rs}' + R_L'\right)\right]\right)\left(Z_{rs}' + R_L'\right) N_{CLLLC}} \quad (1)$$

Similarly, for the reverse power flow, the circuit can be simplified as shown in Figure 2-6, and the gain can be written as Equation 2.

$$\frac{V_{prim}}{V_{sec}} = \frac{N_{CLLLC}\left[Z_m \,||\, \left(Z_{rp} + R_L\right)\right] R_L}{\left(Z_{rs}' + \left[Z_m \,||\, \left(Z_{rp} + R_L\right)\right]\right)\left(Z_{rp} + R_L\right)} \quad (2)$$

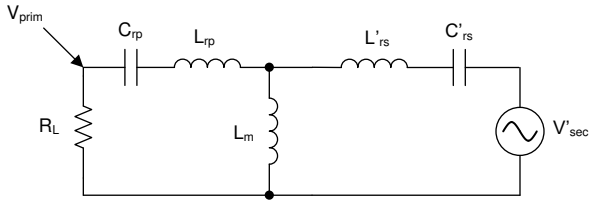



**Figure 2-6. FHA Model for Gain Calculation in RCM**

Equation 1 and Equation 2 are used in the following section to study the voltage gain based on the parameters selected for the design.

#### 2.2.1.2 Transformer Gain Ratio Design ($N_{CLLLC}$)

Resonant converters are typically most efficient when operating at or near the resonant frequency. Since this is a bidirectional battery charger, this design needs to cover a range of output voltages. This implies that *n* should be chosen such that the converter operates at as low a current as possible to help reduce $I^2R$ losses. Following this design concept the highest output current will occur at the lowest output voltage at which you need to supply full power. We will set *n* at this point such that the converter operates as close to resonance as possible. In this design this works out to be a turns ratio of 1.1:1. This enables the lowest losses while still allowing for a wide output voltage range.

#### 2.2.1.3 Magnetizing Inductance Selection ($L_m$)

To ensure ZVS operation of the primary side FETs, we need to make sure the energy stored in the resonant tank is greater than the energy stored in the FET output capacitors. We can use Equation 3 to determine the needed $L_m$ for full-bridge LLC SRCs.

$$L_m \leq \frac{Tt_{dead}}{16 * C_{oss}} \tag{3}$$

where the intended switching frequency for the converter is 500 kHz, hence T = 1/(500 * $10^3$), and based on the power device. Selected parameters such as $t_{dead}$ and $C_{oss}$ can also be identified from the power device data sheet. Typically, the effective $C_{oss}$ must be calculated using curve fitting. On this design, based on the design parameters discussed, $L_m$ must be less than 20 µH. In addition to what is accounted for in the above calculation, there is inter-winding capacitance in a real transformer that needs to be discharged by the resonant tank current. Therefore, using simulation, a value of 14 µH was selected to ensure ZVS across the operating range of the converter; this value is used in the subsequent selection processes.

#### 2.2.1.4 Resonant Inductor and Capacitor Selection ($L_{rp}$ and $C_{rp}$)

While selecting $L_{rp}$, the ratio of $L_m$ to $L_{rp}$ is widely used as a design parameter,

$$L_n = \frac{L_m}{L_{rp}} \tag{4}$$

The $L_n$ value is selected such that it ensures the voltage gain in the resonant tank is enough across the operating range of the converter. In this design, as the input voltage comes from a PFC stage and will have a estimated 10% ripple, a gain variation of at least 10% is needed. With this criteria in mind and the fact that $L_n$ should be kept higher to reduce the inductor value, and hence the losses, $L_n$ equal to 14 is selected for this design, based on the plot of the FHA with $L_n$ varying with load (see Figure 2-7).

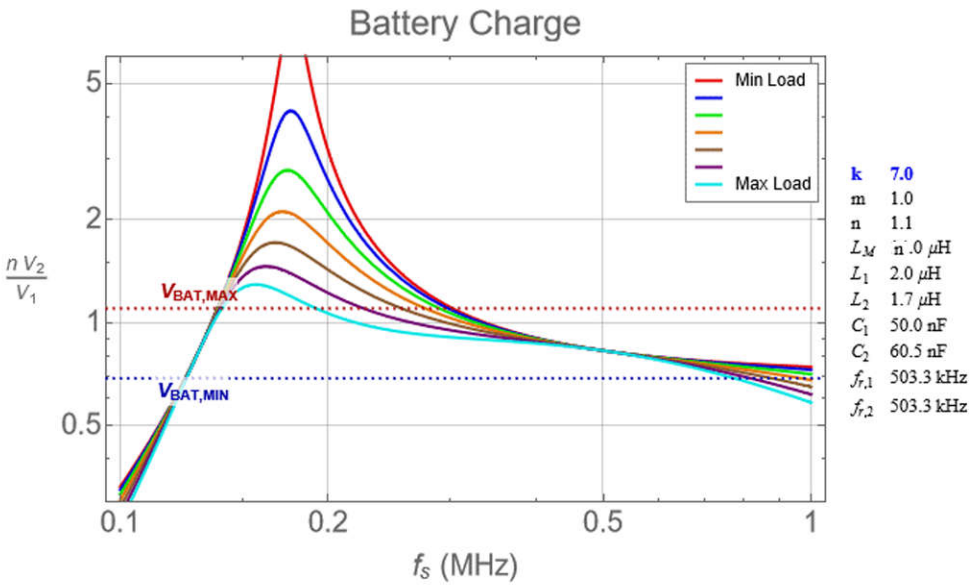



**Figure 2-7. CLLLC Tank Gain Variation With $L_n$ Varying**

Now that the selection of $L_n$ is made, $L_{rp}$ can be calculated using Equation 4. $L_{rp}$ and $C_{rp}$ determine the series resonant frequency of the converter and they are related by Equation 5.

$$f_{res} = \frac{1}{2\pi\sqrt{L_{rp}C_{rp}}} \tag{5}$$

Equation 5 can then be used to calculate the $C_{rp}$ needed on the design. However, due to component availability, the next closest value of $C_{rp}$ is used on the design. With these component values, the BCM gain is shown in Figure 2-7.

In Figure 2-7, as the load increases (that is, $R_{L\_dc}$ goes lower), the gain curve becomes non-monotonic in the region below series resonant frequency. This can lead to the loss of ZVS on the primary FETs and, more critically, the loss of control. Therefore assuming maximum load at nominal Vout, the load is limited or clamped to $R_{L\_dc}$ = 30 Ω, for which the gain is monotonic (see Figure 2-7).

Additionally Figure 2-7 shows that in BCM we have enough gain across our operating frequency of 200kHz to 800kHz to cover all operating conditions. Lastly, it is worth noting that if the PFC ripple can be reduced then the totally expected input range will also reduce. This causes the required gain range to reduce, and ultimately helping to reduce the frequency variation needed to support all load conditions.

### 2.2.2 Current and Voltage Sensing

In the following sections, the sensing scheme for different currents and voltages on the design are discussed. On the design, multiple schemes are implemented so that users can select the appropriate schemes for their application needs.

#### *2.2.2.1 VPRIM Voltage Sensing*

The C2000 MCU is biased on the primary side; hence, the primary voltage is sensed by a resistor divider to the ground of the board. As oversampling is used, an op amp in voltage follower arrangement is used to buffer the signal for the ADC as shown in Figure 2-8. The buffer helps reduce impedance as seen by the ADC, and hence, a faster sampling rate can be used. Otherwise, the sampling will be limited by the time constant of the resistor divider resistance, which is typically high, and hence, only slow sampling can be done.

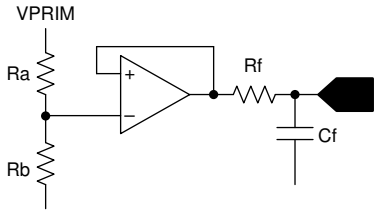



**Figure 2-8. VPRIM Voltage Sensing Circuit**

#### *2.2.2.2 VSEC Voltage Sensing*

The secondary side voltage is sensed in an isolated manner using the AMC3330-Q1, as shown in Figure 2-9.

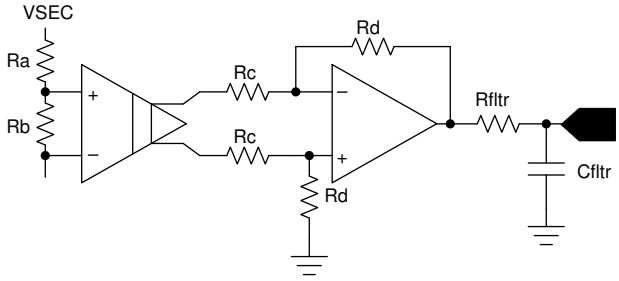



**Figure 2-9. VSEC Voltage Sensing Circuit**

#### *2.2.2.3 ISEC Current Sensing*

The secondary side output current is also sensed in an isolated manner using the AMC3302-Q1, as shown in Figure 2-10.

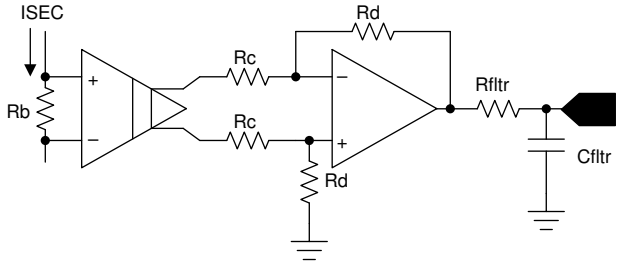



**Figure 2-10. ISEC Current Sensing Circuit**

### 2.2.2.4 ISEC TANK and IPRIM TANK

A Rogowski coil-based sensing mechanism is chosen to sense the high-frequency current in the tank on the primary side and the secondary side in an isolated manner, as shown in Figure 2-11. The ADC pin is internally connected to the Comparator Subsystem (CMPSS), which can generate the correct pulses that go through the X-Bar to the PWM to get the action required for synchronous rectification.

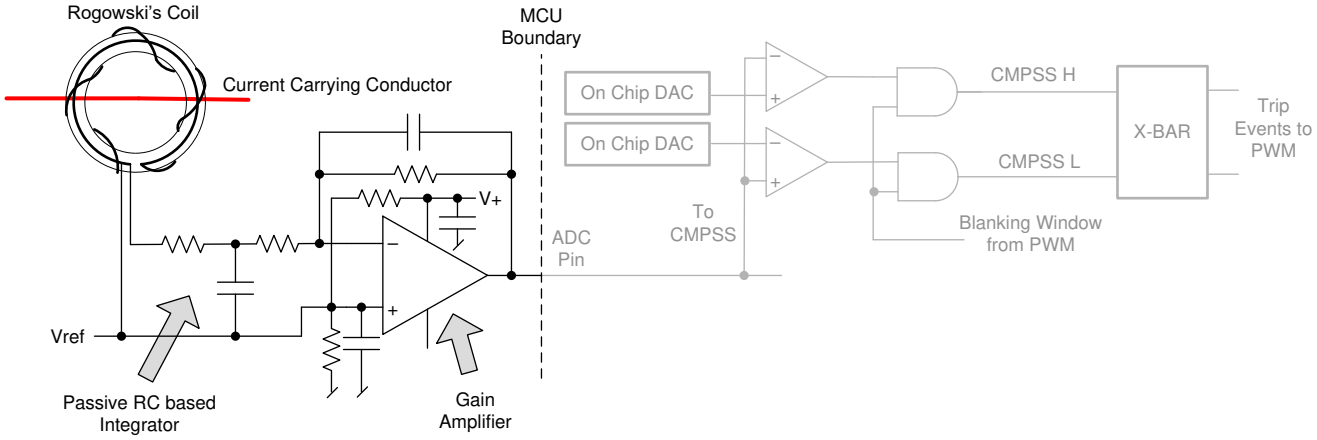



**Figure 2-11. ISEC Tank Current Sensing Using Rogowski's Coil**

### 2.2.2.5 IPRIM Current Sensing

The primary side current, IPRIM, is sensed using LMV796-Q1.(see Figure 2-12).

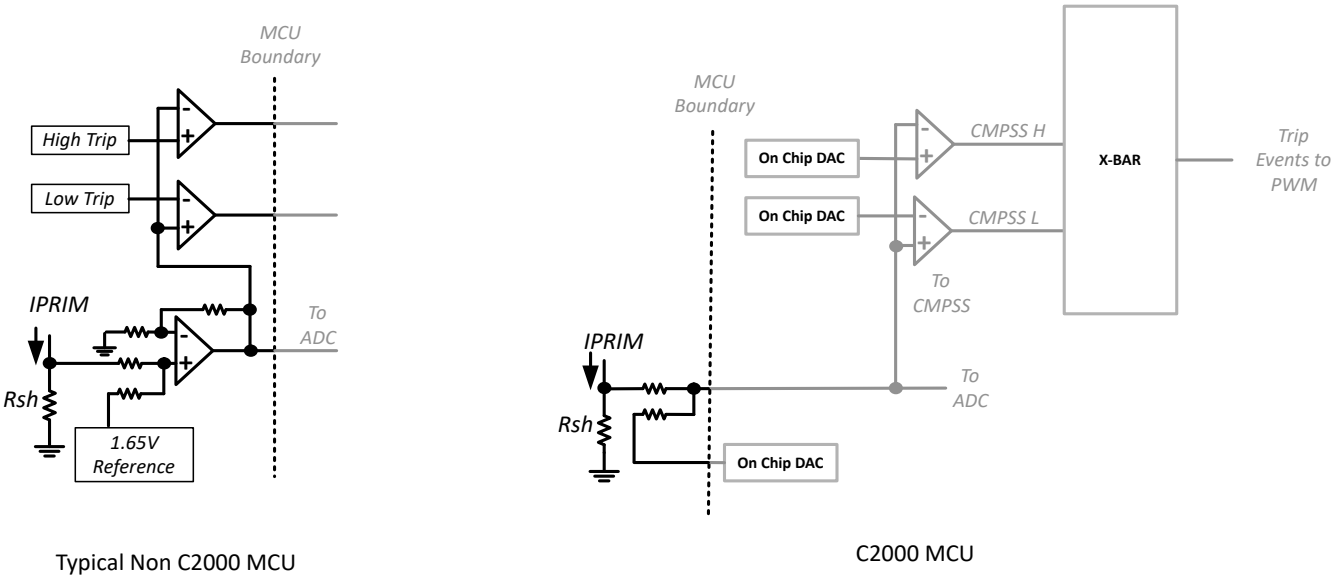



**Figure 2-12. IPRIM Current Sensing Circuit, Comparison With Typical MCU vs. C2000 MCU**

### 2.2.2.6 Protection (CMPSS and X-Bar)

Most power electronics converters need protection from overcurrent events. For this design, multiple comparators are needed, and references for the trip points need to be generated. Using C2000 MCUs—such as TMS320F280039, which has an on-chip windowed comparator as part of the Comparator Subsystem (CMPSS) along with 12-bit DACs for trip set points—that are internally connected to the PWM module enables fast tripping of the PWM without the need of external hardware. This saves board space and cost in the end application as extra components can be avoided by using on-chip resources such as DAC, comparators, and ADC. All of these resources can be used together and at the same time, without any extra external connections. Furthermore, the CMPSS-generated signals go to the X-Bar, where they can be combined in different and unique fashions to flag unique trip events from multiple sources.

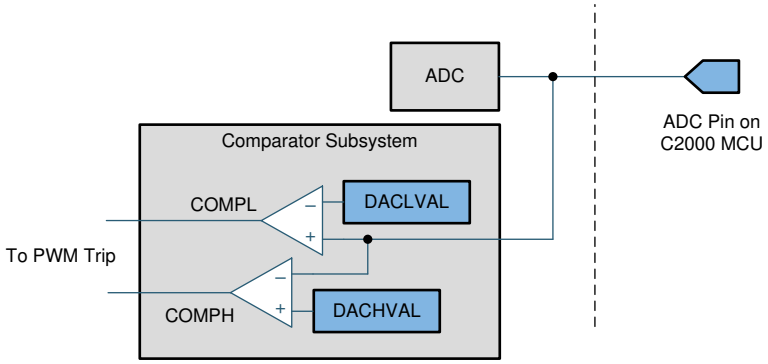



**Figure 2-13. Comparator Subsystem (CMPSS) Used for Overcurrent Protection**

### 2.2.3 PWM Modulation

Figure 2-14 shows the PWM waveform configuration used on this design.

High-resolution PWM is used for the primary legs and the secondary legs. Up-down count mode is used to generate the PWMs. To use the high-resolution PWMs, the PRIM_LEG1_H PWM pulse is centered on the period event and the time base is configured to be up-down count. A complementary pulse with high-resolution dead time is then generated for the complementary switch. Between LEG1 and LEG2, there is a 180-degree phase shift for a full-bridge operation. This is achieved by using the feature on the PWM module to swap the xA and xB output. (Alternatively, a phase shift can also be implemented, but is not needed on this design.)

The PWM pulse to the secondary side goes through an isolator, which adds additional propagation delay. To account for this propagation delay, a small advance of the PWM is required. This is implemented in form of a phase-shift delay with respect to the primary active PWM pulse's falling edge. The phase shift of the secondary side is a combination of the period and the delay needed for the isolator, as shown in Figure 2-14. As active synchronous rectification scheme is used, the rising edge is controlled by the primary side PWM switch timing. As the switching event can be noisy, a blanking window is used. The current in the secondary tank can be discontinuous depending on the operating frequency and load. Hence, the falling edge is controlled by the trip action that is triggered as soon as the secondary current reaches zero. The trip is then latched until the next zero or period event to avoid any spurious turn on of the secondary side switches because of noise. The blanking pulse is generated by the PWM time base but the trip latch and the blanking actions happen as part of the CMPSS. Depending on whether it is the positive half or the negative half of the tank current, two different trip signals are generated and sent to the PWM module through the X-Bar. The Type-4 PWM on the C2000 MCU can uniquely use these events to trip the xA pulse during the up count and xB during the down count. For details, refer to the code in the function CLLLC_HAL_setupSynchronousRectificationAction(), which is the HAL file for the solution, see Section 5.1.2.

The global link mechanism on the Type-4 PWM is used to reduce the number of cycles needed to update the registers and enables high-frequency operation. For example, the following code in the CLLLC_HAL_setupPWM() function links the TBPRD registers for all the PWM Legs. Using this linkage, a single write to the PRIM_LEG1 TBPRD register will write the value to PRIM_LEF2, SEC_LEG1, and SEC_LEG2.

```
EPWM_setupEPWMLinks(CLLLC_PRIM_LEG2_PWM_BASE,
                    EPWM_LINK_WITH_EPWM_1,
                    EPWM_LINK_TBPRD);
```

```
EPWM_setupEPWMLinks(CLLLC_SEC_LEG1_PWM_BASE,
                    EPWM_LINK_WITH_EPWM_1,
                    EPWM_LINK_TBPRD);
```

```
EPWM_setupEPWMLinks(CLLLC_SEC_LEG2_PWM_BASE,
                    EPWM_LINK_WITH_EPWM_1,
                    EPWM_LINK_TBPRD);
```

High-resolution PWM relies on carrying forward remainder calculation from the previous cycle into the next; hence, a periodic sync should not be used between the primary and secondary side PWMs to maintain the phase relation. Whenever a frequency change or duty change is detected, a one-time sync is issued using a fast interrupt service routine (ISR1, see Section 5.1.2.2).

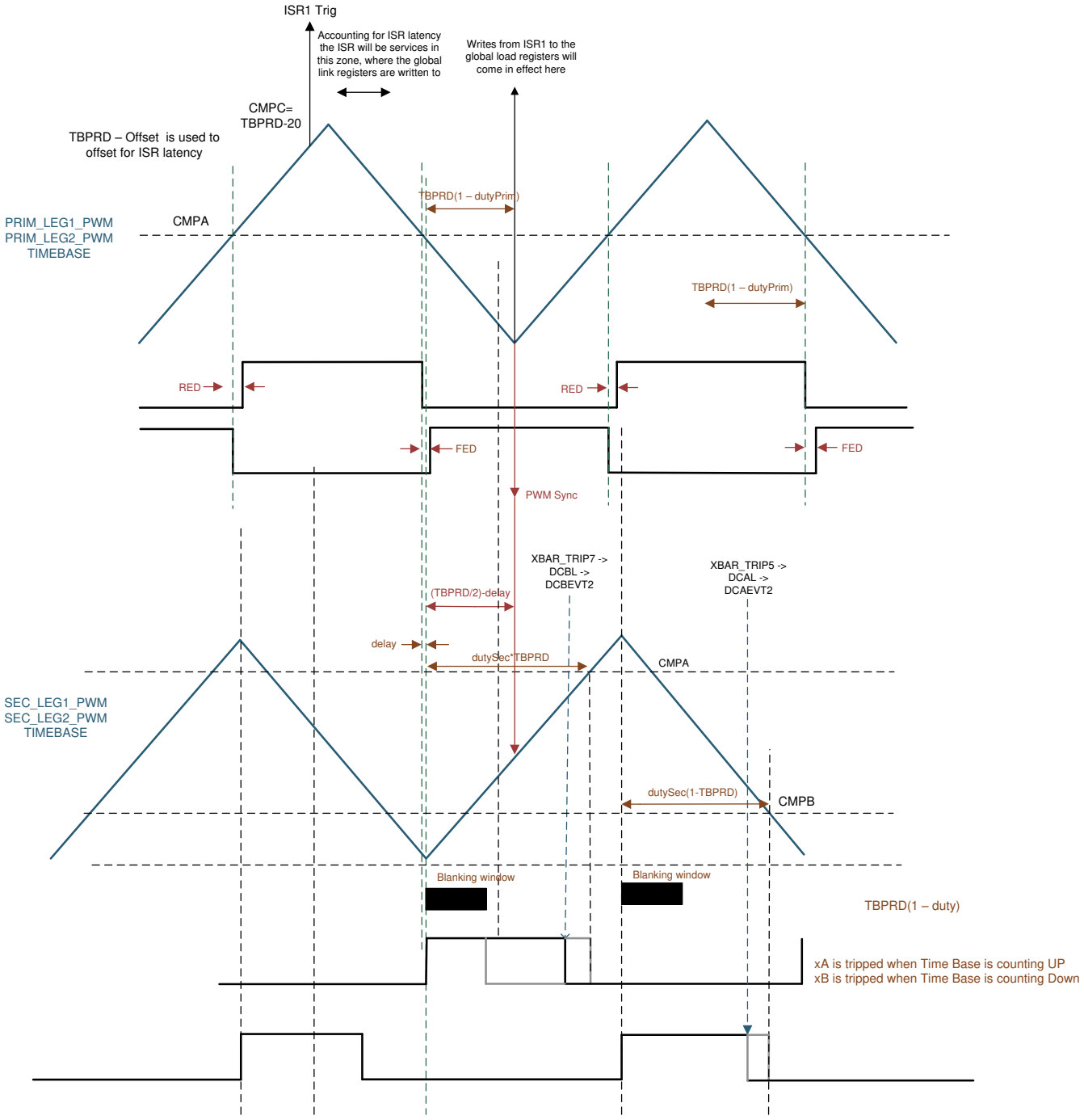



**Figure 2-14. PWM Scheme Used on CLLLC Design With Active Synchronous Rectification with Power Flow Primary to Secondary**

Similarly for the reverse power flow direction, the PWM configuration used is shown in Figure 2-15

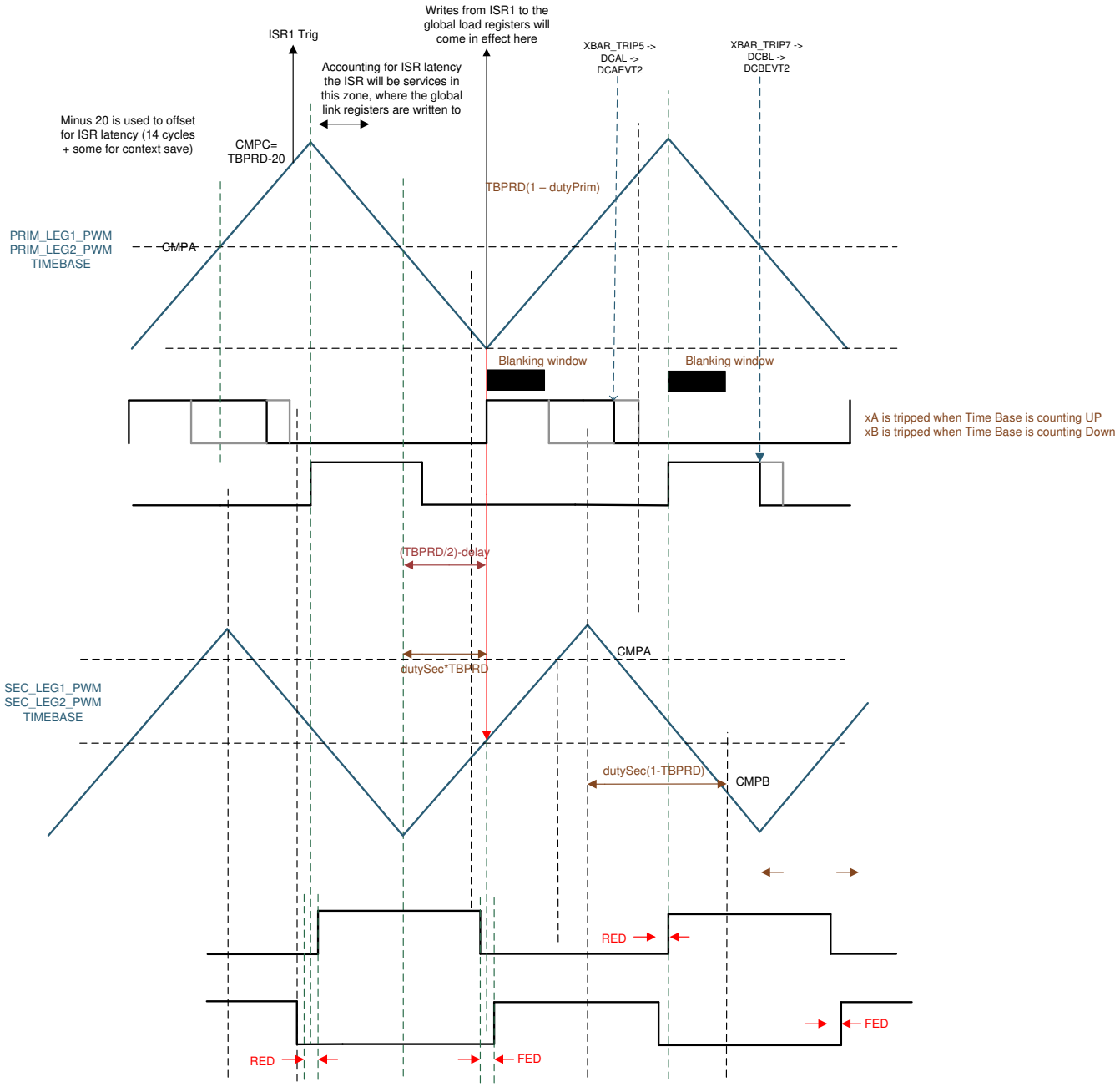



**Figure 2-15. PWM Scheme Used on CLLLC Design With Active Synchronous Rectification with Power Flow Secondary to Primary**

# 3 Totem Pole PFC System Description

**WARNING**

Do not leave the device powered when unattended.

High voltage! There are accessible high voltages present on the board. Electric shock is possible. The board operates at voltages and currents that may cause shock, fire, or injury if not properly handled. Use the equipment with necessary caution and appropriate safeguards to avoid injury or damage to property. For safety, use of isolated equipment with overvoltage and overcurrent is highly recommended. 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 design or components connected to the design.

Hot surface! Contact may cause burns. Do not touch! 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.

## 3.1 Benefits of Totem-Pole Bridgeless PFC

All plug-in hybrid electric vehicles (PHEVs) require an onboard charger (OBC) between the power grid and the high-voltage battery pack located inside the vehicle. Implementing a power factor correction (PFC) converter is mandatory to connect directly to the power grid for AC/DC power conversion and maximize the real power that flows to the downstream DC/DC converters.

Conventional PFC converters implement a passive diode bridge for rectification, which is now known as a passive PFC technique. The advantages of such a scheme are: simple design, reliability, slow-system control loop, and low cost. However, the disadvantages are also very obvious: the passive components are heavy with a low-power factor and generate significant power losses, which results in bulky heat sinks and a lot of heat dissipation. Further investigation into the matter shows that an input bridge consumes approximately 2% of the input power at the low line of a wide mains application. If the designer can suppress one of the series diodes, then they can save 1% of the input power, which allows the efficiency to rise from 94% to 95% (Turchi; Dalal; Wang; Lenck 2014). Due to previously-mentioned drawbacks, the power rating of bridged traditional PFCs is limited under hundreds of watts, especially in a hybrid-electric vehicle (HEV) or electric vehicle (EV) where reduced space and weight are the key design parameters.

As a result, the trend continues to move toward a bridgeless architecture with the elimination of the traditional diode bridge. The OBC is based on a silicon power device and has limitations such as low efficiency, low power density, and high weight. With the advantages of the SiC MOSFET, the designer can greatly improve these limitations by utilizing the superior performance of fast switching, low reverse recovery charge, and a low $R_{DS(ON)}$.

Figure 3-1 shows the basic structure of the totem-pole bridgeless PFC boost rectifier. The component consists of a boost inductor, two high-frequency boost GaN or SiC switches (Labeled $SiC_1$ and $SiC_2$ in the diagram below), and two components for conducting current at the line frequency. The line frequency components can be two slow diodes, as Figure 3-1 shows. Side (A) shows two silicon diodes ($D_1$ and $D_2$). Side (B) shows that the use of $Si_1$ and $Si_2$ further increases the efficiency.

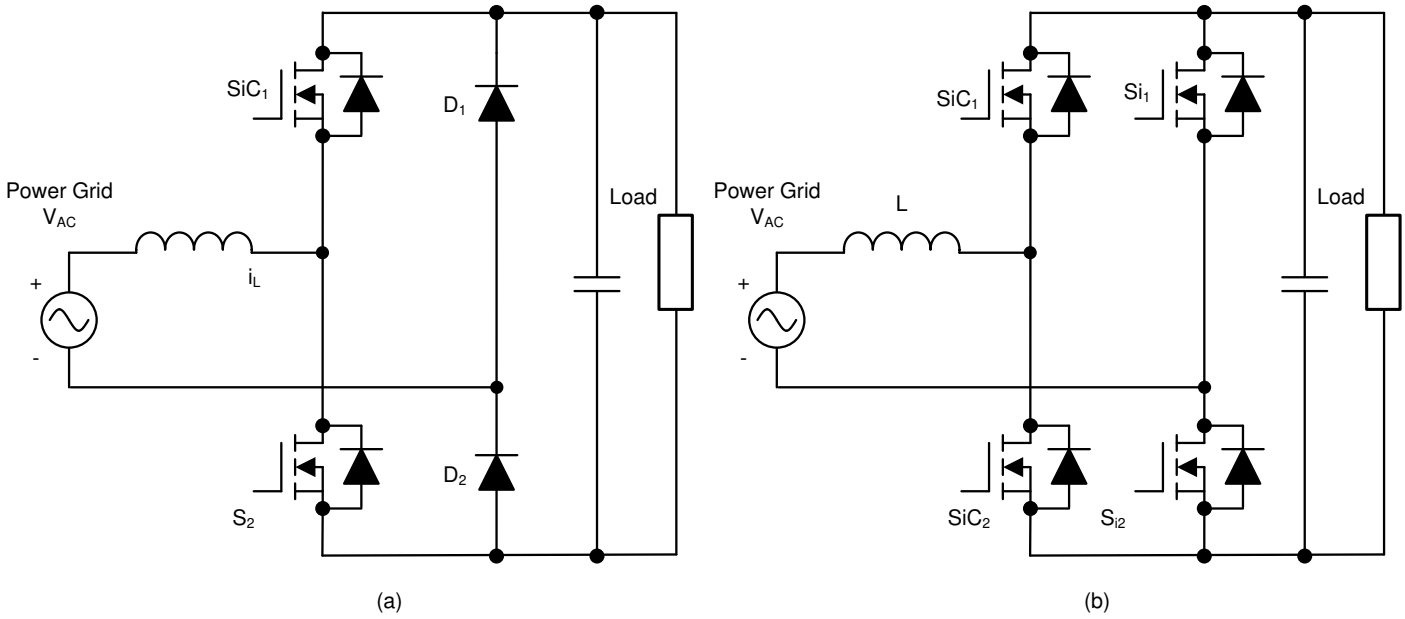



**Figure 3-1. Totem-Pole Bridgeless PFC Boost Converter Topology: (A) Diode for Line Rectification (B) MOSFET for Line Rectification**

The inherent issue in the totem-pole PFC is the operation mode transition at the AC voltage zero-crossing. When the AC input changes from the positive half line to the negative half line at the zero-crossing, the duty ratio of the low-side high-frequency switch $SiC_2$ changes from 100% to 0%, and the duty cycle of $SiC_1$ changes from 0% to 100%. Because of the slow reverse recovery of the high-side diode (or body diode of the MOSFET), the voltage at the cathode of $D_2$ cannot jump from ground to DC+ voltage instantly (this causes a large current spike). Because of this issue, the designer cannot use an Si MOSFET in a continuous-conduction mode (CCM) totem-pole PFC. Therefore $SiC_1$ and $SiC_2$ must be either gallium nitride (GaN) or SiC MOSFET field-effect transistors (FETs), which have a low reverse recovery, for TIDM-02013 we have chosen GaN FETs.

The biggest advantage of the totem-pole PFC is the reduced power losses in the conduction path. Table 3-1 shows the device comparison between a conventional PFC and a totem-pole PFC.

**Table 3-1. Device Comparison of Conventional Bridged PFC and Totem-Pole Bridgeless PFC**

| PARAMETER | LOW-FREQUENCY DIODES | HIGH-FREQUENCY DIODES | HIGH-FREQUENCY SWITCHES | CONDUCTION PATHS |
|---|---|---|---|---|
| Conventional bridged PFC | Four | One | One | Two low-speed diodes + one switch or (two low-speed diodes + one high-speed diode) |
| Totem-pole bridgeless PFC | Two | Zero | Two | One high-speed GaN switch + one low-speed Si (or SiC) MOSFET |

The following list summarizes the benefits of the totem-pole PFC:

- Although the conventional PFC boost converter is the most popular topology, its efficiency suffers from the conduction losses of the front-end diode bridge rectifier and it is not bidirectional. A totem-pole PFC is inherently capable of bidirectional operation.
- Bridgeless PFC boost converters greatly reduce the number of diodes, increase the power density, and increase the efficiency.
- This PFC is superior in terms of: high efficiency, small common mode noise, small AC current ripple, small reverse recovery current, and fewer components.
- The low reverse recovery charge of the GaN body diode and the low turn on resistance of the GaN FET make the converter an efficient and cost-effective solution for bidirectional onboard chargers.

## 3.2 Totem-Pole Bridgeless PFC Operation

The totem-pole PFC operates in the positive and negative cycles of the AC mains input, respectively, and determines the current flow depending on how the high frequency GaN MOSFETs are switched (see Figure 3-2 and Figure 3-3, respectively).

The high-frequency GaN MOSFETs together with the inductor create a synchronous mode boost converter. During the positive half cycle, $S_2$ is the boost switch which is driven with duty cycle D and $S_1$ is driven with a complementary pulse-width modulation (PWM) signal (1-D). Figure 3-2 (A) shows the direction in which the current flows. Similarly, during the period when $S_2$ is switched with 1-D, $S_1$ is switched with D; Figure 3-2 (B) shows the direction in which the current flows. Note that, during this cycle, $S_{D2}$ conducts continuously.

During the negative half cycle, the operation is similar except that the role of the high-side- and low-side, high-frequency switches are swapped. Figure 3-3 shows the direction in which the current flows. Note that, during this cycle, $S_{D1}$ conducts continuously.

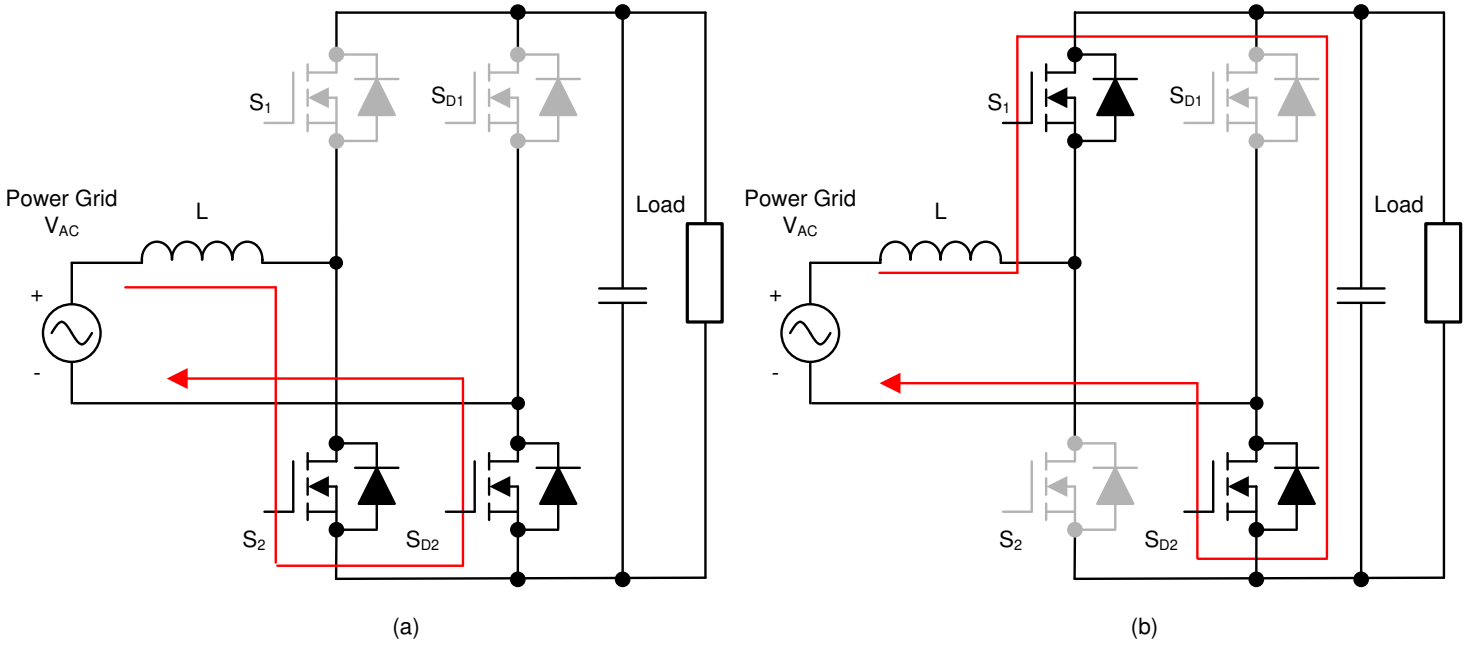



**Figure 3-2. Totem-Pole Bridgeless PFC Operation During Positive Half Cycle: (A) While $S_2$ is Switched ON (B) While $S_2$ is Switched OFF**

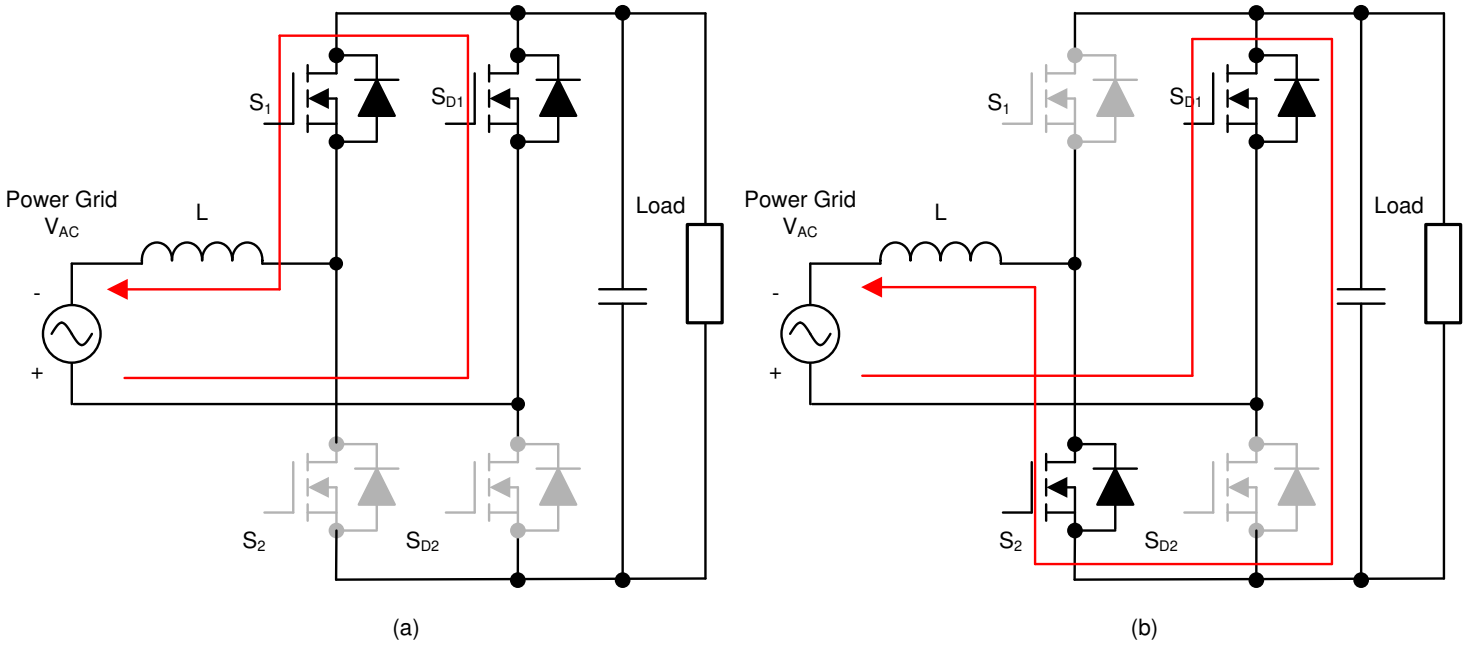



**Figure 3-3. Totem-Pole Bridgeless PFC Operation During Negative Half Cycle: (A) While $S_1$ is Switched ON (B) While $S_1$ is Switched OFF**

This reference design uses GaN FETs (LMG3522R030-Q1) and TI's C2000™ Piccolo™ (TMS320F280039C) high-performance MCU. The high-frequency GaN FETs operate at a 120-kHz switching frequency and the pair of Si MOSFETs operate at the line frequency (approximately 45 Hz to 60 Hz). Thus the conduction path includes one GaN switch and one low-frequency Si switch with significantly-reduced conduction losses. The use of two-channel interleaving to reduce conduction loss and input current ripple. Test results demonstrate a high efficiency above 98.5%.

## 3.3 Key System Specifications

Table 3-2 lists the key system specifications of this design.

**Table 3-2. TIDA-02013 PFC Key System Specifications**

| PARAMETER | SPECIFICATIONS |
|---|---|
| Input | • Single phase<br>• Voltage: ≈ 90-V $AC_{RMS}$ to 264-V $AC_{RMS}$<br>• AC line frequency range: 50Hz to 60 Hz<br>• Input current: 32 $A_{RMS\_MAX}$ at 240 V, 32 $A_{RMS\_MAX}$ at 120 V<br>• Power factor: ≥ 0.99 |
| Output | • PFC output: ≈ 400 V Typical<br>• Maximum output power: 7.4kW at ≈ 400 V<br>• Peak efficiency: 98.5% |
| Performance | • PFC stage for high-voltage li-ion battery OBC<br>• Switching frequency: 120 kHz<br>• Isolation: Reinforced<br>• Input AC sensing<br>• PFC output voltage sensing |
| Protection | • Overtemperature protection<br>• Short-circuit protection<br>• Overcurrent protection<br>• Undervoltage protection<br>• Overvoltage protection at |

## 3.4 System Overview

### 3.4.1 Block Diagram

Figure 3-4 shows the system block diagram of the TIDM-02013 reference design, which includes the following elements.

- Power switches G1-G4 are high-frequency GaN MOSFETs, for which there is a 180° phase shift between each half bridge leg. G5 and G6 forms a low-frequency (40- to 60-Hz) synchronous rectifier bridge which virtually has no switching loss; a low conduction loss feature is desirable for these two devices.
- TMS320F280039C C2000 real-time microcontroller functions as the controller, which has all the voltage and current sensor inputs and generates the correct PWM signals for G1-G6. The controller also reads any fault signal from the gate driver boards and shuts down the system if a fault occurs. The reset function is used during start-up or when a fault clears.
- Hall sensors are used to sense the total input current and current for each channel. Voltage dividers are used to sense the input line and neutral voltages as well as output DC bus voltages.

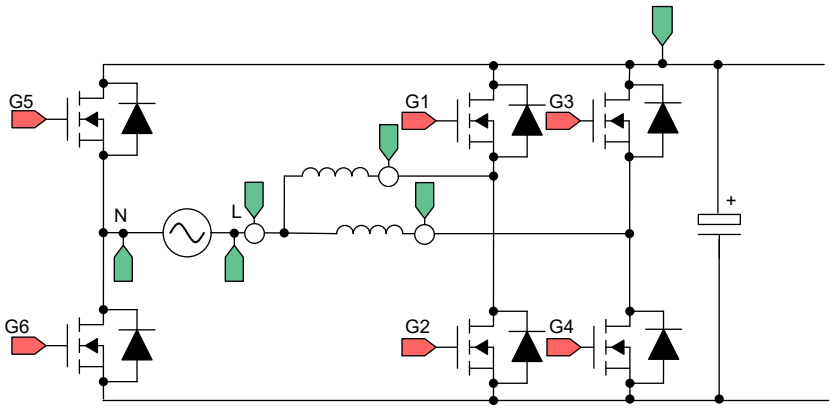



**Figure 3-4. TIDM-02013 PFC Block Diagram**

## 3.5 System Design Theory

### 3.5.1 PWM

Figure 3-5 shows a simplified diagram of a single phase of the interleaved TTPL PFC topology. To control this rectifier, the duty cycle is controlled to regulate the voltage directly. This regulation is possible if the software variable Duty, or D, is set so that when it is equal to 1, Q3 is always ON, and the setting makes the voltage $V_{xiN}$ equal to the $V_{bus}$ voltage. When Duty is set to 0, Q3 never turns on, and Q4 is always connected to DC bus negative, which makes the voltage go to 0.

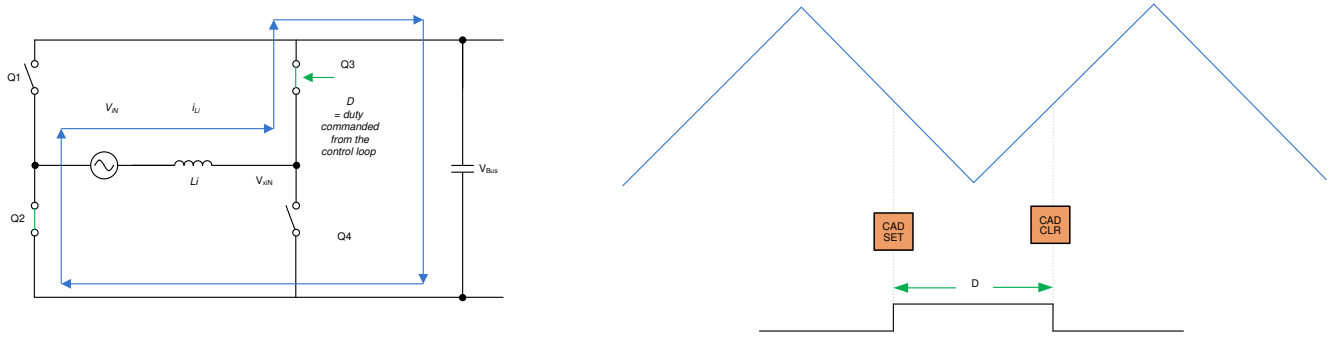



**Figure 3-5. Single-Phase Diagram of TTPL PFC**

### 3.5.2 Current Loop Model

To understand the current loop model, first look at the inductor current closely. In Figure 3-5, the duty cycle (D) is provided to the PWM modulator, which is connected to the switch Q3 and Q4. From here, Equation 6 is written as:

$$V_{xiN} = D \times V_{bus} \quad (6)$$

**Note**

When D is set to 1, Q3 is on all of the time, and when D is 0, Q3 is off all of the time.

To modulate the current through the inductor, the voltage $V_{xiN}$ is regulated using the duty cycle control of Q3 and Q4 switches. It is assumed that the direction of current is positive in the direction from the AC line into the rectifier and that the grid is fairly stiff when using the DC bus feedforward and the AC voltage feedforward. Figure 3-6 shows the simplified current loop, and the current loop plant model is written as Equation 7.

$$H_{p\_i} = \frac{i_{Li}^{*}}{D} = \frac{1}{K_{v\_gain}} \times K_{i\_gain} \times G_d \times \frac{1}{Z_i} \quad (7)$$

where,

- $K_{v\_gain}$ is the inverse of maximum bus voltage sensed, $\frac{1}{V_{busMaxSense}}$
- $K_{i\_gain}$ is the inverse of maximum AC current sensed, $\frac{1}{I_{AC\_MaxSense}}$
- $K_{i\_fltr}$ is the response of the RC filter connected from the current sensor to the ADC pin
- $G_d$ is the digital delay associated with the PWM update and digital control is the current command
- $i_{Li}^{*}$ is the current command

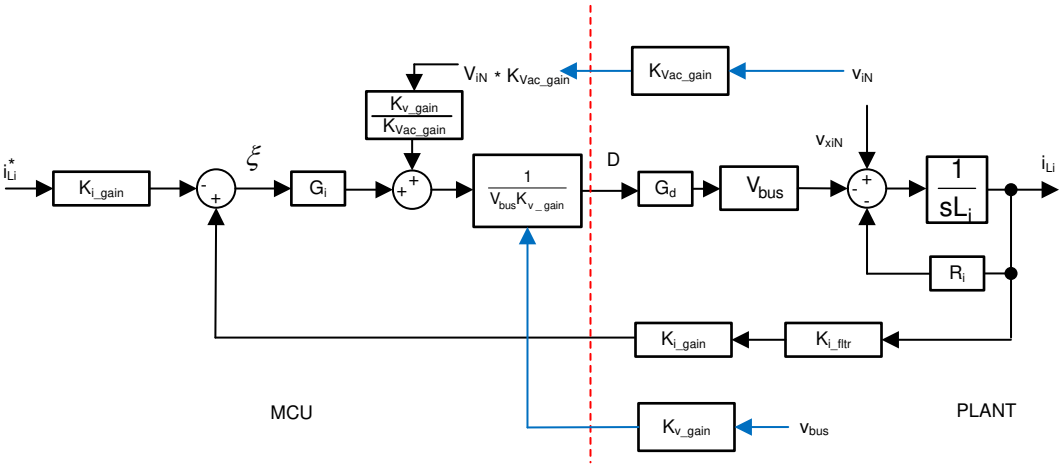



**Figure 3-6. Current Loop Control Model**

**Note**

The negative sign on the reference is used because the current loop is thought to be regulating the voltage, $V_{xiN}$. To increase the current, $V_{xiN}$ must be reduced—hence, the opposite sign for reference and feedback in Figure 3-6.

This current loop model is then used to design the current compensator. A simple proportional integral controller is used for the current loop.

In the case of two interleaved phases, the current is simply two times more as the same duty cycle is provided to each leg. Hence, the plant model is given as Equation 8.

$$H_{p\_i} = \frac{i_{Li}^{*}}{D} = 2 \times \frac{1}{K_{v\_gain}} \times K_{i\_gain} \times K_{i\_fltr} \times G_d \times \frac{1}{Z_i} \quad (8)$$

### 3.5.3 DC Bus Regulation Loop

The DC bus regulation loop is assumed to provide the power reference. The power reference is then divided by the square of the line voltage's RMS to provide the conductance, which is further multiplied by the line voltage giving the instantaneous current command.

Small signal model of the DC bus regulation loop is developed by linearizing Equation 9 around the operating point.

$$i_{DC}v_{bus} = \eta V_{Nrms} i_{Nrms} \rightarrow \hat{i}_{DC} = \eta \frac{\overline{V}_{Nrms}}{\overline{V}_{bus}} i_{Li} \tag{9}$$

For a resistive load, the bus voltage and current are related as shown in Equation 10:

$$\hat{V}_{bus} = \frac{R_L}{1 + sR_L C_o} \hat{i}_{DC} \tag{10}$$

The DC voltage regulation loop control model can be drawn as shown in Figure 3-7. An additional $V_{bus}$ feed forward is applied to make the control loop independent of the bus voltage. Therefore, the plant model for the bus control can be written as in Equation 11:

$$H_{p\_bus} = H_{load} * \eta * \frac{1}{K_{i\_gain}} * K_{v\_gain} * K_{v\_fltr} * \left( \frac{K_{v\_gain}}{K_{vac\_gain}} \right) \tag{11}$$

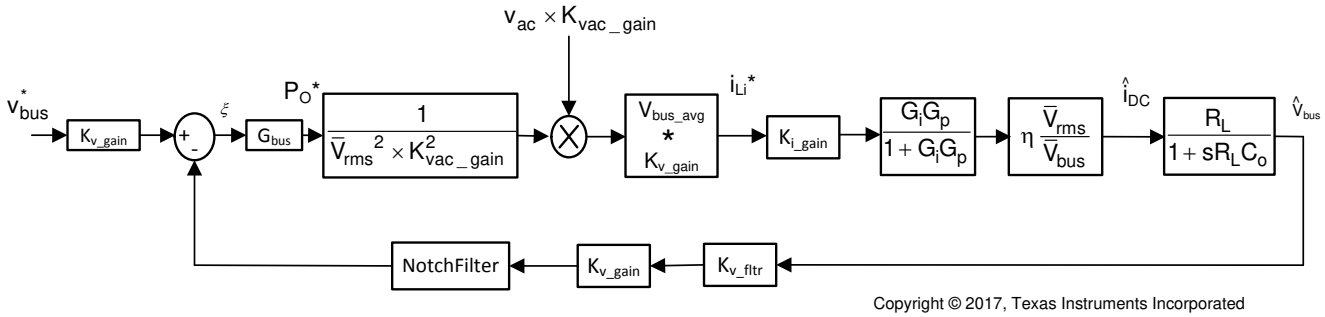




**Figure 3-7. DC Voltage Loop Control Model**

Using Figure 3-7, 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.

### 3.5.4 Soft Start Around Zero-Crossing for Eliminating or Reducing Current Spike

Zero-crossing current spikes present a challenging issue for TTPL PFC topologies. This issue is solved by implementing a soft-start scheme with a state machine to turn on and off switches in a particular sequence.

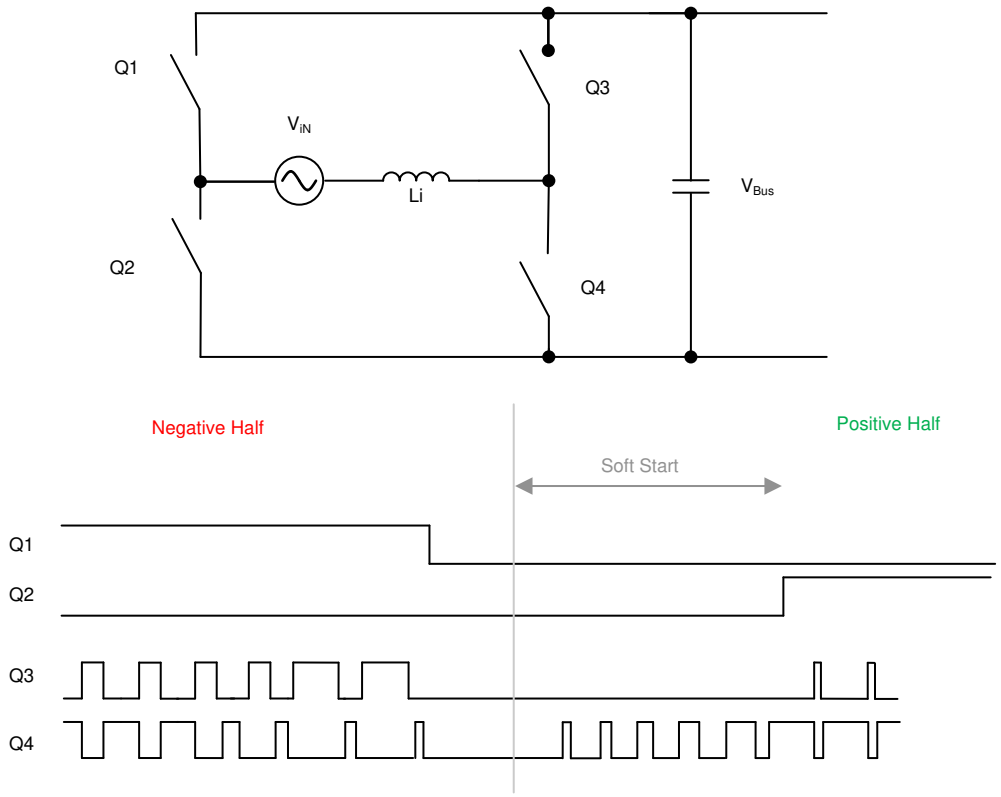



**Figure 3-8. PWM Sequence With Soft Starting to Reduce Current Spike at Zero-Crossing**

Figure 3-8 shows the switching sequence when the AC wave goes from negative to positive. During the negative half, Q1 is ON, Q3 is the active FET, and Q4 is the sync FET. During this time, the voltage across Q2 is the DC bus voltage. When the AC cycle changes, Q2 must be on 100% or close to 100%. If Q2 is turned ON immediately, a substantial positive spike results. Therefore, a soft-start sequence is used to turn Q4 ON as shown in Figure 3-8. The tuning of this soft start depends on the inductance value and other power stage parameters such as device $C_{oss}$.

Another reason for a negative current spike around zero-crossing is the relatively low AC voltage around the zero-crossing. When Q3 is turned ON, though the duty cycle is low, a high-voltage difference is applied and can result in a high negative current spike. Therefore, a sufficient delay is applied before Q3 starts switching back again.

Similarly, Q2 is turned on after some delay after the soft start has started.

### 3.5.5 Current Calculation

Select the input fuse, filter current ratings based on the max input current, which Equation 12 calculates as:

$$I_{inrms} = \frac{P_{out\_max}}{\eta \cdot V_{inrms} \cdot PF} = 28.2\ A \tag{12}$$

where,

- $P_{OUT\_MAX}$ is the maximum output power 6.6 kW
- η is the efficiency (assumed as 98.6%)
- $V_{IN\_RMS}$ is the input voltage RMS value (240 V)
- PF is the power factor (assumed as 0.99)

### 3.5.6 Inductor Calculation

The inductor plays important role in affecting system efficiency, current ripple, and overall size. It is always a balance between the efficiency and the power density. The inductance value is calculated based on the input voltage, output voltage, and worst case ripple.

The duty cycle is calculated as:

$$D = 1 - \frac{V_{in}}{V_{out}} \tag{13}$$

Calculation of the current ripple into inductor can be distinguished into three periods:

$$I_{ripple} = \left(\frac{V_{in}}{L} - 2 \times \frac{V_{out} - V_{in}}{L}\right) \times D \times T_s \quad \leftarrow For\ D \leq 1/3 \tag{14}$$

$$I_{ripple} = \left(\frac{2 \times V_{in}}{L} - \frac{V_{out} - V_{in}}{L}\right) \times \left(D - \frac{1}{3}\right) \times T_s \quad \leftarrow For\ 1/3 < D < 2/3 \tag{15}$$

$$I_{ripple} = \left(\frac{3 \times V_{in}}{L}\right) \times \left(D - \frac{2}{3}\right) \times T_s \quad \leftarrow For\ D \geq 2/3 \tag{16}$$

In the worst cases, the equation becomes:

$$I_{ripple} = \frac{V_{out} \times T_s}{12 \times L} \tag{17}$$

This design targets at achieving 10% current ripple at maximum input power and maximum AC current:

$$I_{ripple} < 10\,\% \times \frac{\sqrt{2} \times P_{out\_max}}{V_{in\_max} \times \eta} \tag{18}$$

where,

- Pout_max is the maximum output power
- η is the efficiency
- $V_{in\_max}$ is the maximum input voltage

As a result, the inductance is calculated as 126 µH at 12 A RMS current.

### 3.5.7 Output Capacitor Calculation

Due to the input double-line frequency ripple on the DC link capacitor, its capacitance is mainly determined by the output voltage ripple, as calculated in Equation 19:

$$C_{out(\min)} \geq \frac{P_{out}/V_{out}}{4 \cdot \pi \cdot f_{line\_\min} \cdot V_{ripple\_\max}} = 860\mu F \tag{19}$$

where,

- $P_{OUT}$ is the output power
- $V_{OUT}$ is the output voltage
- $f_{LINE\_MIN}$ is the minimum line frequency
- $V_{RIPPLE}$ is the output ripple

The actual capacitor used is 1410 µF(3 x 470uF).

### 3.5.8 Current and Voltage Sense

The Hall-effect sensor ACS733KLATR-40AB-T is used for the total input current sensing, as Figure 3-10 shows. The OPA320-based amplifier circuit tunes the low output voltage of the sensor to a higher level and sends this voltage to the controller ADC pin. The ACS733KLATR-40AB-T device senses each individual interleaved phase's current which allows phase current balancing.

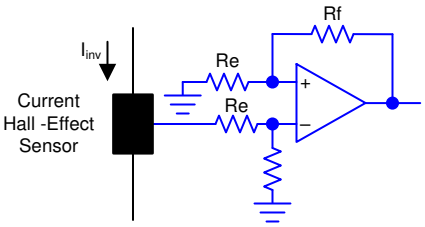



**Figure 3-9. Hall Effector Sensor Signal Conditioning Circuit**

The output voltage from the signal conditioning circuit is scaled to match the ADC range using the circuit as shown in Figure 3-9. The voltage is calculated as:

$$I_{out} = \frac{R_f}{R_e}\left(I_{inv} \times \frac{V_{nominal}}{I_{nominal\_max}} + V_{offset}\right) \quad (20)$$

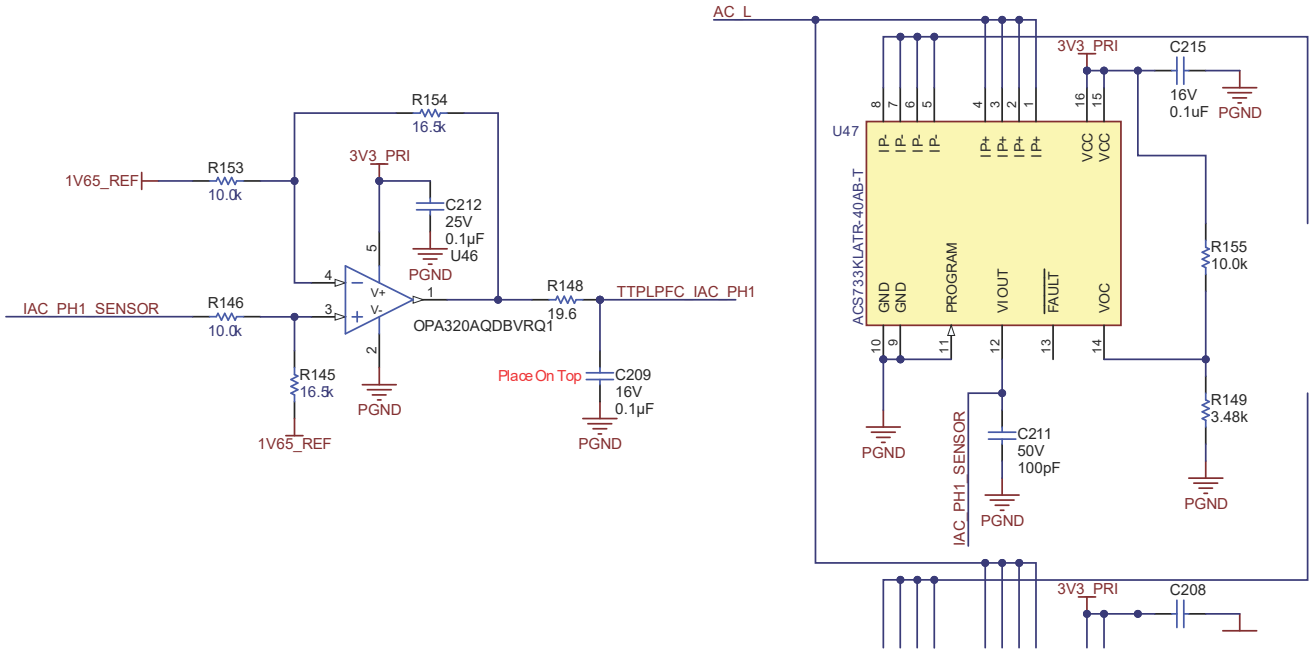



**Figure 3-10. Schematic of Current Sensing**

The input AC voltage is sensed differentially by sensing the line, and the neutral inputs refer to the control ground separately with two voltage dividers, as Figure 3-11 shows. The control ground is the DC link negative terminal; therefore, a single voltage divider can be used to sense the DC bus voltage. An RC filter filters the signals before connecting to the controller. A common RC filter is used for all the sensing signals on this design.

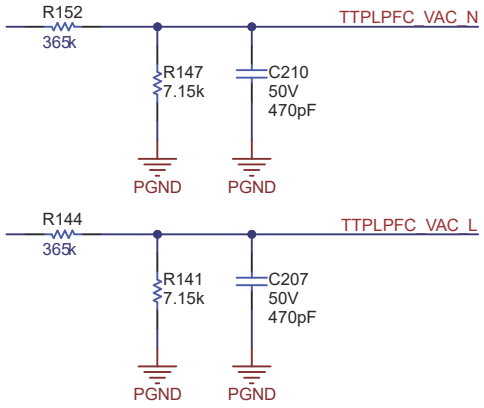



**Figure 3-11. Schematic of Voltage Dividers for AC Input Voltage Sense**

# 4 Highlighted Products

## 4.1 C2000 MCU TMS320F28003x

C2000™ 32-bit microcontrollers are optimized for processing, sensing, and actuation to improve closed-loop performance in real-time control applications such as industrial motor drives; solar inverters and digital power; electrical vehicles and transportation; motor control; and sensing and signal processing.

The TMS320F28003x (F28003x) is a powerful 32-bit floating-point microcontroller unit (MCU) that lets designers incorporate crucial control peripherals, differentiated analog, and nonvolatile memory on a single device.

The CLA allows significant offloading of common tasks from the main C28x CPU. The CLA is an independent 32-bit floating-point math accelerator that executes in parallel with the CPU. Additionally, the CLA has its own dedicated memory resources and it can directly access the key peripherals that are required in a typical control system. Support of a subset of ANSI C is standard, as are key features like hardware breakpoints and hardware task-switching.

High-performance analog blocks are integrated on the F28003x MCU to further enable system consolidation. Three separate 12-bit ADCs provide precise and efficient management of multiple analog signals, which ultimately boosts system throughput. Four analog comparator modules provide continuous monitoring of input voltage levels for trip conditions.

The TMS320C2000™ devices contain industry-leading control peripherals with frequency-independent Enhanced Pulse Width Modulator/High-Resolution Pulse Width Modulator (ePWM/HRPWM) and Enhanced Capture (eCAP) module allow for a best-in-class level of control to the system. The built-in Sigma-Delta Filter Module (SDFM) allows for seamless integration of an oversampling sigma-delta modulator across an isolation barrier.

Connectivity is supported through various industry-standard communication ports [such as Serial Peripheral Interface (SPI); Serial Communication Interface (SCI); Inter-integrated Circuit (I2C); and Controller Area Network (CAN)] and offers multiple multiplexing options for optimal signal placement in a variety of applications. New to the C2000™ platform is the fully compliant Power-Management Bus (PMBus). Additionally, in an industry first, the Fast Serial Interface (FSI) enables high-speed, robust communication to complement the rich set of peripherals that are embedded in the device.

A specially enabled device variant, TMS320F28003xC, allows access to the Configurable Logic Block (CLB) for additional interfacing features. See the Device Comparison table in the TMS320F28003x microcontrollers data manual for more information.

The Embedded Real-Time Analysis and Diagnostic (ERAD) module enhances the debug and system analysis capabilities of the device by providing additional hardware breakpoints and counters for profiling.

Following is the subset of features of the C2000 MCU that are highlighted on this design to enable control of high-frequency CLLLC topology:

1. **High-resolution PWM**: With picosecond resolutions possible, the ePWM module on the C2000 MCU can generate high-frequency PWM with accuracy. With the Type-4 PWM high-resolution period control, high-resolution duty control, high-resolution dead-band control, along with high-resolution phase shift control, is possible. This enables generation of balanced pulses for the resonant tank excitation and is an enabling feature for high-frequency power converters.
2. **Comparator Subsystem (CMPSS)** with ePWM for active synchronous rectification: Active synchronous rectification enables higher efficiency and for high-frequency resonant converters that operate both below and above the resonant point, it is a necessary feature for the topology. The C2000 MCUs' integrated CMPSS enables generation of the active synchronous rectification pulses by using the integrated comparators and the integrated Digital-to-Analog Converters (DACs). (See Section 2.2.2.4.)
3. **Blanking window**: Due to noise, which is unavoidable in switching converters, the blanking window feature is used to suppress the CMPSS output during noisy switching events. This blanking window is provided by the ePWM time base and can be applied at different times during the PWM cycle. (See Section 2.2.3.)
4. **X-Bar**: Multiple trip sources are possible in a power converter. The X-bar enables combining different trip actions from either different CMPSS or from GPIO to generate trip behavior desired in the ePWM without need of external logic.
5. **Control Law Accelerator (CLA)** enables integration of control of multiple topologies on a single controller. The software provided with this design provides the option to run the control loop on the CLA or the C28x.

6. **Global Link feature in PWM module** enables update to multiple PWMs through a single write, which reduces the CPU burden and enables higher-frequency converters to be controlled easily.

### 4.2 LMG352xR30-Q1

The LMG352xR030-Q1 is an automotive qualified 650-V 30-mΩ GaN FET with integrated driver, protection and temperature reporting. The integrated driver enables switching speed up to 150 V/ns. TI's integrated precision gate bias results in higher switching SOA compared to discrete external gate drivers. This integration, combined with a low-inductance package, delivers clean switching and minimal ringing in hard-switching power supply topologies. Other features, including adjustable gate drive strength for EMI control, overtemperature, and robust overcurrent protection with fault indication, provide optimized BOM cost, board size, and footprint. Advanced power management features include digital temperature reporting; the temperature of the GaN FET is reported through a variable duty cycle PWM output, which enables the system to optimally manage loading.

### 4.3 UCC21222-Q1

The UCC21222 device is an isolated dual channel gate driver with programmable dead time. It is designed with 4-A peak-source and 6-A peak-sink current to drive power MOSFET, IGBT, and GaN transistors.

The UCC21222 device can be configured as two low-side drivers, two high-side drivers, or a half-bridge driver. 5ns delay matching performance allows two outputs to be paralleled, doubling the drive strength for heavy load conditions without risk of internal shoot-through.

The input side is isolated from the two output drivers by a 3.0-$kV_{RMS}$ isolation barrier, with a minimum of 100-V/ns common-mode transient immunity (CMTI).

Resistor programmable dead time gives the capability to adjust dead time for system constraints to improve efficiency and prevent output overlap. Other protection features include: Disable feature to shut down both outputs simultaneously when DIS is set high, integrated deglitch filter that rejects input transients shorter than 5-ns, and negative voltage handling for up to -2-V spikes for 200-ns on input and output pins. All supplies have UVLO protection.

### 4.4 AMC3330-Q1

The AMC3330 is a precision, isolated amplifier with a fully integrated, isolated DC/DC converter that allows single-supply operation from the low-side of the device. The reinforced capacitive isolation barrier is certified according to VDE V 0884-11 and UL1577 and separates sections of the system that operate on different common-mode voltage levels and protects low-voltage domains from damage. The input of the AMC3330 is optimized for direct connection to high-impedance, voltage-signal sources such as a resistor-divider network to sense high-voltage signals. The integrated isolated DC/DC converter allows measurement of non-ground-referenced signals and makes the device a unique design for noisy, space-constrained applications.

### 4.5 AMC3302-Q1

The AMC3302 is a precision, isolated amplifier optimized for shunt-based current measurements. The fully integrated, isolated DC/DC converter allows single-supply operation from the low-side of the device, which makes the device a unique solution for space-constrained applications. The reinforced capacitive isolation barrier is certified according to VDE V 0884-11 and UL1577 and supports a working voltage of up to 1.2 $kV_{RMS}$. The isolation barrier separates parts of the system that operate on different common-mode voltage levels and protects the low-voltage side from hazardous voltages and damage. The input of the AMC3302 is optimized for direct connection to a low-impedance shunt resistor or another low-impedance voltage source with low signal levels. The excellent DC accuracy and low temperature drift supports accurate current measurements over the extended industrial temperature range from –40°C to +125°C.

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

## 5.1 Required Hardware and Software

This section details the hardware and explains the different sections on the board and how to set them up for the experiments as outlined in this design guide.

### 5.1.1 Hardware Settings

The design follows a High-Speed Edge Card (HSEC) control card concept, and 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 microcontroller are listed in Table 5-1. Figure 5-1 shows the key power stage and connectors on the reference design. Table 5-3 lists the key connectors and their functions.

1. Make sure no power source is connected to the board.
2. Insert the control card in the J25 slot.
3. Connect a power source (*but do not power up*) for the 12V bias supplies (+12 V, 2 A) at the J15 shown in Figure 5-1.
4. Now, switch the power source on for the bias supply. A green LED on the control card will light up. This indicates the C2000 MCU device is powered. Note: The bias for the microcontroller is separated from the power stage; this enables a safe bring up of the system in this set of instructions.
5. To connect JTAG, use a USB cable from the control card and connect it to a host computer.
6. For operation of the TTPLPFC stage an AC input must be connected to J33 (90V - 264V). For testing a >10kW supply has been used, however a clean and stable lower-rated supply can be used in the case where only low-power tests are being conducted.
7. For stand alone operation of the PFC stage a load may be connected to J37 and J38, alternatively the CLLLC can be used to load the PFC stage.
8. For stand alone operation of the CLLLC stage a DC power supply (400V) may be connected to VBUS at J15. If this is done the TTPLPFC should not be started in software and the AC source described in step 6 above should not be connected.
9. A load should be connected to the secondary side of the CLLLC converter when in use. J7 and J10 can be used to connect such a load.
10. When operating both the PFC and DCDC stages connect an AC supply as in Step 6 above and a load as in step 9 above. No connection to VBUS is needed however a current bleed resistor may be helpful to ensure excess voltage can be quickly bled off after execution of the OBC.
11. Current and voltage probes can be connected to observe the tank current at primary and secondary. Optionally, a power meter can be connected to measure the efficiency.

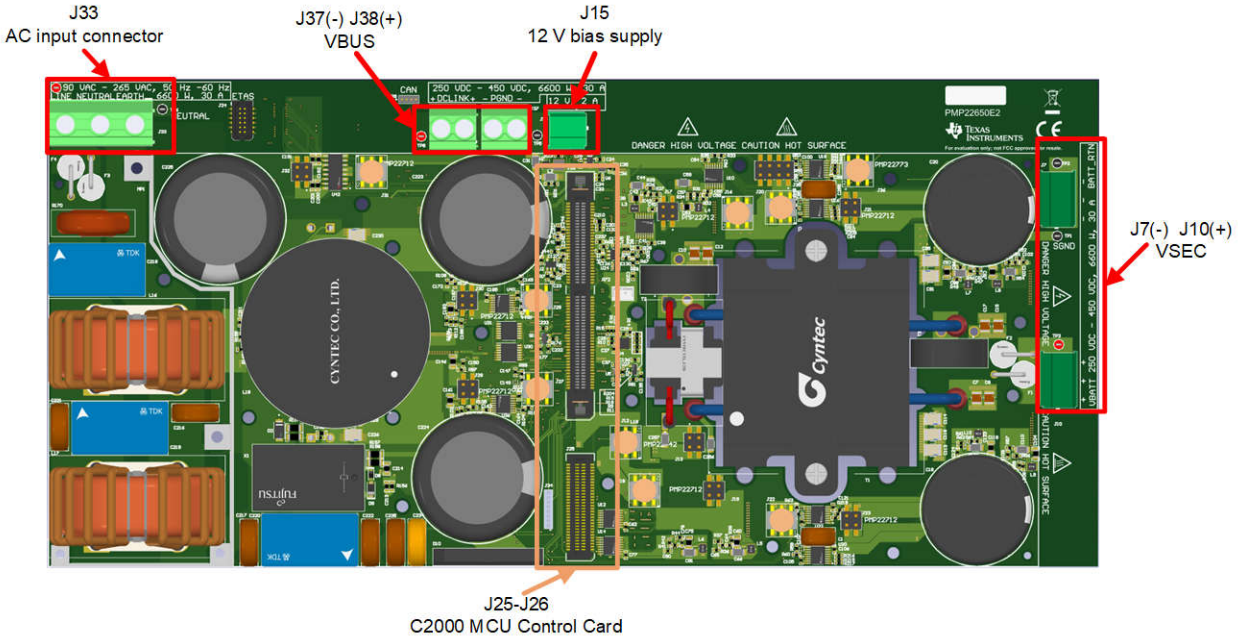



**Figure 5-1. Board Overview**

There are 7 bias supply daughter cards required shown in red

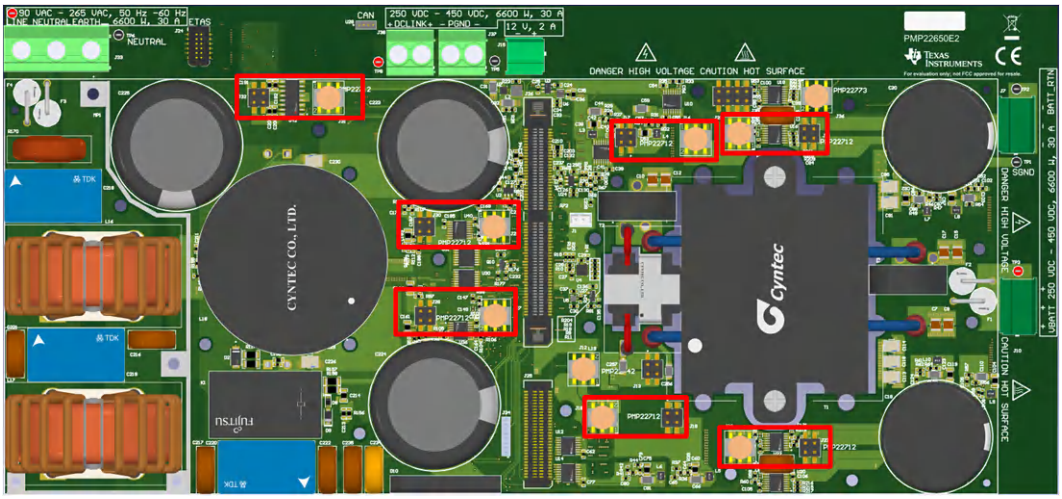


**Figure 5-2. PMP22712 - bias supplies**

There is one Feedback isolation daughter card PMP22773 required indicated in red

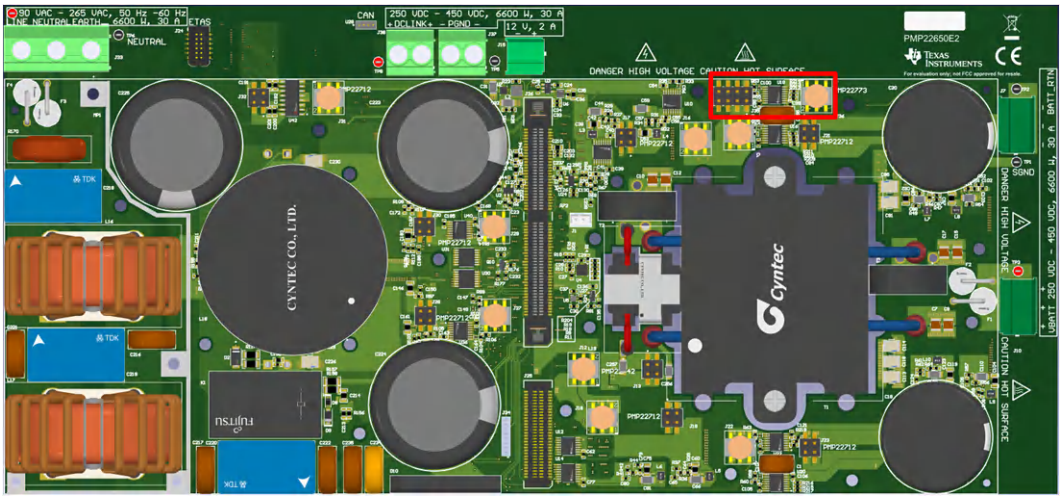



**Figure 5-3. PMP22773 – Feedback Isolation Daughter Card**

**Table 5-1. Key Digital Pin Assignments**

| Signal name | HSEC Pin Number | F28003x peripheral |
|---|---|---|
| SYSTEM ISR Trigger | - | ECAP1 |
| CLLLC_CONTROL_OUTPUT_DAC_PIN | 14 | DACA |
| CLLLC_PRIM_LEG1_H/L | 49/ 51 | EPWM1 (A/B) |
| CLLLC_PRIM_LEG2_H/L | 53/ 55 | EPWM2 (A/B) |
| CLLLC_SEC_LEG1_H/L | 50/ 52 | EPWM3 (A/B) |
| CLLLC_SEC_LEG2_H/L | 54/ 56 | EPWM4 (A/B) |
| CLLLC_FAULTn | 74 | GPIO-23 → INPUTXBAR2 |
| CLLLC_LC_CHANGE | 62 | GPIO-14 |
| CLLLC_SEC_SIDE_DIAG | 80 | GPIO-30 |
| TTPLPFC_LOW_FREQ_H/L | 57/ 59 | EPWM5 (A/B) |
| TTPLPFC_HIGH_FREQ_PH1_H/L | 61/ 63 | EPWM6 (A/B) |
| TTPLPFC_HIGH_FREQ_PH2_H/L | 58/ 60 | EPWM7 (A/B) |
| TTPLPFC_FAULTn | 72 | GPIO-22 → INPUTXBAR1 |
| TTPLPFC_INRUSH_RELAY_CTRL | 64 | GPIO-15 |
| ERRORSTSn | 102 | GPIO55 |
| SYSTEM_WATCHDOG_OUT<br>SYSTEM_WATCHDOG_DISABLE<br>SYSTEM_PMIC_SPI (resv)<br>SYSTEM_PMIC_SPI (resv) | 75<br>77<br>79<br>81 | GPIO24<br>GPIO25(Resistor option)<br>GPIO26(Resistor option)<br>GPIO27(Resistor option) |
| SYSTEM_DISABLE_FET_SUPPLY | 85 | GPIO32 |
| SYSTEM_TEMP_MUX_OUT1<br>SYSTEM_TEMP_MUX_OUT2 | 91<br>96 | GPIO41 -> ECAP2 → INPUTXBAR3<br>GPIO60 -> ECAP3 → INPUTXBAR4 |
| SYSTEM_TEMP_MUX_SEL_1-3 | 93<br>94<br>95 | GPIO47<br>GPIO58<br>GPIO59 |
| SYSTEM_PROFILING1-3 | 89<br>92<br>101 | GPIO40<br>GPIO44<br>GPIO49 |
| FSI_TX_D0<br>FSI_TX_D1<br>FSI_TX_CLK | 101<br>103<br>105 | GPIO-49/FSITXA_D0<br>GPIO-50/FSITXA_D1<br>GPIO-51/FSITXA_CLK |

**Table 5-1. Key Digital Pin Assignments (continued)**

| Signal name | HSEC Pin Number | F28003x peripheral |
|---|---|---|
| LED1 | 82 | GPIO-31 → LED1 |
| LED2 | 86 | GPIO-34 → LED2 (SFRA) |

This table describes the sampling scheme for the reference design. Across the top each column represents one independent ADC. Each ADC acts entirely independently from the others. Each signal is assigned one or more Start Of Conversions (SOCs). Each SOC represents one independent reading of that channel, for example TTPLPFC_IAC_PH1 is assigned SOC0 and SOC1 within ADCA. This means that this signal will be sampled twice every cycle, once triggered by ePWM6_SOCA and once triggered by ePWM6_SOCB. Since this trigger is running at 120kHz this signal is effectively oversampled by a factor of two times during each 120kHz sampling period. Similarly CLLLC_ISEC is oversampled 11 times, and CLLLC_IPRIM is not oversampled. The table also indicates several low-frequency sampled signals which can be seen that these signals use a different SOC signal. Finally since a round robin counter is used to process the SOCs in numeric order the table reads as a timeline from top to bottom in the order of sampling.

**Table 5-2. Key Analog Signals**

| | ADC-A | ADC-B | ADC-C |
|---|---|---|---|
| **Highest Priority Signals (120kHz)** | **TTPLPFC_IAC_PH1** (A2, CMPSS1)<br>SOC0 → ADC_TRIGGER_EPWM6_SOCA<br>SOC1 → ADC_TRIGGER_EPWM6_SOCB | **TTPLPFC_IAC_PH2** (B12, CMPSS3)<br>SOC0 → ADC_TRIGGER_EPWM6_SOCA<br>SOC1 → ADC_TRIGGER_EPWM6_SOCB | **TTPLPFC_VAC** (C7)<br>SOC0 → ADC_TRIGGER_EPWM6_SOCA<br>SOC1 → ADC_TRIGGER_EPWM6_SOCB |
| | **CLLLC_ISEC** (A5, CMPSS2)<br>SOC2 → ADC_TRIGGER_EPWM6_SOCA<br>SOC3 → ADC_TRIGGER_EPWM6_SOCA<br>SOC4 → ADC_TRIGGER_EPWM6_SOCA<br>SOC5 → ADC_TRIGGER_EPWM6_SOCA<br>SOC6 → ADC_TRIGGER_EPWM6_SOCA<br>SOC7 →ADC_TRIGGER_EPWM6_SOCA<br>SOC8 → ADC_TRIGGER_EPWM6_SOCB<br>SOC9 → ADC_TRIGGER_EPWM6_SOCB<br>SOC10 → ADC_TRIGGER_EPWM6_SOCB<br>SOC11 → ADC_TRIGGER_EPWM6_SOCB<br>SOC12 → ADC_TRIGGER_EPWM6_SOCB | **TTPLPFC_VBUS / CLLLC_VBUS** (B4)<br>SOC2 → ADC_TRIGGER_EPWM6_SOCA<br>SOC3 → ADC_TRIGGER_EPWM6_SOCB<br>SOC4 → ADC_TRIGGER_EPWM7_SOCA<br>SOC5 → ADC_TRIGGER_EPWM7_SOCB | **CLLLC_VSEC** (C11, CMPSS2)<br>SOC2 → ADC_TRIGGER_EPWM6_SOCA<br>SOC3 → ADC_TRIGGER_EPWM6_SOCA<br>SOC4 → ADC_TRIGGER_EPWM6_SOCA<br>SOC5 → ADC_TRIGGER_EPWM6_SOCA<br>SOC6 → ADC_TRIGGER_EPWM6_SOCA<br>SOC7 → ADC_TRIGGER_EPWM6_SOCA<br>SOC8 → ADC_TRIGGER_EPWM6_SOCA<br>SOC9 → ADC_TRIGGER_EPWM6_SOCA<br>SOC10 → ADC_TRIGGER_EPWM6_SOCA<br>SOC11 → ADC_TRIGGER_EPWM6_SOCA<br>SOC12 → ADC_TRIGGER_EPWM6_SOCA |
| | **CLLLC_IPRIM** (A9, CMPSS2)<br>SOC13→ADC_TRIGGER_EPWM1_SOCA | | |
| **Low-Frequency Sampling Signals (10kHz)** | **TTPLPFC_VAC_L** (A4)<br>SOC14 →ADC_TRIGGER_CPU1_TINT2 | **TTPLPFC_VAC_N** (B2)<br>SOC10 → ADC_TRIGGER_CPU1_TINT2 | **TTPLPFC_VBUS2** (C10, CMPSS2)<br>SOC14 → ADC_TRIGGER_CPU1_TINT2 |
| | **SYSTEM_ TEMP_1** (A11)<br>SOC15 → ADC_TRIGGER_CPU1_TINT2 | **SYSTEM_VREF_1_65** (B5)<br>SOC11 →ADC_TRIGGER_CPU1_TINT2 | **CLLLC_VSEC** (C11, CMPSS2)<br>VSEC13 → SOC15 →ADC_TRIGGER_CPU1_TINT2 |

**Table 5-2. Key Analog Signals (continued)**

| | ADC-A | ADC-B | ADC-C |
|---|---|---|---|
| **Not Sampled, CMPSS only** | | **CLLLC_IPRIM_TANK** (A12/C5, CMPSS2) | **CLLLC_ISEC_TANK** (C1, CMPSS4) |

**Table 5-3. Key Connectors and Their Function**

| CONNECTOR NAME | FUNCTION |
|---|---|
| J33 | AC input |
| J37/J38 | VBUS connection; PFC output, DCDC VPRIM |
| J7/J10 | DCDC output connection; DCDC VSEC |
| J15 | 12 V 2 A power supply |
| J25/J26 | HSEC control card connector slot |

#### *5.1.1.1 Control Card Settings*

Certain settings on the device control card are required to communicate over JTAG and use the isolated UART port. The user must also provide a correct ADC reference voltage. The following are the required settings for revision A of the F280039C control card. The user can also refer to the information sheet located inside C2000Ware at \c2000ware\boards\controlcards\TMDSCNCD280039C or alternatively get it from the *F280039 controlCARD Information Guide*.

1. S1:A on the control card must be set on both ends to the *ON (Left)* position to enable JTAG connection to the device and UART connection for the SFRA GUI. If this switch is set to *OFF (Right)*, the user cannot use the isolated JTAG built-in on the control card nor can the SFRA GUI communicate to the device.
2. J1:A is the connector for the USB cable that is used to communicate to the device from a host PC on which Code Composer Studio™ Integrated Development Environment (IDE) runs.
3. A 3.3-V reference is desired for the control-loop tuning on this design. Internal reference of the F28003x is used, and for this, the S3 switch must be moved to the top position(that is, pointing to INT).
4. A capacitor is connected between the isolated grounds on the control card, C7:A. It must be removed before connecting HV power.

For best performance RC filter caps are required to be added to several ADC channels. Each of the component designators are clearly marked in the silkscreen and assembly plots can be found in C2000Ware for TMDSCNCD280039C. \c2000ware\boards\controlcards\TMDSCNCD280039C

| SIGNAL | CAPACITOR COMPONENT DESIGNATOR | CAPACITOR VALUE |
|---|---|---|
| TTPLPFC_VBUS | C46 | 1 µF 0603 |
| CLLLC_VSEC | C43 | 1 µF 0603 |
| TTPLPFC_VAC | C32 | 1 µF 0603 |
| TTPLPFC_IAC_PH1 | C31 | 1 µF 0603 |
| TTPLPFC_IAC_PH2 | C49 | 1 µF 0603 |
| CLLLC_ISEC_TANK | C50 | 560 pF 0603 |
| CLLLC_ISEC | C34 | 1 µF 0603 |

### 5.1.2 Software

The software of this design is available in the DigitalPower Software Development Kit (SDK) for C2000 MCUs (C2000WARE-DIGITALPOWER-SDK).

#### *5.1.2.1 Opening the Project Inside Code Composer Studio*

To start:

1. Install Code Composer Studio from the Code Composer Studio (CCS) Integrated Development Environment (IDE) tools folder. Version 12.0 or above is recommended.
2. Install C2000WARE-DIGITAL-POWER-SDK in one of two ways:

- Through the C2000Ware Digital Power SDK tools folder
- Go to CCS and under View → Resource Explorer. Under the TI Resource Explorer, go to C2000WARE-DIGITAL-POWER-SDK, and click on the install button.

3. When installation completes, close CCS, and open a new workspace. CCS automatically detects powerSUITE. A restart of CCS may be required for the change to be effective.

**Note**

powerSUITE is installed with the SDK by default.

The firmware project can now be imported as follows:

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 <SDK>/solutions/tidm_02013/f28003x/ccs.

A project spec will appear, clicking on this will create a self-contained folder of the project with all the dependencies inside it.

#### *5.1.2.2 Project Structure*

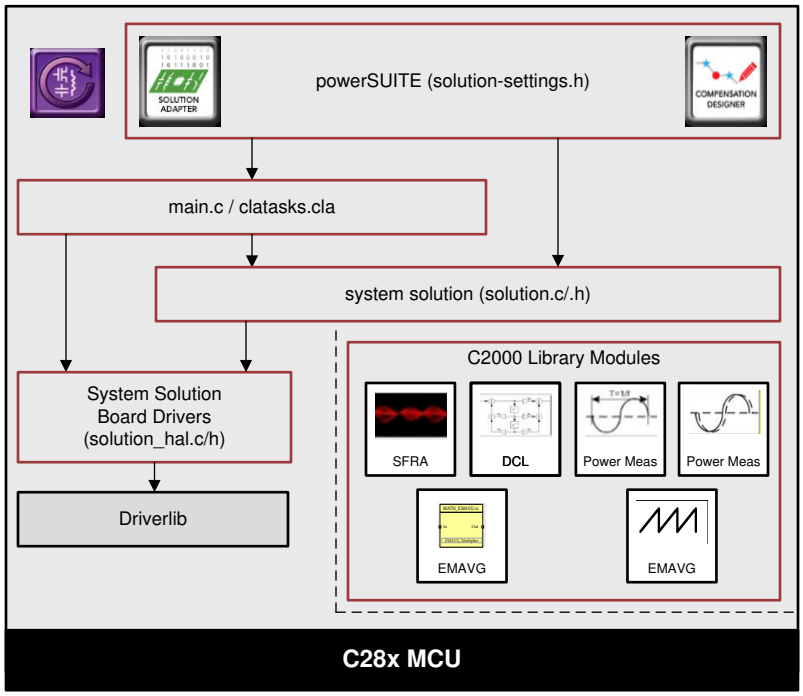



**Figure 5-4. Project Structure Overview**

The general structure of the project is shown in Figure 5-4. Once the project is imported, the Project Explorer will appear inside CCS as shown in Figure 5-5.

Solution-specific and device-independent files that consist of the core algorithmic code are in <*solution*>.*c/h*. For example *TTPLPFC.c* or *CLLLC.h*.

Board-specific and device-specific files are in <*solution*>*_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 <*solution*>*-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, there are three <solution> monikers *obc_7_4kw*, *clllc*, and *ttplpfc*. Please note that to maintain maximum flexibility we have chosen to keep the CLLLC code base and the TTPLPFC code base as independent

as possible while the *obc_7_4kw* files were added where needed to contain settings independent to the TTPLPFC and CLLLC. This allows the end user to operate each stage independently and incorporate different topologies for either the PFC or DCDC stages easily in their final design if desired.

The <*solution*>*_settings.h* file contains code configuration settings like which lab to build. While the <*solution*>*_user_settings.h* contains the board level configurations such as #defines for ADC mapping, GPIOs etc.

*solution.js* files contain a script file which can help populate relevant variables to observer during the execution of each lab. To use these scripts open the scripting console(View - Scripting Console). Paste the contents of the solution.js file into the scripting console and press enter. This will populate the expressions window for debug later.

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 CLLLC name (for example, CLLLC_vSecSensed_pu). This naming convention lets the user combine different solutions while avoiding naming conflicts.

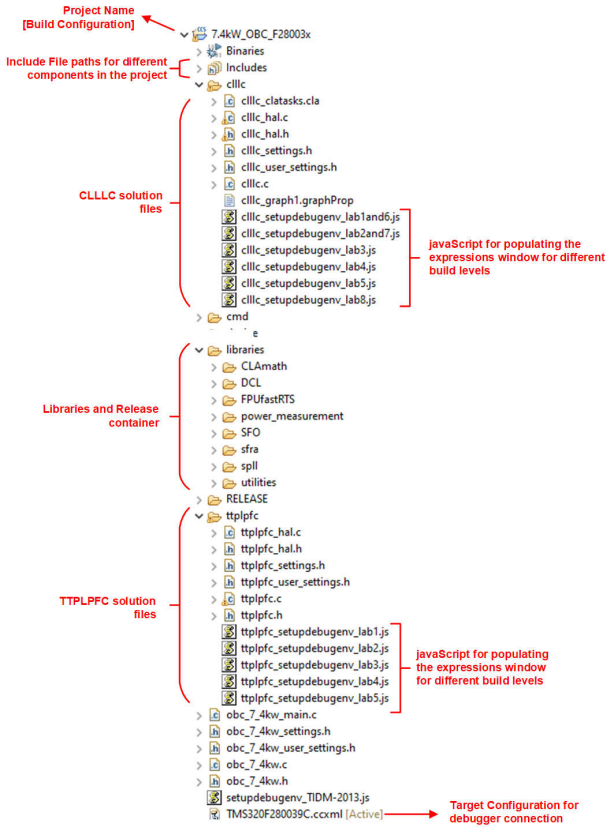



**Figure 5-5. Project Explorer View of the CLLLC Project**

The OBC project consists of three ISRs (ISR1, ISR2, and ISR3) running on 2 cores, the C28x core and the CLA core. Using an ePWM for the ISR1 trigger, an eCAP for ISR2's trigger, and ADC for ISR3's trigger allows the ISR priority to be controlled entirely by hardware. Table 5-4 shows how each ISR is partitioned and its tasks.

**Table 5-4. ISR Partitions and Tasks**

| ISR | Trigger source | C28x | CLA |
|---|---|---|---|
| ISR1 (120kHz) | ePWM | N/A | Update CLLLC PWM values |
| ISR2 (120kHz) | eCAP | PFC current loop | CLLLC control code, enable ISR1 |
| ISR3 (10kHz) | ADC | PFC voltage loop, instrumentation | N/A |

**ISR1** is the fastest and non-nestable ISR reserved for PWM updates and is ran entirely on CLA. ISR1 is triggered by the PRIM_LEG1_PWM_BASE → EPWM_INT_TBCTR_U_CMPC event. In general, this interrupt is disabled by writing a value to CMPC, which is greater than all possible TBPRD register values for PRIM_LEG1_PWM_BASE. This is done through the CLLLC_HAL_setupISR1Trigger function. Following are the defines that are related to this ISR.

```
#define CLLLC_ISR1_PERIPHERAL_TRIG_BASE CLLLC_PRIM_LEG1_PWM_BASE
#define CLLLC_ISR1_TRIG INT_EPWM1
#define CLLLC_ISR1_PIE_GROUP INTERRUPT_ACK_GROUP3
```

```
#define CLLLC_ISR1_TRIG_CLA CLA_TRIGGER_EPWM1INT
```

**ISR2** is split across both cores. This allows a simple modular code segregation for the TTPLPFC and CLLLC code. Both the ISR2 ran on the C28x and the ISR2 ran on the CLA are triggered by the same source and operate in tandem. The C28x core run task related to the TTPLPFC while the CLA core runs task related to the CLLLC's execution.

**ISR2** is responsible for writing a valid value to trigger ISR1 by a write to CMPC when ISR1 is desired. (Note: The CMPC is not tied to the global load mechanism to enable this. Also, shadow load of CMPC is disabled.) The CMPC value can be adjusted to get the desired timing from the ISR1. Each time ISR1 is enabled, it triggers two times. In the first ISR1, PWM registers are updated and a sync is enabled. In the second ISR1, the PWM sync is disabled and CMPC is set to a value such that ISR1 does not trigger again. For simplicity the software diagrams and structure only show ISR1 that is triggered the first time.

**ISR2** is periodically triggered at ISR2_FREQUENCY. A spare CAP module is used to generate the time base and trigger the interrupt. A spare ePWM module is also configured with the same time base and is used to trigger the ADC conversions. ISR2 is responsible for running the control law and calculating the clock ticks required for the PWM. Once the writes to the shadow registers are complete, the ISR2 enables the ISR1 trigger by writing a valid value (that is, one that is less than the current TBPRD register) to the CMPC register. ISR2 has two variants ISR2_primToSecPowerFlow and ISR2_secToPrimPowerFlow, one for the power flow from primary to secondary side and other for secondary to primary. This is done to optimize CPU cycles when controlling power flow in different directions. For simplicity of explanation either of them are just referred to as ISR2 in respective labs. Depending on the timing, the ISR1 may nest ISR2 for the update writes, which are very timing-critical. Following are the defines that are related to this ISR.

```
#define CLLLC_ISR2_ECAP_BASE ECAP1_BASE
#define CLLLC_ISR2_PWM_BASE EPWM5_BASE
#define CLLLC_ISR2_TRIG INT_ECAP1
#define CLLLC_ISR2_PIE_GROUP INTERRUPT_ACK_GROUP4
```

```
#define CLLLC_ISR2_TRIG_CLA CLA_TRIGGER_ADCA2
```

**ISR3** is ran entirely on the C28x core and is triggered by ADCINT2. ADCINT2 is initiated by a conversion that is started using a CPU timer. It is used to run the TTPLPFC's voltage loop as well as housekeeping functions such

as doing a running average on the currents and voltage signals to remove noise. Even so much as running the slew rate function for commanded references.

```
#define CLLLC_ISR3_TIMEBASE CLLLC_TASKC_CPUTIMER_BASE
#define CLLLC_ISR3_PERIPHERAL_TRIG_BASE ADCC_BASE
#define CLLLC_ISR3_TRIG INT_ADCC2
```

```
#define CLLLC_ISR3_PIE_GROUP INTERRUPT_ACK_GROUP10
```

With this, the interrupts can be nested easily. Figure 5-6 shows the nesting of the three interrupts occurring. The image is taken for the system in open loop and when a period change is initiated through the watch window, hence only one time ISR1 trigger is observed. For a closed-loop system, the period will only have minor changes from one control ISR cycle to the other and hence the ISR1 will be triggered repeatedly.

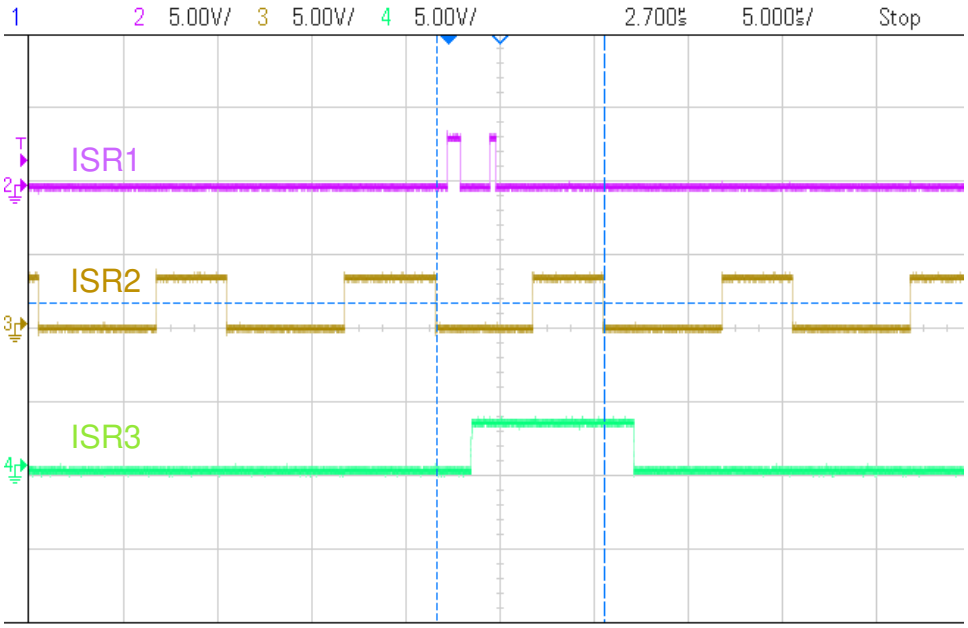



**Figure 5-6. Three-Level Nested ISRs**

Additionally, CPU timers are used to trigger slow background tasks (these are not interrupt-driven but polled).

"A" tasks are triggered at TASKA_FREQ, which is 100 Hz. The SFRA GUI must be called at this rate. One task, A1, is executed at this rate.

"B" tasks are triggered at TASKB_FREQ, which is 10 Hz. These are used for some basic LED toggles and state machine items that are not timing-critical. Three tasks—B1, B2, B3—are serviced by this; hence, each has an execution rate of 3.33 Hz.

```
#define TASKA_FREQ 100
#define TASKB_FREQ 10
```

The software of this reference design is organized into labs separated by solution, each with incremental builds (INCR_BUILD). These tests simplify the system bring up and design.

### CLLLC Labs

**Lab 1:** Primary to Secondary Power Flow, Open Loop Check PWM driver, no high power applied to the board. See Section 5.2.2.1

**Lab 2:** Primary to Secondary Power Flow, Open Loop Check PWM driver and ADC with protection, resistive load connected on secondary. See Section 5.2.2.2.

**Lab 3:** Primary to Secondary Power Flow, Closed Voltage Loop Check, with resistive load connected on secondary. See Section 5.2.2.3.

**Lab 4:** Primary to Secondary Power Flow, Closed Current Loop Check, with resistive load connected on secondary. See Section 5.2.2.4.

**Lab 5:** Primary to Secondary Power Flow, Closed Current Loop Check, with resistive load connected on secondary in parallel to a voltage source to emulate a battery connection on secondary side. See Section 5.2.2.5.

These defines are in the "*settings.h*" file, and can be changed directly within that file.

## 5.2 Testing and Results

### 5.2.1 Test Setup (Initial)

Hardware test setup is shown in Figure 5-7 to begin testing this design. Detailed hardware setup is described in Section 5.1.1.

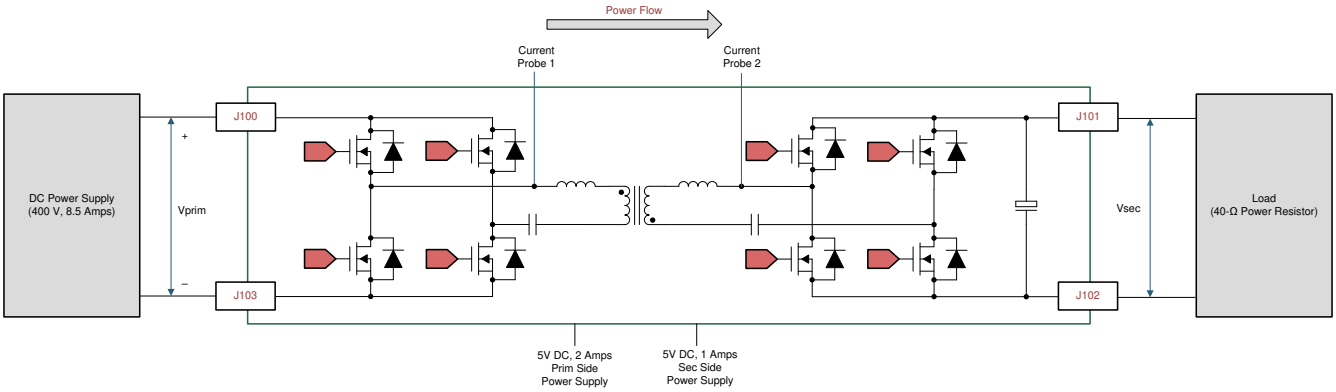



**Figure 5-7. Hardware Setup to Run Software**

### 5.2.2 CLLLC Test Procedure

#### *5.2.2.1 Lab 1. Primary to Secondary Power Flow, Open Loop Check PWM Driver*

This lab option is primarily provided as a focused test just for the PWM from a software perspective so that it can be ran independent of the hardware connections of the reference design. With this lab the user can run the code on a C2000 control card or launchpad just to observe the PWM waveforms.

Steps used to load and run are similar to Section 5.2.2.2 .

This lab can be easily skipped and the user can go to Lab 2 directly if no changes to the PWM driver are anticipated. Hence this lab procedure is not documented as it is primarily for PWM driver development and debug purpose.

#### *5.2.2.2 Lab 2. Primary to Secondary Power Flow, Open Loop CheckPWM Driver and ADC with Protection, Resistive Load Connected on Secondary*

In this build, the board is excited in open-loop fashion with a specified frequency that can be changed through the watch window. The frequency is controlled with the CLLLC_pwmPeriodRef_pu variable.

This build verifies the sensing of feedback values from the power stage and also operation of the PWM gate driver, and ensures there are no hardware issues. Additionally, calibration of input and output voltage sensing can be performed in this build. Figure 5-8 shows the software structure for this build.

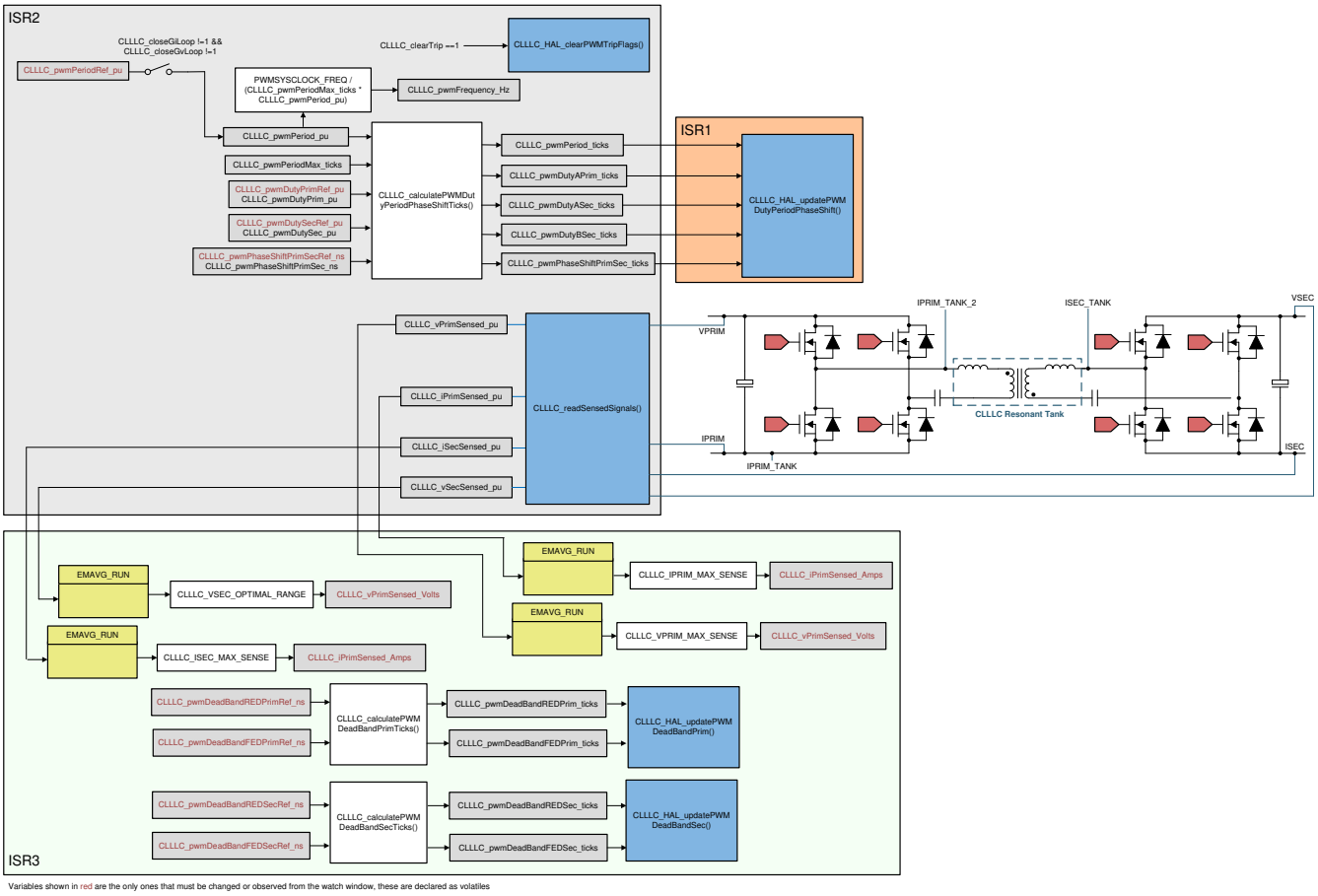



**Figure 5-8. Lab 1 and 2 Software Diagram**

#### 5.2.2.2.1 Setting Software Options for Lab 2

1. To get started, open the CCS project, as outlined in Section 5.1.2.1.
2. The following defines are set in the *settings.h* file for this build.

```
#if CLLLC_LAB == 2
#define CLLLC_CONTROL_RUNNING_ON CLA_CORE
#define CLLLC_POWER_FLOW CLLLC_POWER_FLOW_PRIM_SEC
#define CLLLC_INCR_BUILD CLLLC_OPEN_LOOP_BUILD
#define CLLLC_TEST_SETUP CLLLC_TEST_SETUP_RES_LOAD
#define CLLLC_PROTECTION CLLLC_PROTECTION_ENABLED

#if CLLLC_SFRA_ALLOWED == 1
#define CLLLC_SFRA_TYPE  CLLLC_SFRA_VOLTAGE
#else
#define CLLLC_SFRA_TYPE  CLLLC_SFRA_DISABLED
#endif

#define CLLLC_SFRA_AMPLITUDE (float32_t)CLLLC_SFRA_INJECTION_AMPLITUDE_LEVEL2
#endif
```

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

1. Right-click on the project name and click *Rebuild Project*.
2. The project will build successfully.
3. In the Project Explorer, make sure the correct target configuration file is set as Active under targetConfigs (see Figure 5-5).
4. Then, click *Run* → *Debug* to launch a debugging session. In case of dual-CPU devices, a window might appear for the user to select the CPU on which the debug is to be performed. In this case, select *CPU1*.
5. The project will then load on the device and the CCS debug view will become active. The code will halt 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_lab2and7.js* script file located inside the project folder. This will populate the watch window with the appropriate variables needed to debug the system.
7. Click the *Continuous Refresh* button  on the watch window to enable continuous update of values from the controller. The watch window appears as shown in Figure 5-9.

Variables | Expressions | Registers | Breakpoints

| Expression | Type | Value | Address |
|---|---|---|---|
| CLLLC_buildLevel.CLLLC_BuildLevel_Enum | enum <unnamed> | openLoopCheck | 0x0000805E@Data |
| CLLLC_pwmSwState.CLLLC_PwmSwState_Enum | enum <unnamed> | pwmSwState_primPWMActive_SecPWMActiveSynchRectification | 0x00008058@Data |
| CLLLC_powerFlowState.CLLLC_PowerFlowState_Enum | enum <unnamed> | powerFlow_BatteryCharging | 0x00008060@Data |
| CLLLC_tripFlag.CLLLC_TripFlag_Enum | enum <unnamed> | noTrip | 0x0000805C@Data |
| CLLLC_clearTrip | long | 0 | 0x00008050@Data |
| CLLLC_vPrimSensed_Volts | float | 0.149676159 | 0x00008076@Data |
| CLLLC_iPrimSensed_Amps | float | 0.051383622 | 0x0000804C@Data |
| CLLLC_vSecSensed_Volts | float | 0.450056016 | 0x00008020@Data |
| CLLLC_iSecSensed_Amps | float | 0.0225450285 | 0x0000801C@Data |
| CLLLC_pwmPeriodRef_pu | float | 0.599041522 | 0x00008040@Data |
| CLLLC_pwmPhaseShiftPrimSecRef_ns | float | 81.0 | 0x00008032@Data |
| CLLLC_pwmDeadBandREDPrimRef_ns | float | 102.300003 | 0x0000807E@Data |
| CLLLC_pwmDeadBandFEDPrimRef_ns | float | 102.599998 | 0x00008086@Data |
| CLLLC_pwmFrequency_Hz | float | 500800.0 | 0x0000803C@Data |
| EPwm1Regs.TBPRD | unsigned int | 99 | 0x00004063@Data |
| EPwm2Regs.TBPRD | unsigned int | 99 | 0x00004163@Data |
| EPwm3Regs.TBPRD | unsigned int | 99 | 0x00004263@Data |
| EPwm4Regs.TBPRD | unsigned int | 99 | 0x00004363@Data |
| EPwm1Regs.TZFLG | union TZFLG_REG | {all=4,bit={INT=0,CBC=0,OST=1,DCAEVT1=0,DCAEVT2=0...}} | 0x00004093@Data |
| *Add new expression* | | | |

**Figure 5-9. Lab 2 Expression Window**

#### 5.2.2.2.3 Using Real-time Emulation

Real-time emulation is a special emulation feature that allows windows within Code Composer Studio to be updated *while the MCU is running*. This 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 your mouse on the buttons on the horizontal toolbar and clicking the 

button.
2. A message box *might* appear. If so, select *YES* to enable debug events. This will set bit 1 (DGBM bit) of status register 1 (ST1) to a *0*. 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.

#### 5.2.2.2.4 Running the Code

1. Run the project by clicking  .
2. Now, clear the trip by writing *1* to the CLLLC_clearTrip variable.
3. In the watch view, check if the CLLLC_vPrimSensed_Volts, CLLLC_iPrimSensed_Amps, CLLLC_vSecSensed_Volts, and CLLLC_iSecSensed_Amps variables are updating periodically. (Note: As no power is applied right now, these will be close to zero.)
4. Now, slowly increase the input VPRIM DC voltage from 0 V to 400 V. Make sure CLLLC_vPrimSensed_Volts displays the correct values.
5. By default, the CLLLC_pwmPeriodRef_pu variable is set to 0.599, as shown in Figure 5-10, which is 500.8 kHz. This is close to the series resonant frequency of the converter; however, due to variation in the components on the actual hardware, it can be lower or higher than the series resonant frequency. For example, in Figure 5-11, we see frequency slightly lower than series resonant frequency.
6. The VSEC variable will show a voltage of close to 300 V per the tank gain designed. Verify that CLLLC_vSecSensed_Volts shows the correct voltage. This verifies the voltage sensing on the board.

Variables | Expressions | Registers | Breakpoints

| Expression | Type | Value | Address |
|---|---|---|---|
| CLLLC_buildLevel.CLLLC_BuildLevel_Enum | enum <unnamed> | openLoopCheck | 0x0000805E@Data |
| CLLLC_pwmSwState.CLLLC_PwmSwState_Enum | enum <unnamed> | pwmSwState_primPWMActive_SecPWMActiveSynchRectification | 0x00008058@Data |
| CLLLC_powerFlowState.CLLLC_PowerFlowState_Enum | enum <unnamed> | powerFlow_BatteryCharging | 0x00008060@Data |
| CLLLC_tripFlag.CLLLC_TripFlag_Enum | enum <unnamed> | noTrip | 0x0000805C@Data |
| CLLLC_clearTrip | long | 0 | 0x00008050@Data |
| CLLLC_vPrimSensed_Volts | float | 403.606567 | 0x00008076@Data |
| CLLLC_iPrimSensed_Amps | float | 4.82742071 | 0x0000804C@Data |
| CLLLC_vSecSensed_Volts | float | 296.258301 | 0x00008020@Data |
| CLLLC_iSecSensed_Amps | float | 6.27206373 | 0x0000801C@Data |
| CLLLC_pwmPeriodRef_pu | float | 0.599041522 | 0x00008040@Data |
| CLLLC_pwmPhaseShiftPrimSecRef_ns | float | 81.0 | 0x00008032@Data |
| CLLLC_pwmDeadBandREDPrimRef_ns | float | 102.300003 | 0x0000807E@Data |
| CLLLC_pwmDeadBandFEDPrimRef_ns | float | 102.599998 | 0x00008086@Data |
| CLLLC_pwmFrequency_Hz | float | 500800.0 | 0x0000803C@Data |
| EPwm1Regs.TBPRD | unsigned int | 99 | 0x00004063@Data |
| EPwm2Regs.TBPRD | unsigned int | 99 | 0x00004163@Data |
| EPwm3Regs.TBPRD | unsigned int | 99 | 0x00004263@Data |
| EPwm4Regs.TBPRD | unsigned int | 99 | 0x00004363@Data |
| EPwm1Regs.TZFLG | union TZFLG_REG | {all=0,bit={INT=0,CBC=0,OST=0,DCAEVT1=0,DCAEVT2=0...}} | 0x00004093@Data |
| *Add new expression* | | | |

**Figure 5-10. Lab 2 Expression Window, at Resonance**

1. With the load specified in the test conditions, the current from the PRIM and SEC side will be close to 4.8 A for CLLLC_iPrimSensed_Amps, and 6.8 A for CLLLC_iSecSensed_Amps.

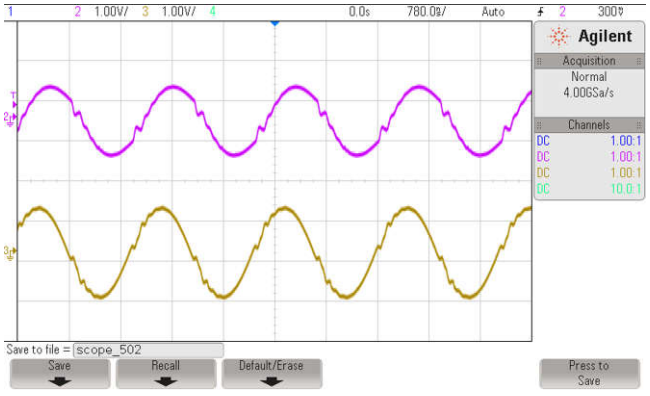



**Figure 5-11. Lab 2, Primary (ch2) and Secondary (ch3) Currents at Resonance**

1. Next, to see operation under different frequencies (that is, above resonance, below resonance), change the CLLLC_pwmPeriodRef_pu variable to be 0.47 which will correspond to a frequency of 639 kHz. The waveform under this condition is shown in Figure 5-12.

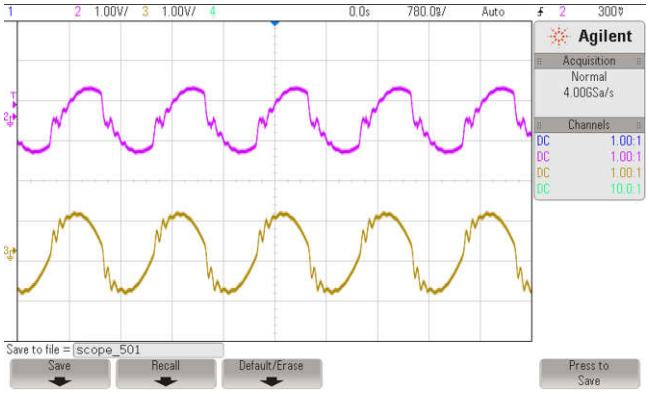



**Figure 5-12. Lab 2, Primary (ch2) and Secondary (ch3) Currents Above Series Resonance Frequency**

1. Next, test behavior with lower than series resonant frequency by entering 0.8 as the CLLLC_pwmPeriodRef_pu, which will make generate frequency of 374 kHz. In this case, the primary current will become discontinuous and the secondary side duty cycle will modulate to achieve diode emulation shown in Figure 5-13.
2. This verifies at a basic level the PWM driver and connection of hardware.

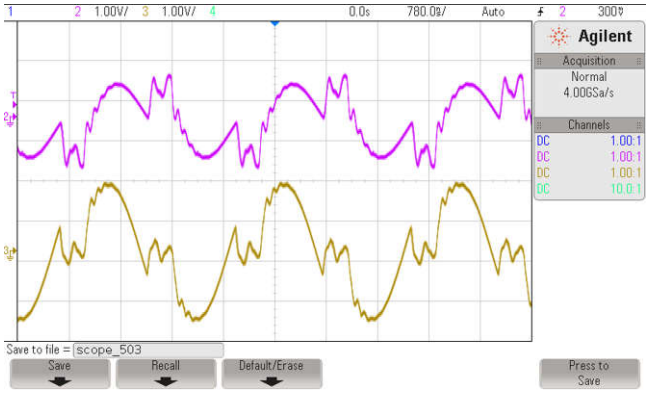



**Figure 5-13. Lab 2, Primary (ch2) and Secondary (ch3) Currents Below Series Resonance Frequency**

#### 5.2.2.2.5 Measure SFRA Plant for Voltage Loop

1. The SFRA is integrated in the software of this build to measure the plant response which can then be used to design a compensator. To run the SFRA, keep the project running, and navigate to *<Install directory >\C2000Ware_DigitalPower_SDK_<version>\libraries\sfra\gui\SFRA_GUI.exe*
2. 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, deselect the *boot-on-connect* option and select an appropriate COM port. Click *OK*. Return to the SFRA GUI and click *Connect*.
3. 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 of the control card, which indicates UART activity. Once complete, a graph with the measurement will appear, as shown in Figure 5-14. (Note that the open-loop measurement is not valid in the lab as the loop is not closed. The user must only refer to the plant measurement.)

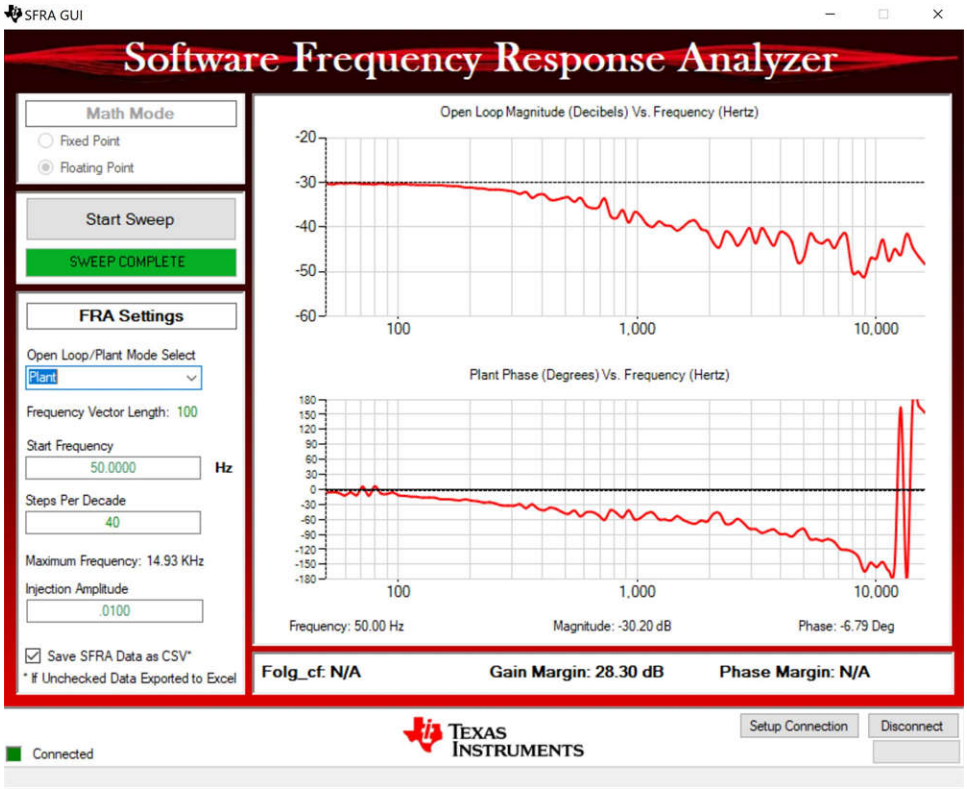



**Figure 5-14. SFRA Open Loop Plot for the Closed Voltage Loop (Vprim 400 V, Vsec 300 V, Power 1.972 kW, Fsw 500 kHz)**

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. SFRA can be run at different frequency set points to cover the range of operation of the system. A compensator will be designed using these measured plots in the next lab; therefore, remember this time stamp, or rename the *SFRA.csv* file to a convenient name that is easy to identify.

Repeat the analysis at different frequency points, the plant gain will be different at different frequency points, see Figure 5-15 for gain measured at 333kHz and see Figure 5-16 for gain measured at 680kHz. Hence a compensator needs to be chosen that will be stable across the frequency range of the converter. All the runs will be saved in CSV file and can then be imported into compensation designer to check stability across the range of operation.

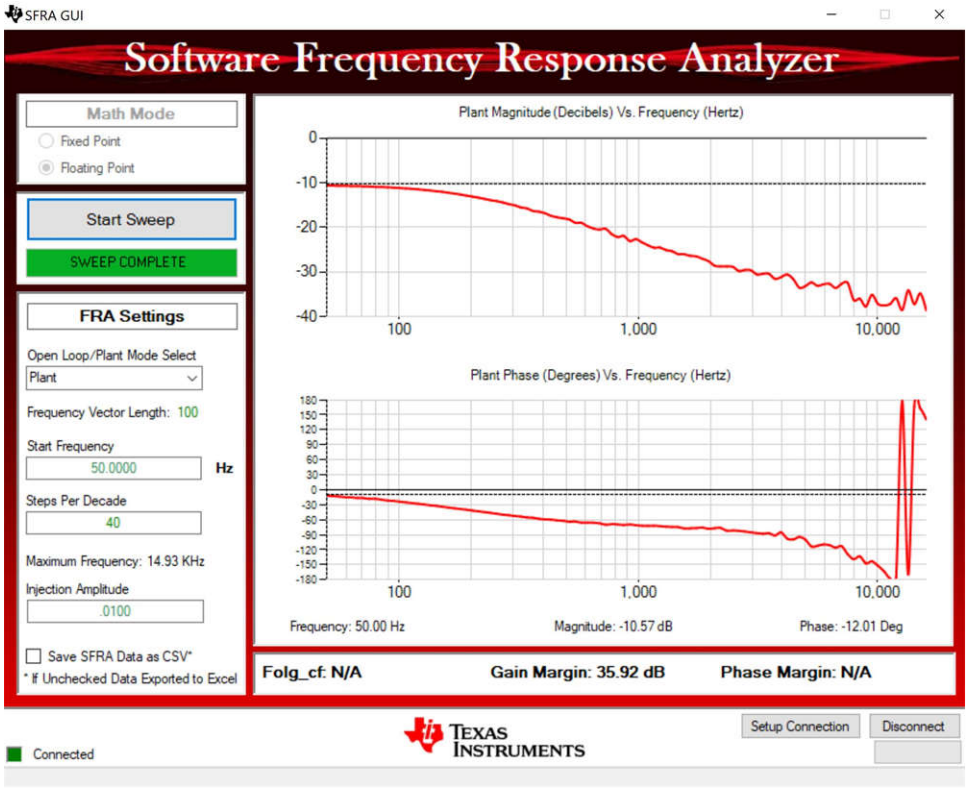



**Figure 5-15. SFRA Open Loop Plot for the Closed Voltage Loop (Vprim 400 V, Vsec 320 V, Power 2.174 kW, Fsw 333 kHz)**

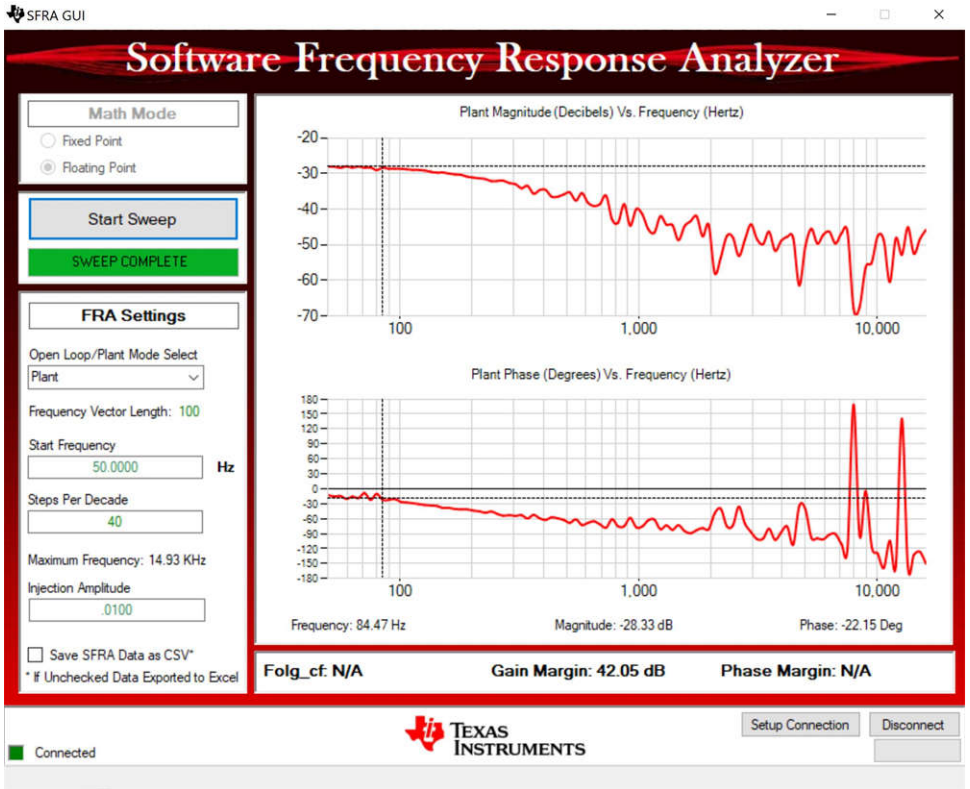



**Figure 5-16. SFRA Open Loop Plot for the Closed Voltage Loop (Vprim 400 V, Vsec 293 V, Power 1.828 kW, Fsw 680 kHz)**

#### 5.2.2.2.6 Verify Active Synchronous Rectification

1. Optionally, to verify active synchronous rectification, the user can also probe the PWM signals and see the change in duty cycle. To connect the probe for it, the user must first stop the power stage as outlined below and de-energize all circuits under test.

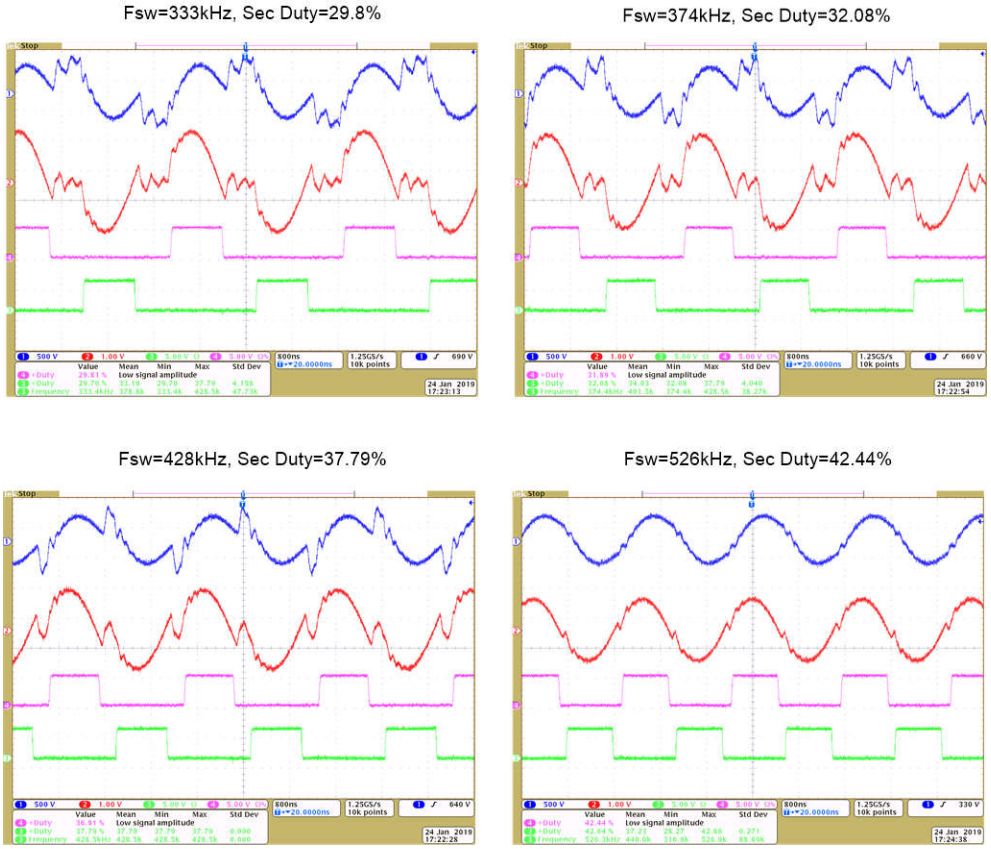



**Figure 5-17. Active Synchronous Rectification Check, ch1 → IPRIM_TANK, ch2 → ISEC_TANK, ch3/ch4 → SEC_LEG1_PWMH/L**

2. Once finished, reduce the input voltage, VPRIM, to zero. Watch for the voltages in the watch window to reduce down to zero.
3. 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  .
4. Close the CCS debug session by clicking *Terminate Debug Session*  (Target → Terminate all).

#### 5.2.2.2.7 Measure SFRA Plant for Current Loop

1. Now, return to the SYSCFG page, and select current in the SFRA options to measure the current loop plant.
2. Rebuild the project, and reload the project. Repeat 2 (in Section 5.2.2.2.1) to Section 5.2.2.2.7 (in Section 5.2.2.2.5). This time, the SFRA sweep will measure the plant for the current loop. Save this CSV file for use later in the Lab 3. The user can measure this at multiple points to ensure all operating conditions are covered, Figure 5-18 shows the measurement of the current loop plant at 500kHz.

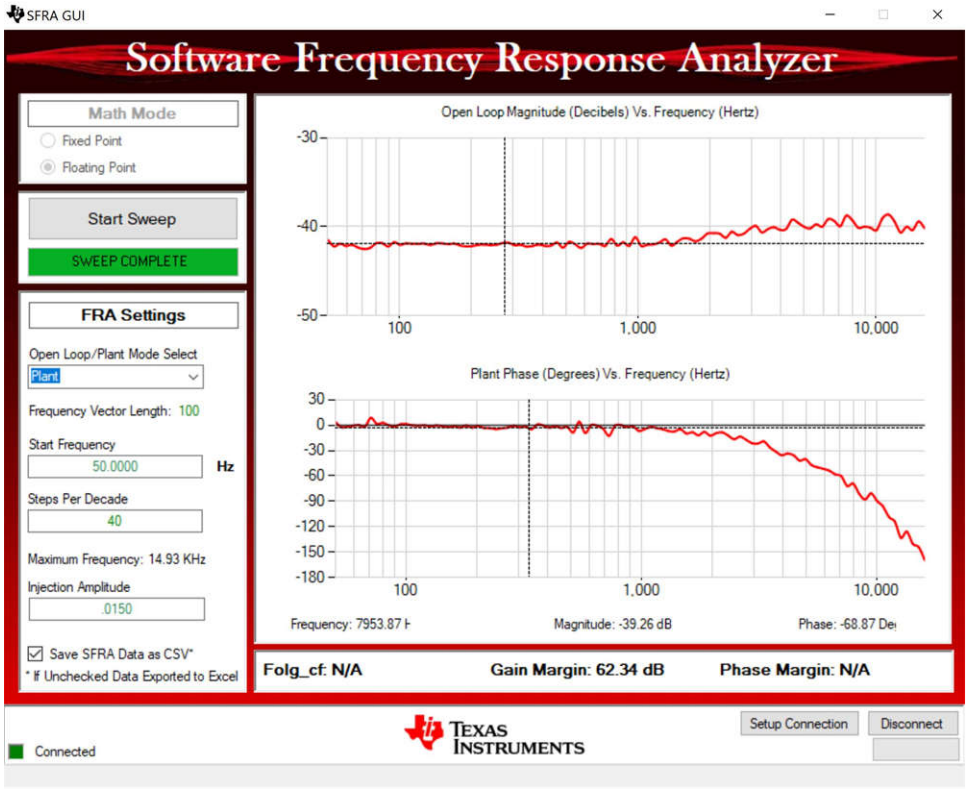



**Figure 5-18. SFRA Plant Measurement for the Current Loop at Vprim 400 V, Vsec 295 V, Switching Frequency at 500 kHz and 1887 W**

3. 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 ISR1, ISR2, and ISR3
   c. PWM driver and switching
   d. Plant measurement for current and voltage loop

   If any issue is observed, a careful inspection of the hardware may be required to eliminate any build issues.
4. The controller can now be halted and the debug connection terminated.
5. 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  .
6. Close the CCS debug session by clicking on Terminate Debug Session  (Target → Terminate all).

### *5.2.2.3 Lab 3. Primary to Secondary Power Flow, Closed Voltage Loop Check, With Resistive Load Connected on Secondary*

In this lab, the voltage loop, $G_v$, is closed with a resistive load at the output. Figure 5-19 shows the complete software diagram for this build. Hardware is assumed to be set up as shown in Figure 5-7.

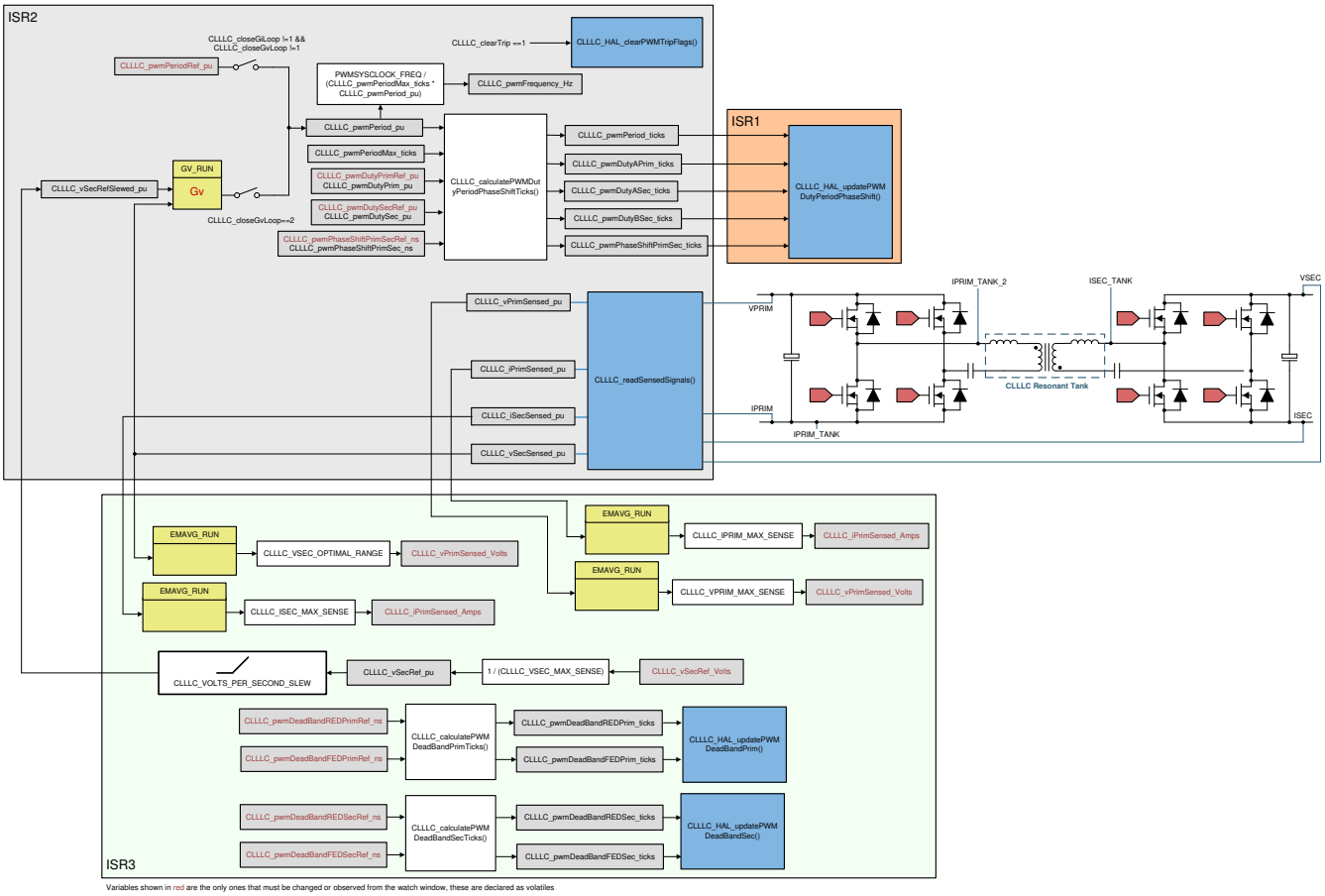



**Figure 5-19. Software Diagram: Closed Voltage Loop Primary to Secondary Power Flow**

#### 5.2.2.3.1 Setting Software Options for Lab 3

1. To run this lab, make sure the hardware is set up as outlined in the previous sections, Figure 5-7. Do not supply any high-voltage power to the board yet.
2. Open "<install Directory>\C2000Ware_DigitalPower_SDK_<version>\libraries\sfra\gui\CompDesigner.exe"
3. The compensation designer will then launch and prompt 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. It is good to keep more margins during this iteration of the design to ensure that when the loop is closed, the system is stable. Plant data from different runs of the SFRA can be checked to get a stable system under all conditions, for example two runs at 500kHz and 300kHz with the designed compensator are shown to be stable in Figure 5-20 and Figure 5-21.

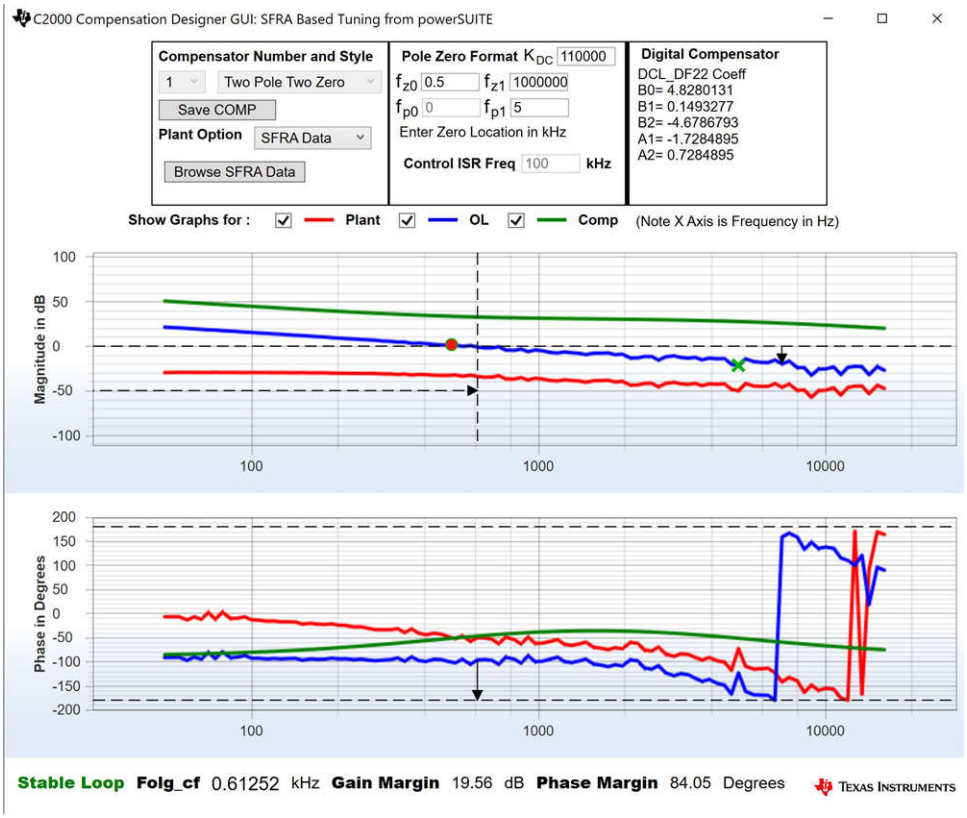



**Figure 5-20. Compensator Design With SFRA Based Plant Measurement for the Voltage Loop When Resistive Load is Connected at the Output, With Measurement Data at 500kHz**

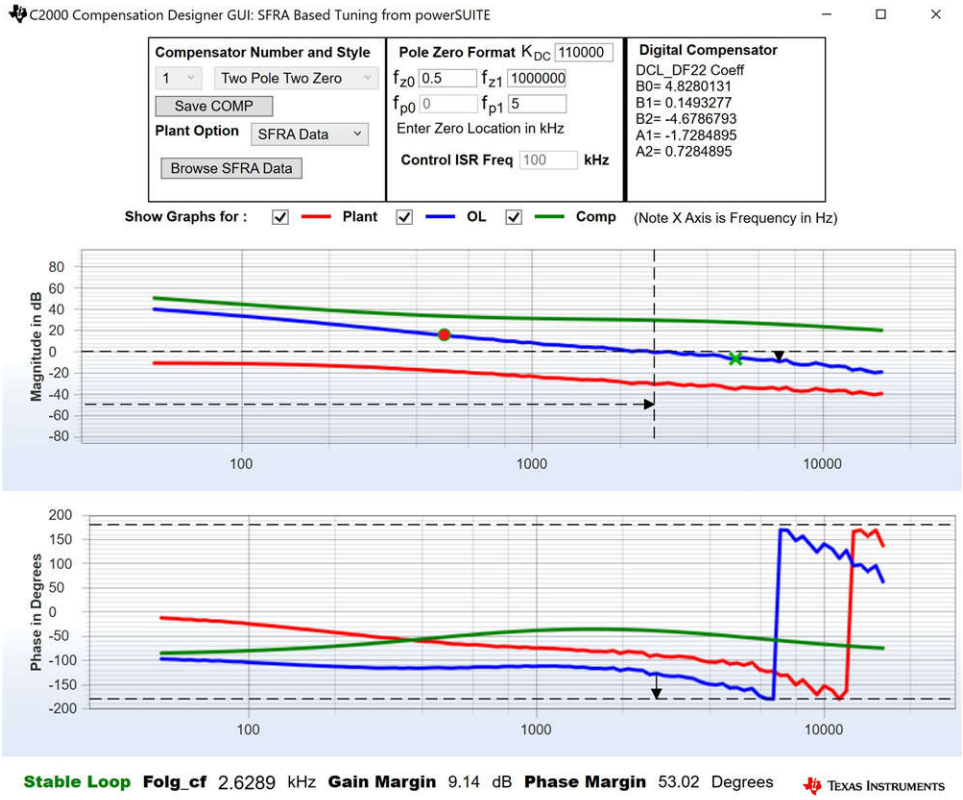



**Figure 5-21. Compensator Design With SFRA Based Plant Measurement for the Voltage Loop When Resistive Load is Connected at the Output, With Measurement Data at 333kHz**

**Note**

The tuning is carried out in DF22 fashion; however, we run the DF13 in the software. This is done because soft-starting the DF13 is easier, whereas not possible with the DF22 structure. The coefficients in both cases remain the same. At the writing of this document, the DF12 structure is not available in DCL.

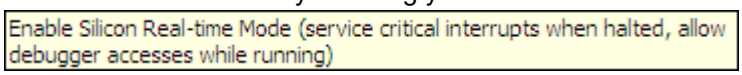
4. Once satisfied with the compensator design, The compensator values can be updated in the CLLLC_settings.h file.
5. Close the compensation designer
6. The following defines are set in the *settings.h* file for this build..

```
#if CLLLC_LAB == 3 #define CLLLC_CONTROL_RUNNING_ON CLA_CORE #define CLLLC_POWER_FLOW
CLLLC_POWER_FLOW_PRIM_SEC #define CLLLC_INCR_BUILD CLLLC_CLOSED_LOOP_BUILD #define
CLLLC_CONTROL_MODE CLLLC_VOLTAGE_MODE #define CLLLC_TEST_SETUP CLLLC_TEST_SETUP_RES_LOAD
#define CLLLC_PROTECTION CLLLC_PROTECTION_ENABLED #if CLLLC_SFRA_ALLOWED == 1 #define
CLLLC_SFRA_TYPE CLLLC_SFRA_VOLTAGE #else #define CLLLC_SFRA_TYPE CLLLC_SFRA_DISABLED #endif
#define CLLLC_SFRA_AMPLITUDE (float32_t)CLLLC_SFRA_INJECTION_AMPLITUDE_LEVEL1 #endif
```

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

1. Now, right-click on the project name and click *Rebuild Project*.
2. The project will build successfully.
3. Click *Run* → *Debug* to launch a debugging session. In case of dual-CPU devices, a window may appear for the user to select the CPU on which the debug is to be performed. In this case, select CPU1.
4. The project will then load on the device and the CCS debug view will become active. The code will halt at the start of the main routine.
5. 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 *open* to browse to the *setupdebugenv_lab3.js* script file located inside the project folder. This will populate the watch window with the appropriate variables needed to debug the system.

6. Click the Continuous Refresh button  on the watch window to enable continuous update of values from the controller.

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


button.

#### 5.2.2.3.3 Running the Code

1. Run the project by clicking 
2. Now, clear the trip by writing *1* to the CLLLC_clearTrip variable. The converter will operate in open loop as the CLLLC_closeGvLoop variable is not yet set to *0*. As there is no soft start implemented in the firmware, first soft-start the voltages on the primary and secondary sides manually.
3. In the watch view, check if the CLLLC_vPrimSensed_Volts, CLLLC_iPrimSensed_Amps, CLLLC_vSecSensed_Volts, and CLLLC_iSecSensed_Amps variables are updating periodically. (Note: As no power is applied right now, these will be close to zero.)
4. Now, slowly increase the input PRIM DC voltage from 0 V to 400 V to soft-start the converter. Make sure CLLLC_vPrimSensed_Volts displays the correct values for VPRIM (that is, close to 400 V).
5. By default, the CLLLC_pwmPeriodRef_pu variable is set to 0.599, which is 500.8 kHz. This is close to the series resonant frequency of the converter; however, due to variation in the components on the actual hardware, it can be lower or higher than the series resonant frequency.
6. For the 400-V primary input, with turns ratio being 1.33, the CLLLC_vSecSensed_Volts variable will be close to 300 V. Set the CLLLC_vSecRef_Volts variable to be 300 V.
7. Now, set the CLLLC_closeGvLoop variable to *1*. This will close the voltage loop and the controller will now try to regulate the voltage.
8. Test the closed-loop operation by varying CLLC_vSecRef_Volts from 295 V to 320 V. The user will observe that the CLLLC_vSecSensed_Volts will track this command reference. The converter will operate below series resonant, at resonance, and above resonance. Now, change the voltage back to 300 V to run the SFRA.

#### 5.2.2.3.4 Measure SFRA for Closed Voltage Loop

1. The SFRA is integrated in the software of this build to verify that the designed compensator provides enough gain and phase margin by measuring on hardware. To run the SFRA, keep the project running, and navigate to *<Install directory >\C2000Ware_DigitalPower_SDK_<version>\libraries\sfra\gui\SFRA_GUI.exe*
2. 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, deselect the boot-on-connect option and select an appropriate COM port. Click Ok. Return to the SFRA GUI and click Connect.
3. 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 of the control card, which indicates UART activity. Once complete, a graph with the open loop plot will appear, as shown in Figure 5-22.

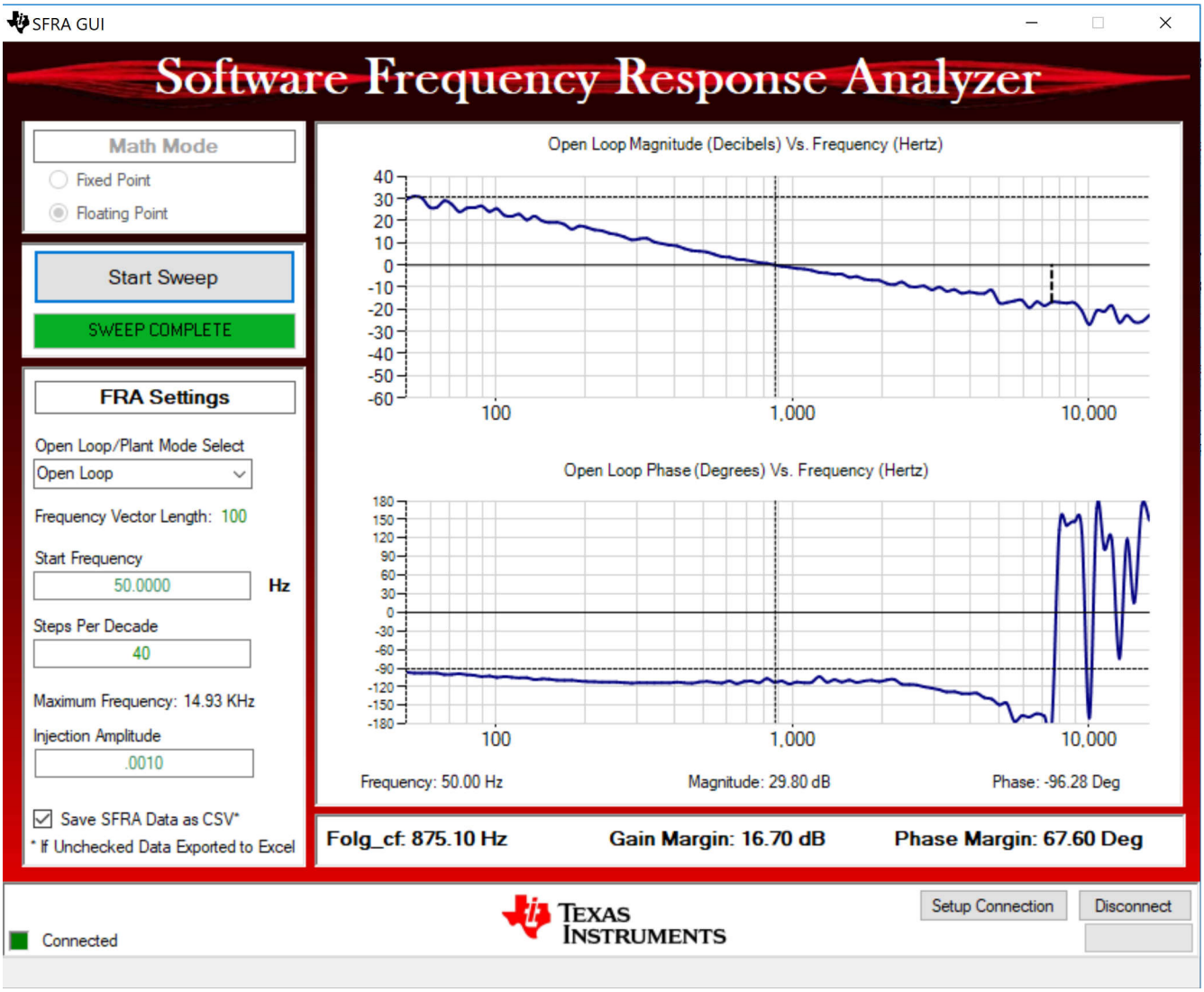



**Figure 5-22. SFRA Open Loop Plot for the Closed Voltage Loop (Vprim 400 V, Vsec 300 V, Power 1.972 kW, with Resistive Load at the Output)**

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.

The data matches closely to the designed compensator, but it is reasonable to expect deviations because the measurement in open loop is susceptible to error, due to small signal injection, which can drift the DC point of the converter.

Test the SFRA at different voltages to verify that the system is stable across the operable range.

4. This verifies the voltage loop design.
5. To bring the system to a safe stop, bring the input VPRIM voltage down to zero. Observe the voltages and currents on the watch window go down to zero.
6. 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  .
7. Close the CCS debug session by clicking on Terminate Debug Session  (Target → Terminate all).

### 5.2.2.4 Lab 4. Primary to Secondary Power Flow, Closed Current Loop Check, With Resistive Load Connected on Secondary

In this lab, the output current control loop is closed. Figure 5-23 shows the complete software diagram for this build. Hardware is assumed to be set up as shown in Figure 5-7.

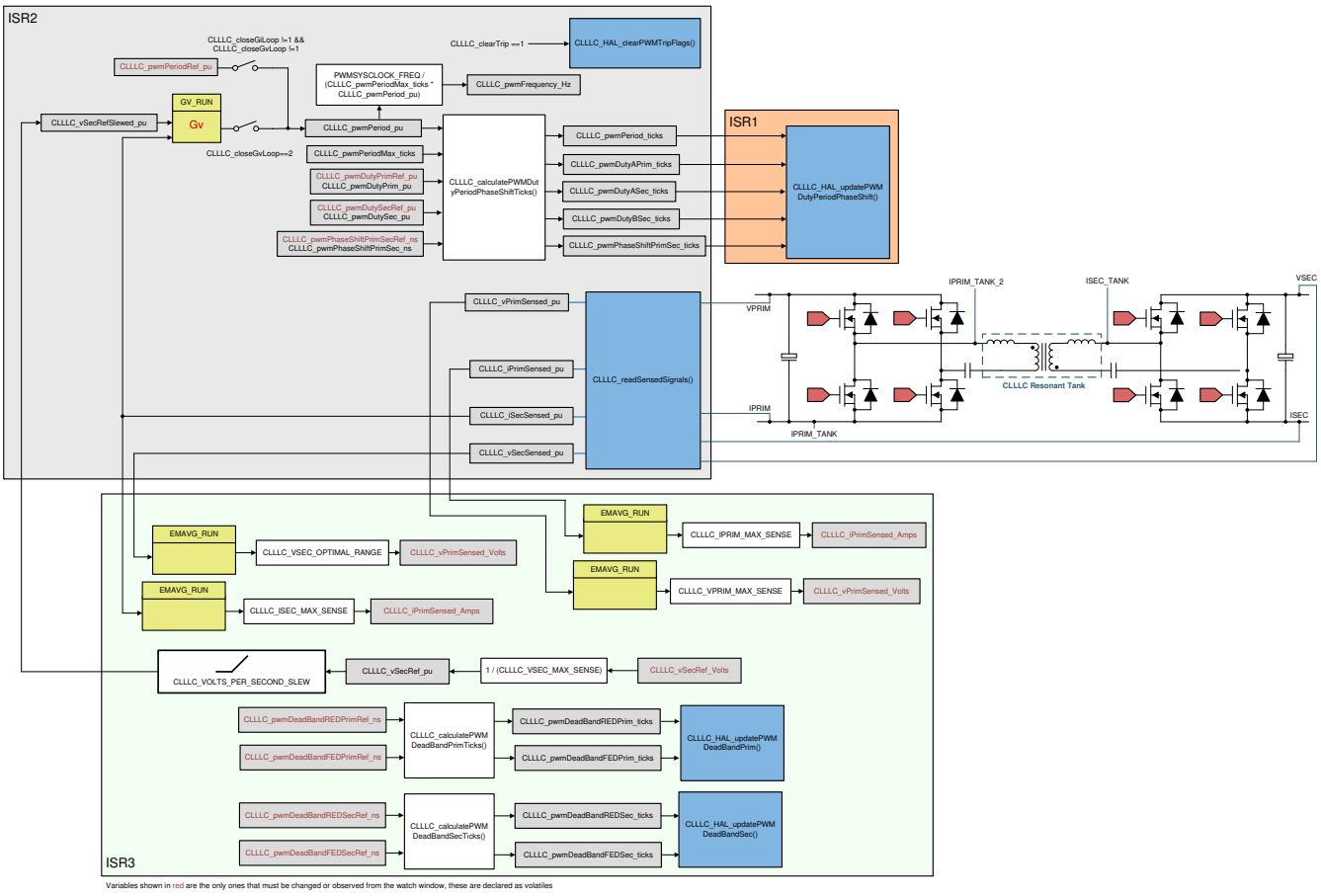



**Figure 5-23. Lab 4 Control Software Diagram: Closed Current Loop**

#### 5.2.2.4.1 Setting Software Options for Lab 4

1. Open *<install Directory>\C2000Ware_DigitalPower_SDK_<version>\libraries\sfra\gui\CompDesigner.exe*
2. The compensation designer will then launch and prompt the user to select a valid SFRA data file. Import the SFRA data from the run in Lab 1 for the current loop, into the compensation designer to design a two-pole, two-zero compensator. It is good to keep more margins during this iteration of the design to ensure that when the loop is closed, the system is stable. Plant data from different runs of the SFRA can be checked to get a stable system under all conditions.

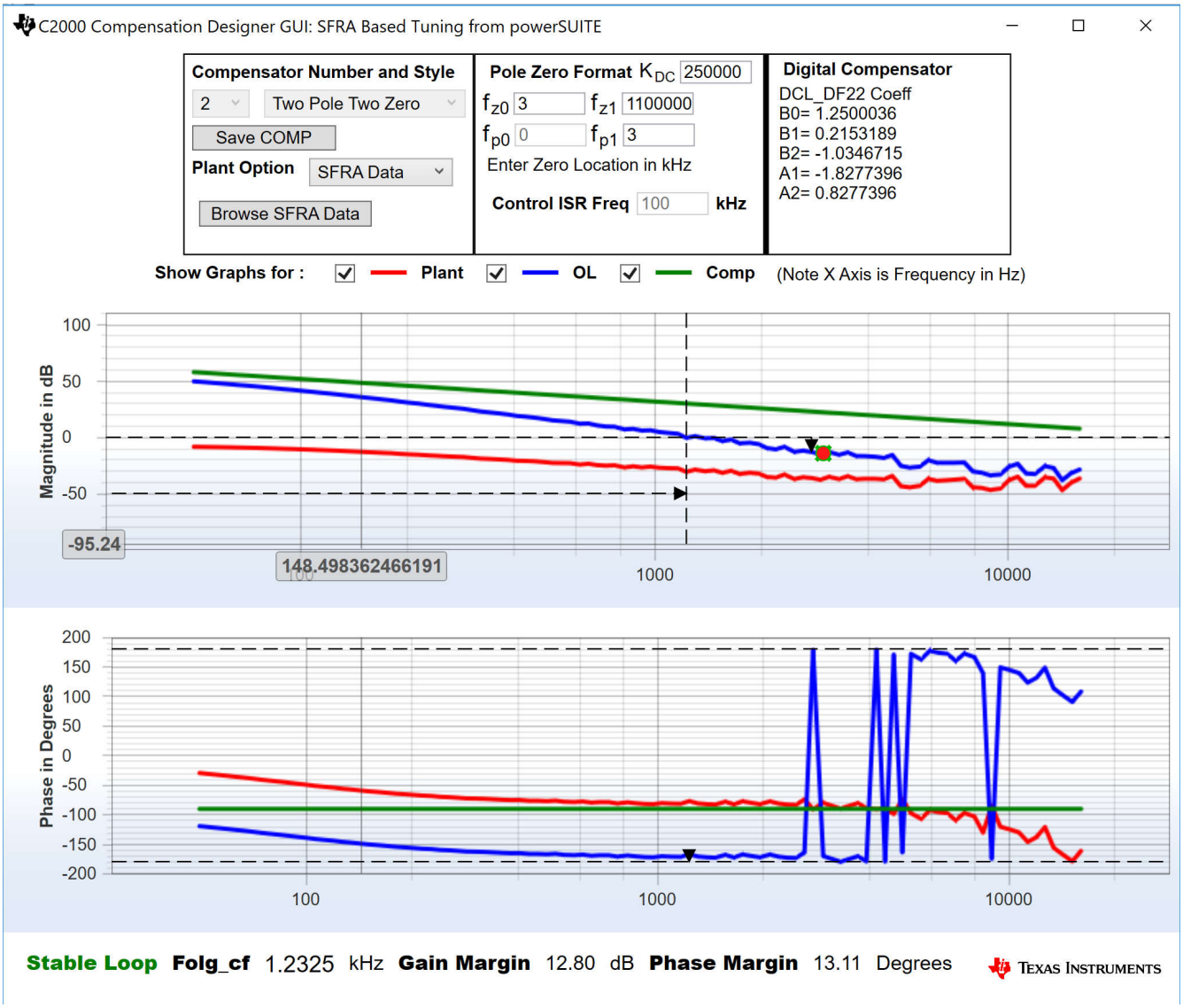



**Figure 5-24. Compensator Design With SFRA Based Plant Measurement for the Current Loop, Lab 4**

**Note**

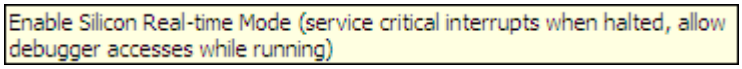
The tuning is carried out in DF22 fashion; however, we run the DF13 in the software. This is done because soft-starting the DF13 is easier, whereas not possible with the DF22 structure. The coefficients in both cases remain the same. At the writing of this document, the DF12 structure is not available in DCL.

3. Once satisfied with the compensator design, The compensator values can be updated in the CLLLC_settings.h file.
4. Close the compensation designer

5. The following defines are set in the *settings.h* file for this build.

```
#if CLLLC_LAB == 4 #define CLLLC_CONTROL_RUNNING_ON CLA_CORE #define CLLLC_POWER_FLOW
CLLLC_POWER_FLOW_PRIM_SEC #define CLLLC_INCR_BUILD CLLLC_CLOSED_LOOP_BUILD #define
CLLLC_CONTROL_MODE CLLLC_CURRENT_MODE #define CLLLC_TEST_SETUP CLLLC_TEST_SETUP_RES_LOAD
#define CLLLC_PROTECTION CLLLC_PROTECTION_ENABLED #if CLLLC_SFRA_ALLOWED == 1 #define
CLLLC_SFRA_TYPE CLLLC_SFRA_CURRENT #else #define CLLLC_SFRA_TYPE CLLLC_SFRA_DISABLED #endif
#define CLLLC_SFRA_AMPLITUDE (float32_t)CLLLC_SFRA_INJECTION_AMPLITUDE_LEVEL1 #endif
```

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

1. Now, right-click on the project name and click *Rebuild Project*.
2. The project will build successfully.
3. Click *Run* → *Debug* to launch a debugging session. In case of dual-CPU devices, a window may appear for the user to select the CPU on which the debug is to be performed. In this case, select CPU1.
4. The project will then load on the device and the CCS debug view will become active. The code will halt at the start of the main routine.
5. 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* to browse to the *setupdebugenv_build4c.js* script file located inside the project folder. This will populate the watch window with the appropriate variables needed to debug the system.
6. Click on the *Continuous Refresh* button  on the watch window to enable continuous update of values from the controller.
7. Enable real-time mode by hovering your mouse on the buttons on the horizontal toolbar and clicking the  

button.

#### 5.2.2.4.3 Running the Code

1. Run the project by clicking 
2. Clear the trip by writing *1* to the CLLLC_clearTrip variable. The converter will operate in open loop as the CLLLC_closeGvLoop variable is not yet set to “0”. As there is no soft start implemented in the firmware, first soft-start the voltages on the primary and secondary sides manually.
3. In the watch view, check if the CLLLC_vPrimSensed_Volts, CLLLC_iPrimSensed_Amps, CLLLC_vSecSensed_Volts, and CLLLC_iSecSensed_Amps variables are updating periodically. (Note: As no power is applied right now, these will be close to zero.)
4. Now, slowly increase the input PRIM DC voltage from 0 V to 400 V to soft-start the converter. Make sure CLLLC_vPrimSensed_Volts displays the correct values for VPRIM (that is, close to 400 V).
5. By default, the CLLLC_pwmPeriodRef_pu variable is set to 0.6, which is 500.8 kHz. This is close to the series resonant frequency of the converter; however, due to variation in the components on the actual hardware, it can be lower or higher than the series resonant frequency.
6. For the 400-V primary input, with turns ratio being 1.33, the CLLLC_vSecSensed_Volts variable will be close to 300 V. Also, for the load specified in the test conditions, the load will be close to 6.5 A. Set the CLLLC_iSecRef_Amps variable to 6.5 A. If, for some reason, the measured current is different from the 6.5 A, set the Ref close to the measured value. As there is no soft start in the software, it is critical to keep this reference close to the operating point.
7. Now, set the CLLLC_closeGiLoop variable to *1*. This will close the current loop and the controller will now try to regulate the current.
8. Test the closed-loop operation by varying CLLC_iSecRef_Amps from 6.3 A to 6.8 A. The user cannot vary the current too much as a resistive load is connected at the output whose voltage will change with current much more than a battery. This rapid increase in voltage can quickly put the converter in a range that exceeds the controllable range for the fixed VPRIM. Within the small range, the user can see the tracking of the current.

#### 5.2.2.4.4 Measure SFRA for Closed Current Loop

1. The SFRA is integrated in the software of this build to verify that the designed compensator provides enough gain and phase margin by measuring on hardware. To run the SFRA, keep the project running, and

navigate to *<Install directory >\C2000Ware_DigitalPower_SDK_<version>\libraries\sfra\gui\SFRA_GUI.exe*. The SFRA GUI will pop up.

2. 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, deselect the boot-on-connect option, select an appropriate COM port, and click Ok. Return to the SFRA GUI and click Connect.
3. 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 of the control card, which indicates UART activity. Once complete, a graph with the open loop plot will appear, as shown in Figure 5-25.

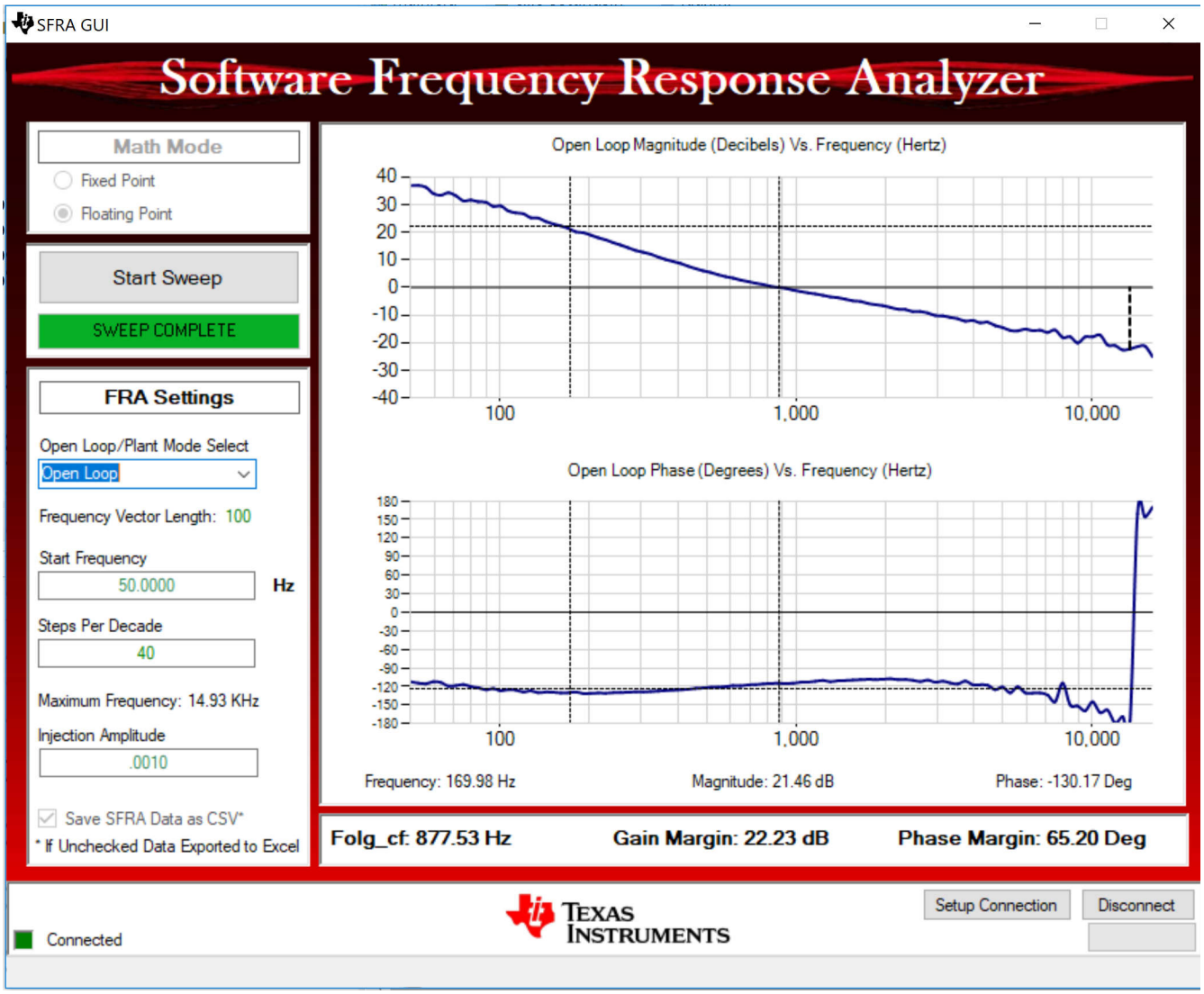



**Figure 5-25. SFRA Open Loop Plot for the Closed Current Loop (Vprim 400 V, Vsec 300 V, Power 1.972 kW, Lab 4)**

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.

The data matches closely to the designed compensator, but it is reasonable to expect deviations because the measurement in open loop is susceptible to error due to small signal injection which can drift the DC point of the converter.

Test the SFRA at different current set points, making sure the period is not clamped, to verify that the system is stable across the operable range.

4. This verifies the Lab 4 current loop design.

5. To bring the system to a safe stop, bring the input VPRIM voltage down to zero. Observe the voltages and currents on the watch window go down to zero.
6. 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 the  . Finally, reset the MCU  .
7. Close the CCS debug session by clicking on Terminate Debug Session  (Target → Terminate all).

### *5.2.2.5 Lab 5. Primary to Secondary Power Flow, Closed Current Loop Check, With Resistive Load Connected on Secondary in Parallel to a Voltage Source to Emulate a Battery Connection on Secondary Side*

In this lab, the output current control loop is closed, with a resistive load connected on the secondary in parallel with a voltage source, to emulate a battery connection. Hardware is assumed to be set up as shown in Figure 5-26. Figure 5-27 shows the complete software diagram for this build.

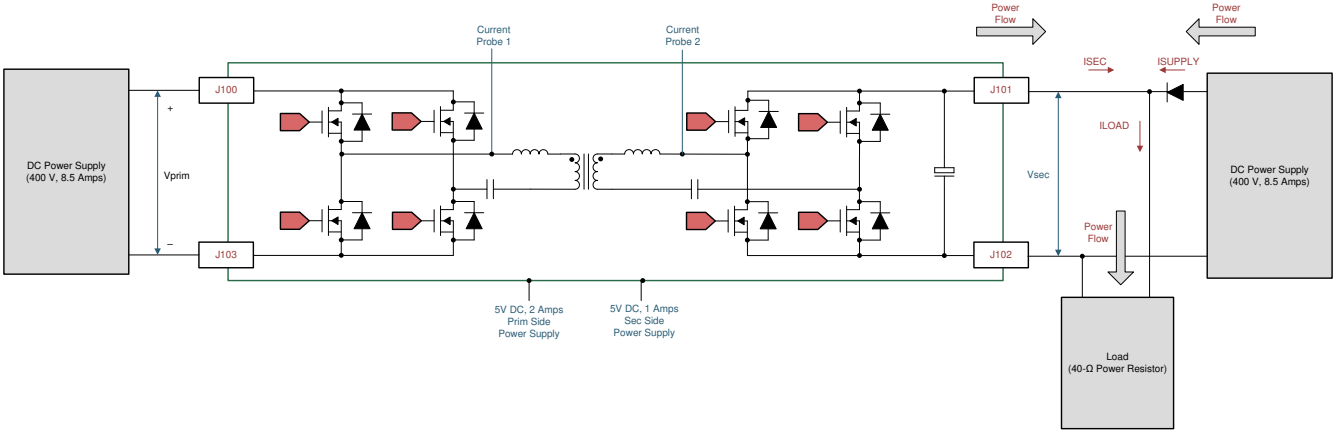



**Figure 5-26. Hardware Setup for Lab 5**

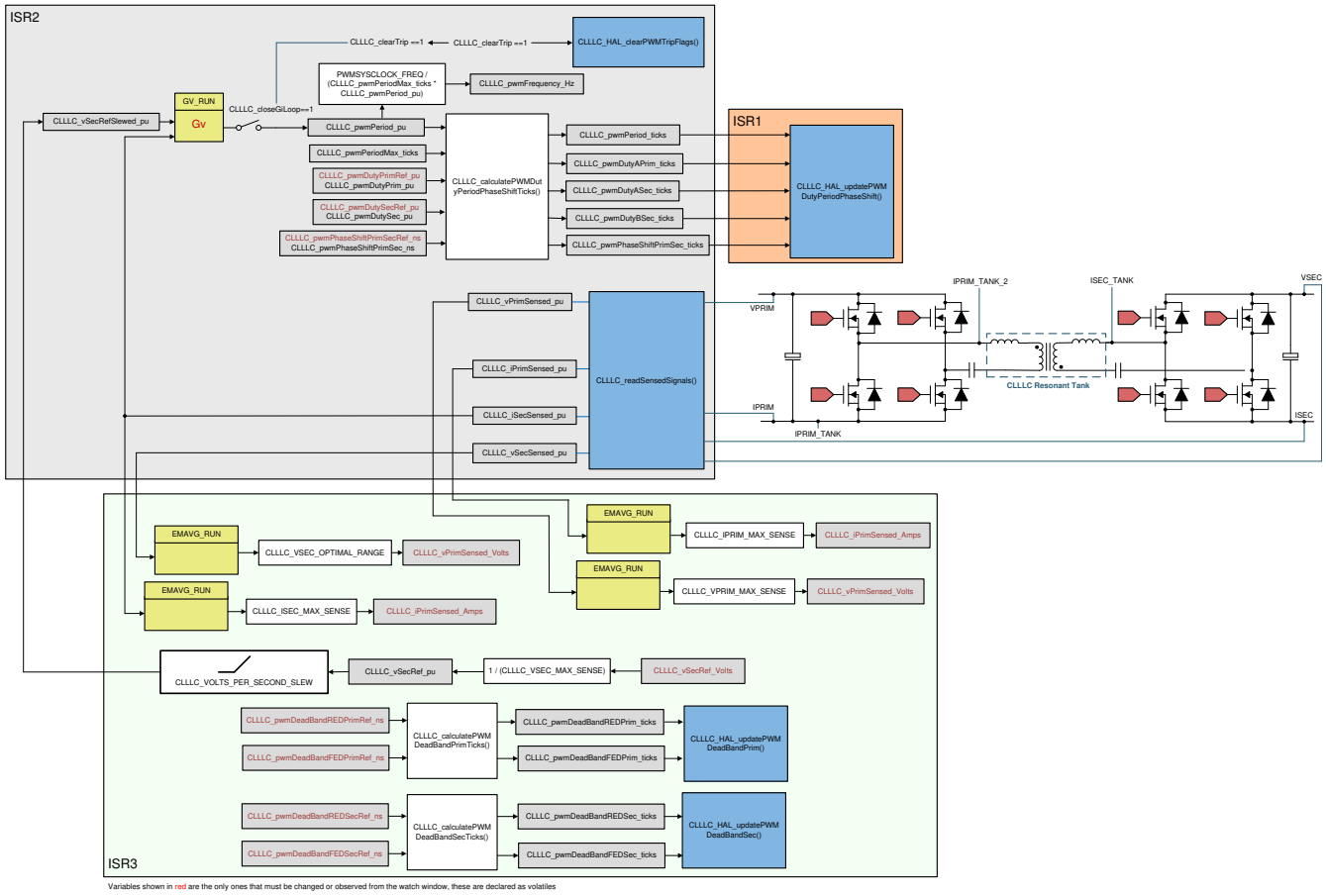



**Figure 5-27. Lab 5 Software Diagram**

#### 5.2.2.5.1 Setting Software Options for Lab 5

1. Open *<install Directory>\C2000Ware_DigitalPower_SDK_<version>\libraries\sfra\gui\CompDesigner.exe*.

#### 5.2.2.5.2 Designing Current Loop Compensator

1. The compensation designer will then launch. Currently, a mathematical model is not available; hence, using tuning done on this board, the following compensation is designed. The plant in the emulated battery mode will have more gain, and hence, the coefficients will need to be reduced to accommodate that. Figure 5-28 shows the coefficients used on this design for this lab.

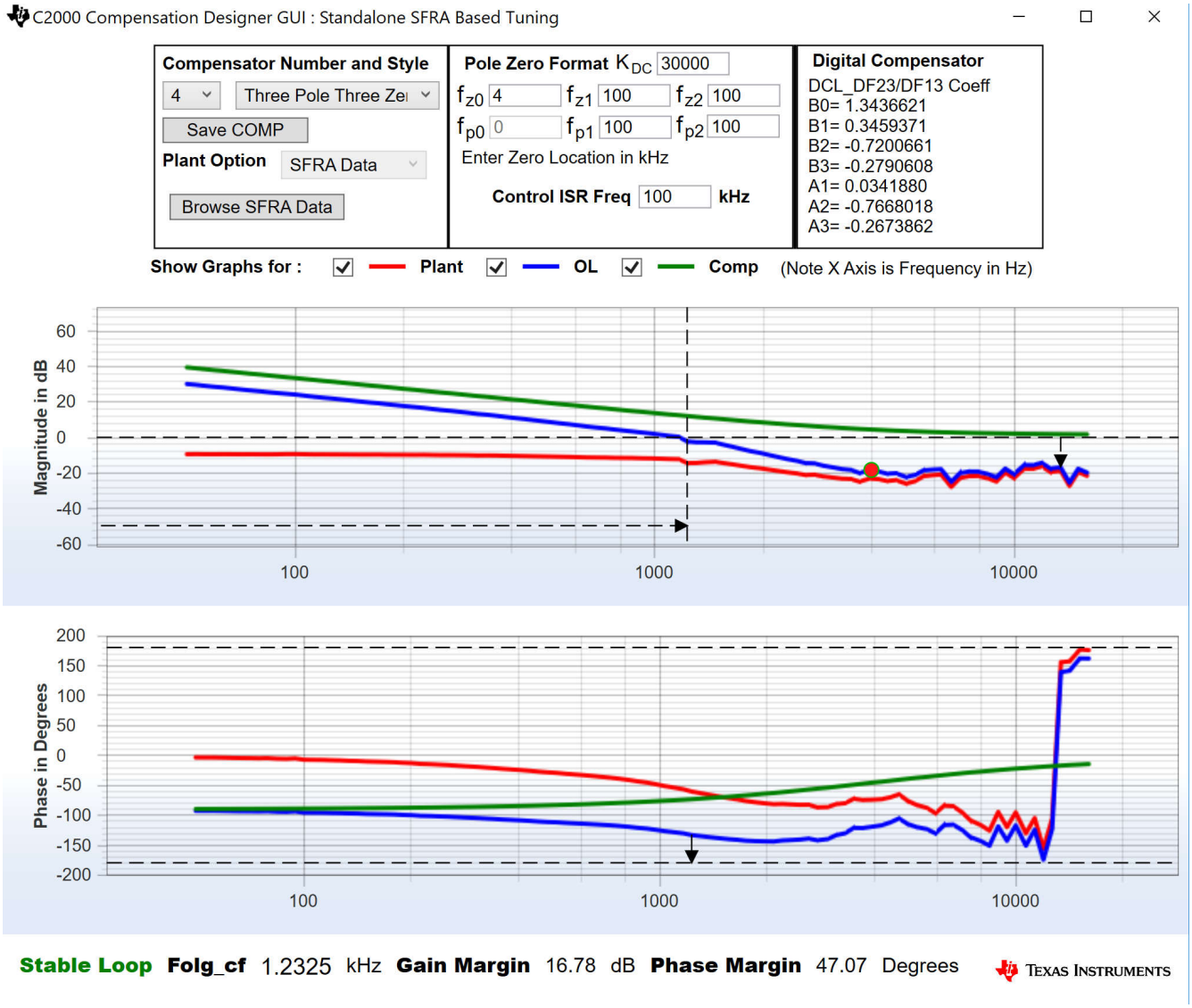
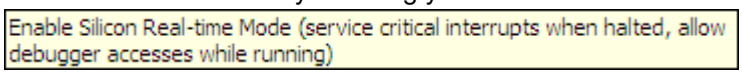



**Figure 5-28. Lab 5, Compensation Designer**

2. Once satisfied with the compensator design, the compensator values can be updated in the CLLLC_settings.h file. It is best to keep the coefficients conservative and much lower than the one used in Lab 3.
3. Close the compensation designer
4. The following defines are set in the *settings.h* file for this build.

```
#if CLLLC_LAB == 5 #define CLLLC_CONTROL_RUNNING_ON 1 #define
CLLLC_POWER_FLOW CLLLC_POWER_FLOW_PRIM_SEC #define CLLLC_INCR_BUILD CLLLC_CLOSED_LOOP_BUILD
#define CLLLC_CONTROL_MODE CLLLC_CURRENT_MODE #define CLLLC_TEST_SETUP
CLLLC_TEST_SETUP_EMULATED_BATTERY #define CLLLC_PROTECTION CLLLC_PROTECTION_ENABLED
#if CLLLC_SFRA_ALLOWED == 1 #define CLLLC_SFRA_TYPE CLLLC_SFRA_CURRENT #else
#define CLLLC_SFRA_TYPE CLLLC_SFRA_DISABLED #endif #define CLLLC_SFRA_AMPLITUDE
(float32_t)CLLLC_SFRA_INJECTION_AMPLITUDE_LEVEL1 #endif
```

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

1. Now, right-click on the project name and click *Rebuild Project*.
2. The project will build successfully.
3. Click *Run* → *Debug* to launch a debugging session. In case of dual-CPU devices, a window may appear for the user to select the CPU on which the debug is to be performed. In this case, select CPU1.
4. The project will then load on the device and the CCS debug view will become active. The code will halt at the start of the main routine.
5. 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* to browse to the *setupdebugenv_build4.js* script file located inside the project folder. This will populate the watch window with the appropriate variables needed to debug the system.
6. Click on the Continuous Refresh button  on the watch window to enable continuous update of values from the controller.
7. Enable real-time mode by hovering your mouse on the buttons on the horizontal toolbar and clicking the  

button.

#### 5.2.2.5.4 Running the Code

1. Run the project by clicking  .
2. Now, slowly increase the input PRIM DC voltage from 0 V to 400 V. Make sure CLLLC_vPrimSensed_Volts displays the correct values for VPRIM (that is, close to 400 V). At this point, the PWMs are tripped; therefore, no current will be drawn from the primary side.
3. Next, increase the VSEC to 300 V. Load will draw all the current from the secondary connected power supply, which will be close to 6.5 A.
4. Now, set the CLLLC_iSecRef_Amps variable to 0.1 A.
5. Clear the trip by writing *1* to the CLLLC_clearTrip variable. The software in this lab will automatically set the CLLLC_closeGiLoop variable to *1*.
6. Due to the narrow range of voltage at a fixed primary side voltage, the converter will saturate to the highest frequency, and the current drawn at the ISEC will be higher than 0.1 A. The user can observe this by monitoring the CLLLC_pwmFrequency_Hz variable, which will be close to 800 kHz during the upper saturation limit and 200 kHz during the lower saturation limit.
7. Slowly raise the current to be 2–3 A. Now, the current will be shared by the secondary connected voltage source and the design under test (DUT).

#### 5.2.2.5.5 Measure SFRA for Closed Current Loop in Battery Emulated Mode

1. The SFRA is integrated in the software of this build to verify that the designed compensator provides enough gain and phase margin by measuring on hardware. To run the SFRA, keep the project running, and navigate to *<Install directory >\C2000Ware_DigitalPower_SDK_<version>\libraries\sfra\gui\SFRA_GUI.exe*. The SFRA GUI will pop up.
2. 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, deselect the boot-on-connect option, select an appropriate COM port, and click *OK*. Return to the SFRA GUI and click *Connect*.
3. 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 of the control card, which indicates UART activity. Once complete, a graph with the open loop plot will appear, as shown in Figure 5-29.

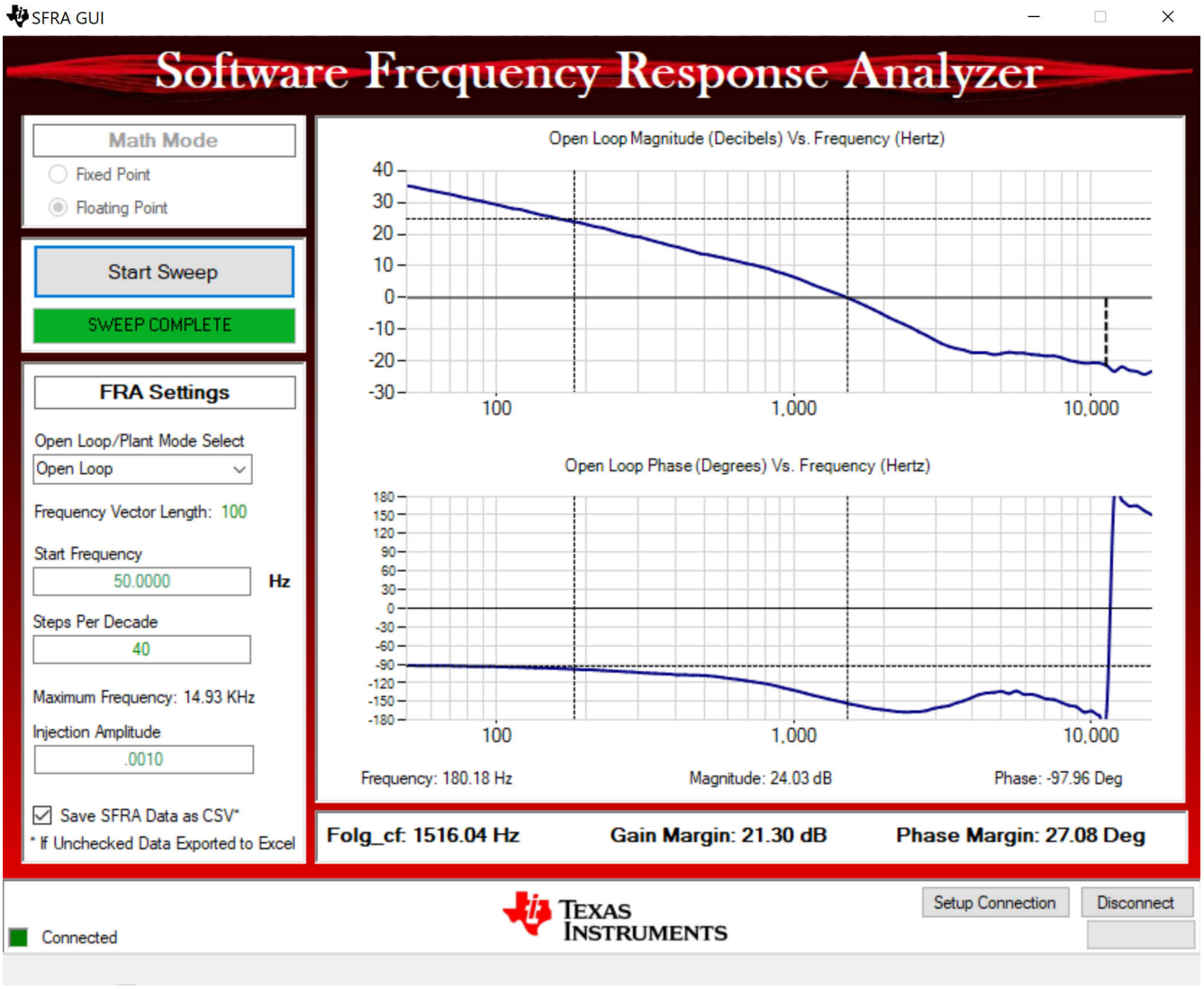



**Figure 5-29. SFRA Open Loop Plot for the Closed Current Loop With Battery Connection Emulated (Vprim 400 V, Vsec 300 V, Power 1.972 kW, Lab 5)**

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.

Test the SFRA at different current set points, making sure the period is not clamped, to verify that the system is stable across the operable range.

4. This verifies the Lab 5current loop design.
5. To bring the system to a safe stop, bring the input VPRIM voltage down to zero. Observe the voltages and currents on the watch window go down to zero.
6. 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  .
7. Close the CCS debug session by clicking on *Terminate Debug Session*  (Target → Terminate all).

### 5.2.3 TTPLPFC Test procedure

#### 5.2.3.1 Lab 1: Open Loop, DC

In this build, the board is excited in 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 operation of the PWM gate driver and ensures that 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 5-30. There are two ISR in the system: fast ISR for the current loop and a slower ISR to run the voltage loop and instrumentation functions. Modules that are run in each ISR are shown in Figure 5-30 (Note that TIDM-02013 is a 2 phase interleaved TTPLPFC).

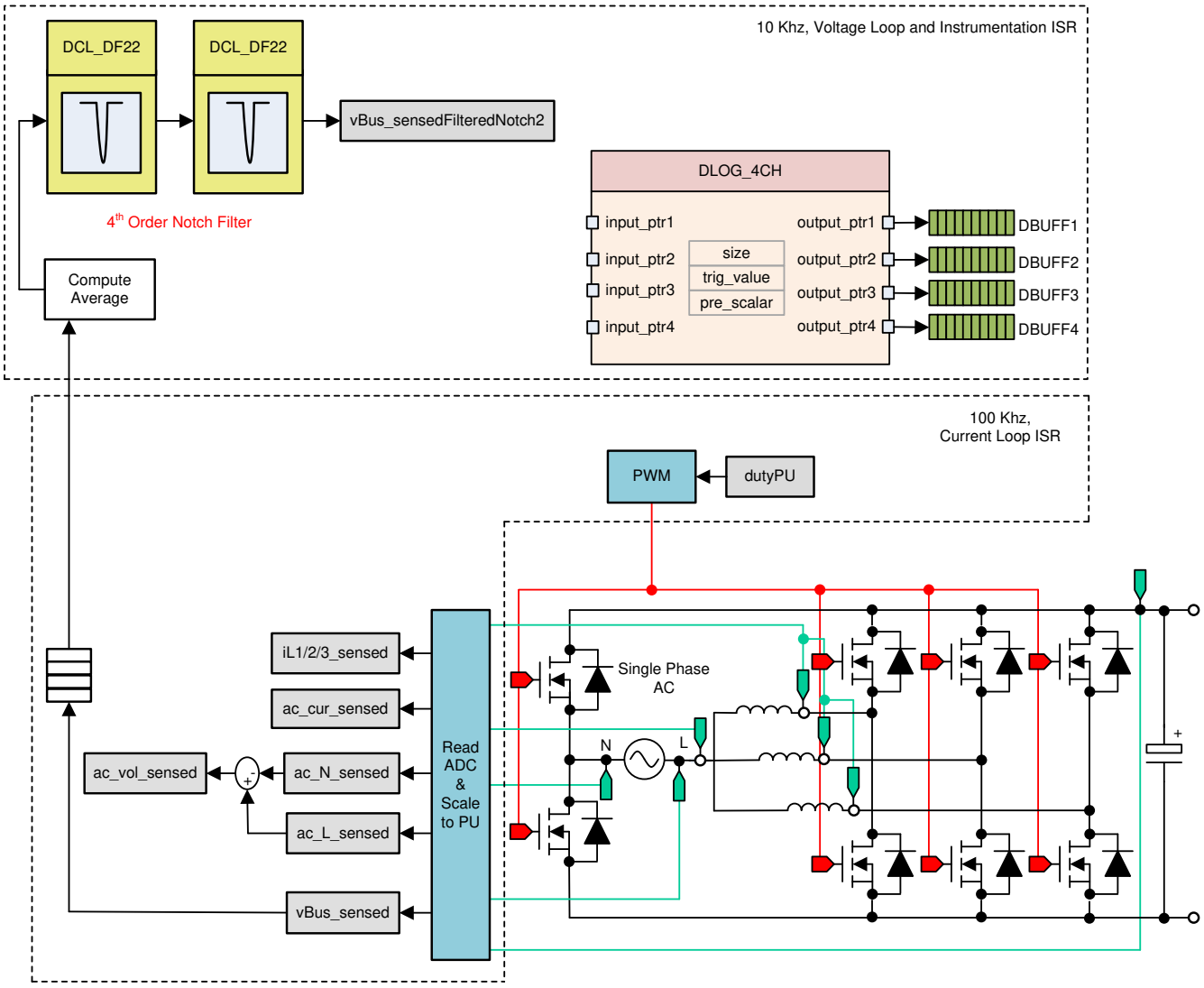



**Figure 5-30. Build Level 1 Control Software Diagram: Open Loop Project**

##### 5.2.3.1.1 Setting Software Options for BUILD 1

Open TTPLPFC_settings.h and enable Lab1

```
#define TTPLPFC_LAB 1
```

##### 5.2.3.1.2 Building and Loading Project

Right-click on the project name, and click *Rebuild Project*.

The project builds successfully.

In the *Project Explorer* ensure that the correct target configuration file is set as Active under targetConfigs.

Click *Run → Debug*. This action launches a debugging session. In the case of dual CPU devices, a window may appear to select the CPU that the debug must be performed. In this case, select CPU1.

The project then loads on the device, and CCS debug view becomes active. The code halts at the start of the main routine.

#### 5.2.3.1.3 Setup Debug Environment Windows

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. Browse to the *setupdebugenv_lab1.js* script file located inside the project folder. This script file populates the watch window with appropriate variables required 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 appears as shown in Figure 5-31.

Variables | Expressions | Registers | Breakpoints

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

**Figure 5-31. Build Level 1 Expressions View**

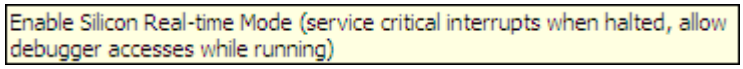
Run the project by clicking on  .

Halt the processor by using the *Halt* button on the toolbar (  ).

#### 5.2.3.1.4 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 for changing of values in watch or memory windows and the ability to see the effect of these changes in the system without halting the processor.

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




A message box may appear. If so, select *YES* to enable debug events. This action sets 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.

#### 5.2.3.1.5 Running Code

Run the project again by clicking on  .

After a few seconds, the inrush relay clicks. The software is programmed to do so in the build level with DC. The trip clears, and a duty cycle of 0.5 is applied.

In the watch view, check if the guiVin, guiVbus, guiIi, variables are updating periodically. As no power is applied, this value is close to zero.

Slowly increase the input DC voltage from zero to 120 V. The output voltage shows a boosted voltage as a steady duty cycle of 0.5 PU is applied as the default setting. If a high current is drawn, verify if the voltage terminals are swapped. If true, reduce the voltage to zero first and correct the issue before resuming the test.

Verify the voltage sensing by ensuring that *TTPLPFC_vBusAvg_pu* displays the correct value. For 120-V DC input. This verifies the voltage sensing of the board in some manner.

The dutyPU_DC variable can be changed to see operation under various boost conditions. This verifies at a basic level the PWM driver and connection of hardware

Once finished, reduce the input voltage to zero and watch for the bus voltages to reduce down to zero.

This completes the check for this build. The following items are verified on successful completion of this build:

- Sensing of voltages and currents and scaling for accuracy
- Interrupt generation and execution of the BUILD 1 code in the current loop ISR and Voltage Loop Instrumentation ISR
- PWM driver and switching

If any issues are observed, a careful inspection of the hardware may be required to eliminate any build issues and so forth.

The controller can now be halted, and the debug connection terminated.

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 .

Close CCS debug session by clicking on *Terminate Debug Session* (*Target* → *Terminate all*).

#### *5.2.3.2 Lab 2: Closed Current Loop DC*

In 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 feed forward are applied to the output of this current compensator to generate the duty cycle of the inverter. This 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. Complete software diagram for this build is illustrated in Figure 5-32 (Note that TIDM-02013 is a 2 phase interleaved TTPLPFC).

$$\text{duty1PU} = \frac{(\text{ac\_cur\_meas} - \text{ac\_cur\_ref\_inst}) \times \text{Gi} + \text{ac\_vol\_sensed}}{\text{vBus\_sensed}} \quad (21)$$

Complete software diagram for this build is illustrated in Figure 5-32.

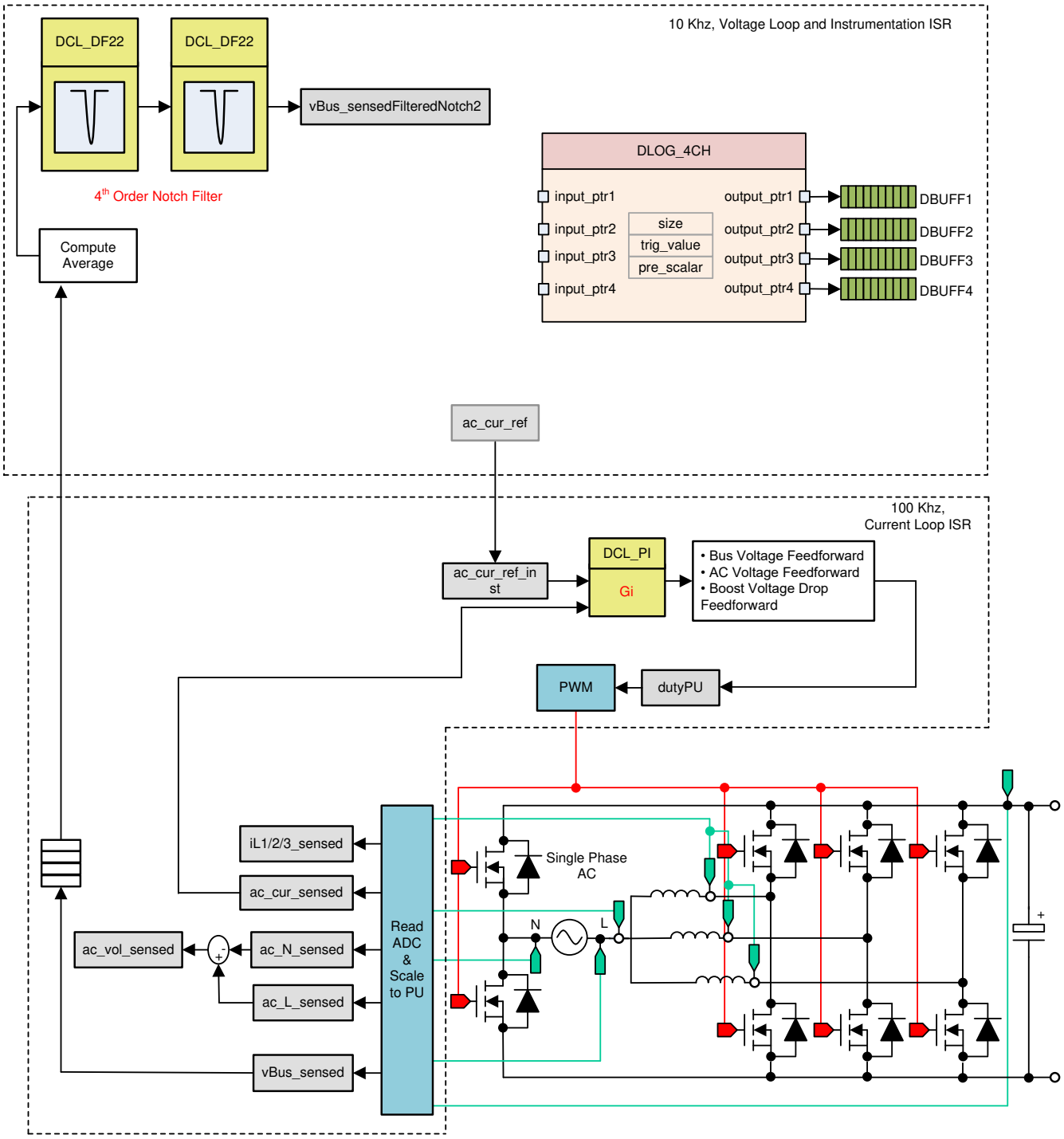



**Figure 5-32. Build Level 2 Control Software Diagram: Closed Current Loop**

#### 5.2.3.2.1 Setting Software Options for BUILD 2

Ensure that the hardware is set up as outlined in Section 5.1.1 for stand alone PFC operation. Do not supply any high voltage (HV) power to the board yet.

Open TTPLPFC_settings.h and enable Lab2

```
#define TTPLPFC_LAB 2
```

Ensure that all other options are the same as specified earlier in Section 5.2.3.2.

1. Open compensation designer *<install Directory>\C2000Ware_DigitalPower_SDK_<version>\libraries\sfra\gui\CompDesigner.exe*

#### 5.2.3.2.2 Designing Current Loop Compensator

Compensation Designer launches. PI-based controller can be tuned from a pole zero perspective 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 5-33.

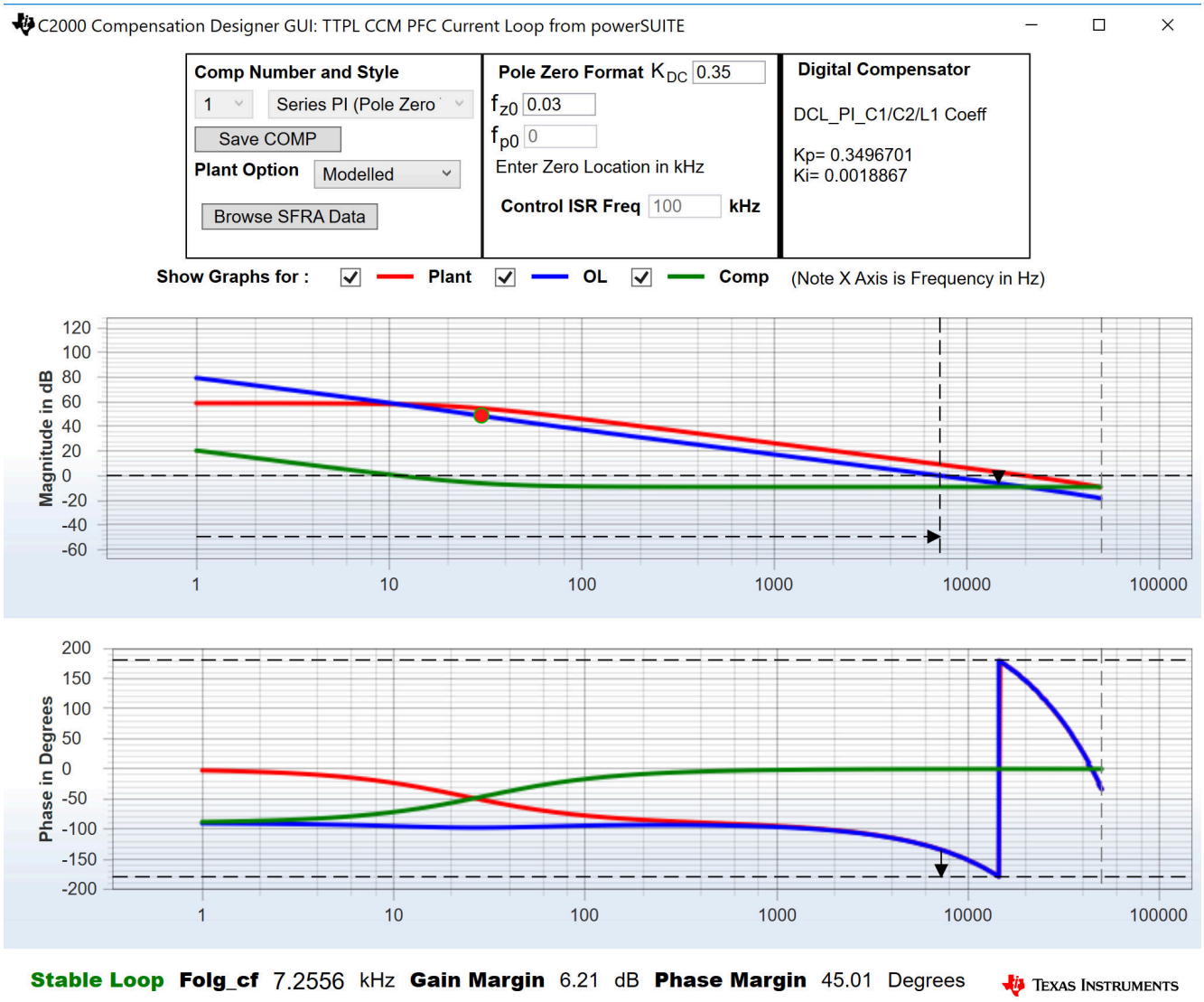



**Figure 5-33. Current Loop Design Using Compensation Designer**

Once satisfied with the open loop gain, the compensator values can be updated in the ttplpfc_settings.h file.

Close the Compensation Designer

#### 5.2.3.2.3 Building and Loading Project and Setting Up Debug

Right-click on the project name, and click *Rebuild Project*. The project builds successfully. Click *Run → Debug*, which launches a debugging session. In the case of dual CPU devices, a window may appear to select the CPU the debug must be performed. In this case, select CPU1. The project then loads on the device, and CCS debug view becomes active. The code halts at the start of the main routine.

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_lab2.js* script file, which is located inside the project folder. This file populates the watch window with appropriate variables required 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 appears as Figure 5-34.

Variables | Expressions | Registers | Breakpoints

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

**Figure 5-34. Build Level 2: Closed Current Loop Expressions View**

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

Run the project by clicking on .

Halt the processor by using the *Halt* button on the toolbar .

#### 5.2.3.2.4 Running Code

The project is programmed to drive the inrush relay and clear the trip after a set amount of time, that is, autoStartSlew==100. The software is programmed to do so in the build level with DC. An input voltage must be applied after hitting run and before this autoslew counter reaches 100. If the counter reaches 100, before voltage is applied at the input, the code must be reset. For which the controller must be brought out of real time mode, a reset performed and restarted. Repeat the step from Section 5.2.3.2.3 of enabling real-time mode by hovering the mouse on the buttons on the horizontal toolbar and click the button.

Run the project by clicking .

Apply an input voltage of approximately 50 V before the autoStartSlew reaches 100. As soon autoStartSlew reaches 100, the inrush relay is triggered, and PWM trip is cleared along with closing the current loop flag.

Variables | Expressions | Registers | Breakpoints

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

**Figure 5-35. Watch Expression, Build lab 2, DC After Closed Current Loop Operation Begins**

The input current regulates approximately 1.5 A, and the output voltage boosts to approximately 193 V.

Slowly increase ac_cur_ref to 0.045, that is, 2.4-A input.

Slowly increase $V_{in}$ = 120 V, and the output voltage will be greater than 370 V.

Variables | Expressions | Registers | Breakpoints

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

**Figure 5-36. Watch Expression, Build lab 2, DC After Closed Current Loop Operation Begins at Full Voltage**

SFRA is integrated in the software of this build to verify that the designed compensator provides enough gain and phase margin by measuring on hardware. To run the SFRA, keep the project running, and navigate to *<Install directory >\C2000Ware_DigitalPower_SDK_<version>\libraries\sfra\gui\SFRA_GUI.exe*. The SFRA GUI appears.

Select the options for the device on the SFRA GUI. For example, for F28003x, select floating point. Click on *Setup Connection*. On the pop-up window, uncheck the boot on connect option, and select an appropriate COM port. Ensure *Boot on Connect* is deselected. Click *OK*. Return to the SFRA GUI, and click *Connect*.

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. 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 appears. 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.

Additionally, the measured frequency response of the plant can be used to design the current compensator with compensation designer. *<install Directory>\C2000Ware_DigitalPower_SDK_<version>\libraries\sfra\gui\CompDesigner.exe*.

Choose *SFRA Data* for plant option on the GUI. This uses the measured plant information to design the compensator. This option can be used to fine tune the compensation. By default, the compensation designer points to the latest SFRA run. If a previous SFRA run plant information must be used, select the SFRAData.csv file by browsing to it by clicking on *Browse SFRA Data*. This action verifies the current compensator design.

Bring the system to a safe stop by bringing the input DC voltage down to zero. Ensure that the guiVbus comes down to zero as well.

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 (  ) .

Close the CCS debug session by clicking on *Terminate Debug Session* (*Target* → *Terminate all*).


#### *5.2.3.3 Lab 3: Closed Current Loop, AC*

In Lab 3, the inner current loop is closed, that is, the inductor current is controlled using a current compensator Gi. Both DC bus and output voltage feed forward are applied to the output of this current compensator to generate the duty cycle of the inverter along with soft start for PWM around the zero-crossing.

Complete software diagram for this build as illustrated in Figure 5-37 (Note that TIDM-02013 is a 2 phase interleaved TTPLPFC).

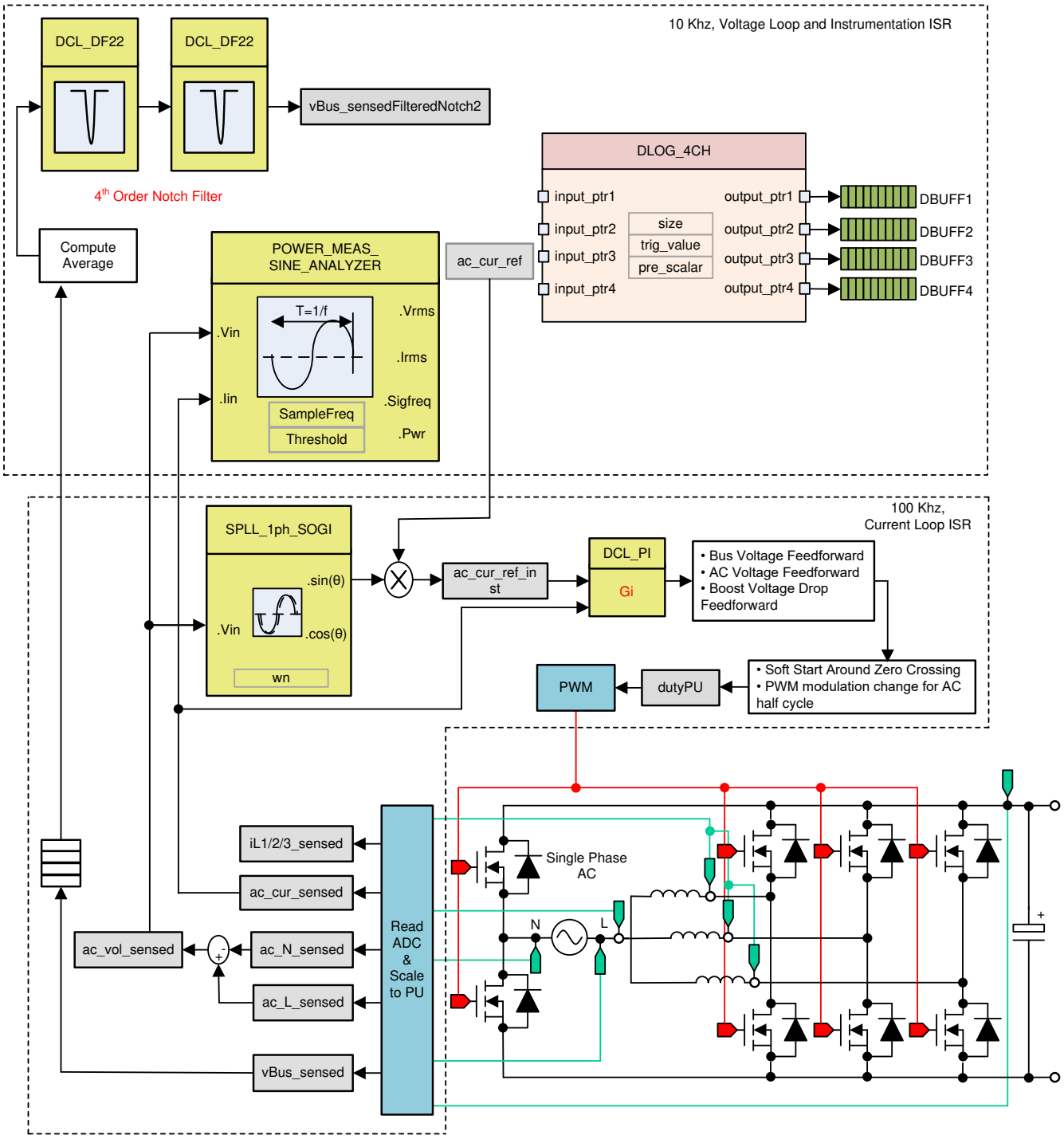



**Figure 5-37. Build Lab3 Control Software Diagram: Closed Current Loop AC**

#### 5.2.3.3.1 Setting Software Options for Lab 3

Open TTPLPFC_settings.h and enable Lab3

```
#define TTPLPFC_LAB 3
```

Current compensator from the previous build is re-used in this build so no additional steps are required for tuning the current loop in the build level.

#### 5.2.3.3.2 Building and Loading Project and Setting Up Debug

Right-click on the project name, and click *Rebuild Project*. The project builds successfully. Click *Run* → *Debug*, which launches a debugging session. In the case of dual CPU devices, a window may appear to select the CPU the debug must be performed. In this case, select CPU1. The project then loads on the device, and CCS debug view becomes active. The code halts at the start of the main routine.

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_Lab3.js* script file, which is located inside the project folder. This file populates the watch window with appropriate variables required 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 appears as Figure 5-38.

Variables | Expressions | Registers | Breakpoints

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

**Figure 5-38. Lab3 AC: Closed Current Loop Expressions View**

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

#### 5.2.3.3.3 Running Code

The project is programmed to wait for input voltage to exceed approximately 70 $V_{rms}$ to drive the in rush relay and clear the trip.

Run the project by clicking .

Apply an input voltage of approximately 120 V. The board comes out of the undervoltage condition and inrush relay is driven. The trip clears, and a small amount of current of approximately 1.3-A RMS is drawn. The watch window looks similar to Figure 5-39. The bus voltage is close to 270 V.

Variables | Expressions | Registers | Breakpoints

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

**Figure 5-39. Watch Expression, Lab2, AC After Closed Current Loop Operation Begins**

Slowly increase ac_cur_ref to 0.078, that is, 2.4-A input, and the bus voltage rises to 400 V. The voltage and current waveform are shown in Figure 5-40.

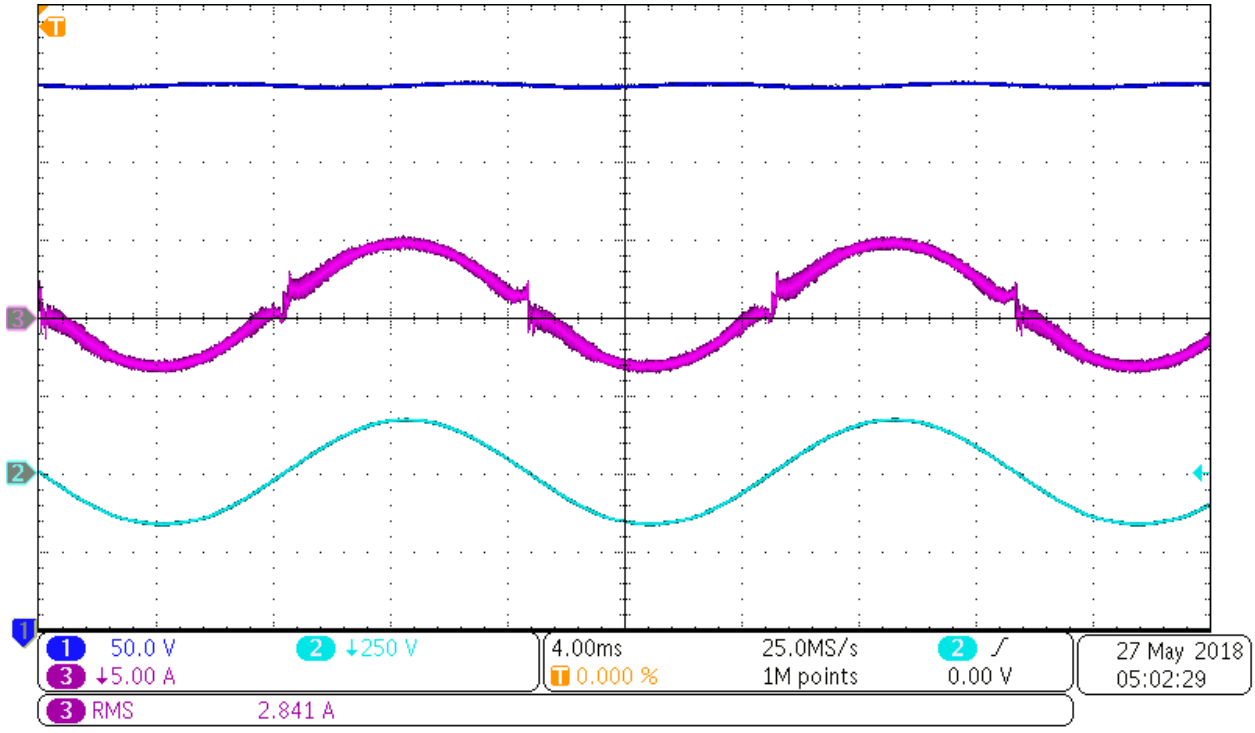



**Figure 5-40. Input AC Current and Output DC Voltage Waveform**

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 navigate to *<Install directory >\C2000Ware_DigitalPower_SDK_<version>\libraries\sfra\gui\SFRA_GUI.exe*. The SFRA GUI appears.

Select the options for the device on the SFRA GUI. For example, for F280039, select floating point. Click on *Setup Connection*. On the pop-up window, deselect the boot on connect option, and select an appropriate COM port. Click *OK*. Return to the SFRA GUI, and click *Connect*.

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. 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 indicates UART activity. Once complete, a graph with the open loop plot appears. This is similar to the plot seen under DC conditions; however, some additional noise is visible due to AC harmonic frequencies close to the measured frequencies. The BW, PM, and GM numbers are very similar to the DC case.

To bring the system to a safe stop, switch off the output from the AC power supply, thus bringing the input AC voltage down to zero. Ensure that the guiVbus comes down to zero, as well.

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 (  ) .

Close the CCS debug session by clicking on *Terminate Debug Session* (*Target* → *Terminate all*).


#### *5.2.3.4 Lab 4: Closed Voltage and Current Loop*

In this build, the outer voltage loop is closed with the inner current loop closed. The model of the outer voltage loop is derived in Figure 5-41(Note that TIDM-02013 is a 2 phase interleaved TTPLPFC). A PI-based compensator is used and tuned through the compensation designer for the outer voltage loop.

Figure 5-41 shows the software diagram for this build.

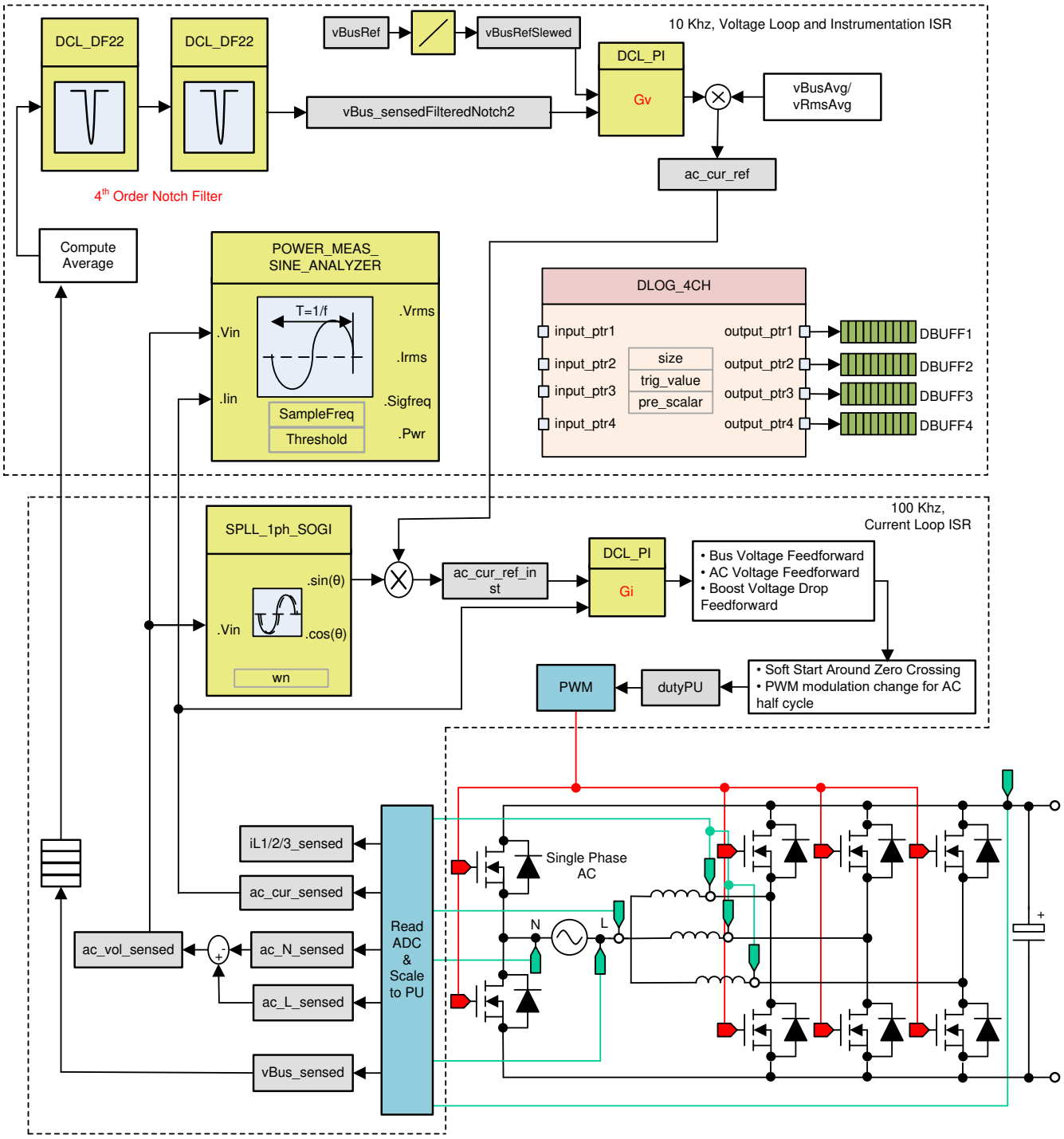



**Figure 5-41. Build Level 4 Control Diagram: Output Voltage Control With Inner Current Loop**

#### 5.2.3.4.1 Setting Software Options for BUILD 4

Ensure that the hardware is set up as outlined in Section 5.1.1 for stand alone PFC operation. Do not supply any high voltage (HV) power to the board yet.

Open TTPLPFC_settings.h and enable Lab4

```
#define TTPLPFC_LAB 4
```

Ensure that all other options are the same as specified earlier in Figure 5-41.

1. Open compensation designer *<install Directory>\C2000Ware_DigitalPower_SDK_<version>\libraries\sfra\gui\CompDesigner.exe*

#### 5.2.3.4.2 Building and Loading Project and Setting up Debug

Right-click on the project name, and click *Rebuild Project*. The project builds successfully. Click *Run → Debug*, which launches a debugging session. In the case of dual CPU devices, a window may appear to select the CPU the debug must be performed. In this case, select CPU1. The project then loads on the device, and CCS debug view becomes active. The code halts at the start of the main routine.

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_lab4.js* script file located inside the project folder. This file populates the watch window with appropriate variables required to debug the system. Click the *Continuous Refresh* button ( ) on the watch window to enable continuous update of values from the controller.
The watch window appears as shown in Figure 5-42.

Variables | Expressions | Registers

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

**Figure 5-42. Build Lab4: Expressions View**

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

Run the project by clicking on .

Halt the processor by using the *Halt* button on the toolbar ( ) .

#### 5.2.3.4.3 Running Code

The project is programmed to wait for input voltage to excel at approximately 70 $V_{rms}$ to drive the in rush relay and clear the trip.

Run the project by clicking .

Apply an input voltage of approximately 120 V. The board comes out of the undervoltage condition and inrush relay is driven. The trip clears, and the output rises to 380-V DC. A sinusoidal current is drawn from the AC input. Figure 5-43 shows the watch window when the program is running at this stage.

Variables | Expressions | Registers

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

**Figure 5-43. Build Lab4: Expressions View After AC Voltage is Applied**

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 navigate to *<Install directory >\C2000Ware_DigitalPower_SDK_<version>\libraries\sfra\gui\SFRA_GUI.exe*. The SFRA GUI appears.

Select the options for the device on the SFRA GUI. For example, for F28003x, select floating point. Click on *Setup Connection*, and on the pop-up window, deselect the boot on connect option and select an appropriate COM port. Click *OK*. Return to the SFRA GUI, and click *Connect*.

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. 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 appears, as seen in Figure 5-44. This action verifies that the designed compensator is indeed stable.

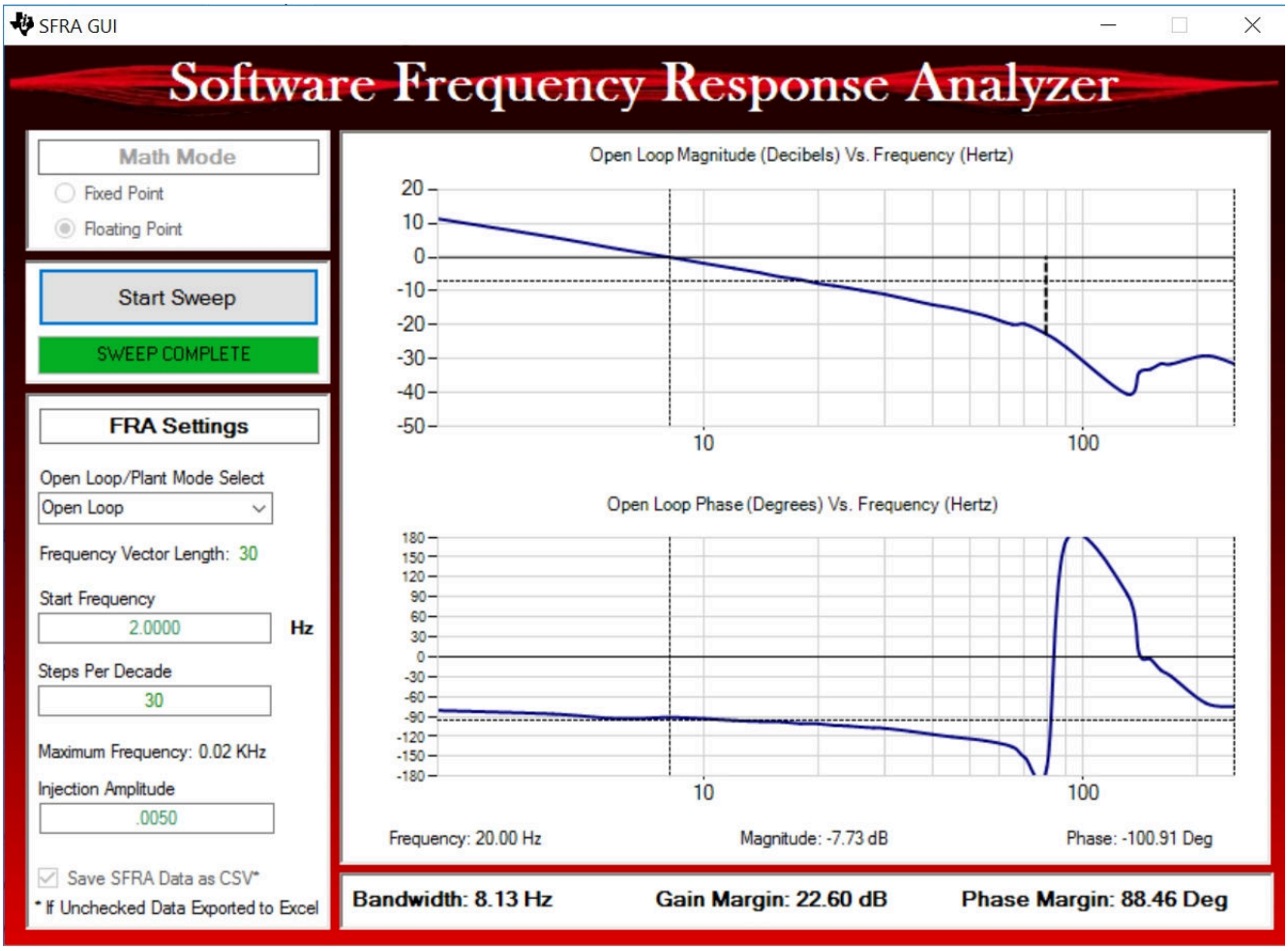



**Figure 5-44. SFRA Run on Closed Voltage Loop**

Alternately, re-open Compensation Designer, and choose *SFRA Data* for plant option on the GUI. This option uses the measured plant information to design the compensator, and can be used to fine tune the compensation. By default, the Compensation Designer points to the latest SFRA run. If a previous SFRA run plant information must be used, select the *SFRAData.csv* file by browsing to it by clicking on *Browse SFRA Data*. Close the Compensation Designer, This verifies the voltage compensator design.

To bring the system to a safe stop, bring the input AC voltage down to zero. Ensure that the guiVBus comes down to zero, as well.

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 ( ).

Close CCS debug session by clicking on *Terminate Debug Session* (*Target* → *Terminate all*).

### 5.2.4 Test Results

The power density achieved in this design is 3.8 kW/L (62.5 W/in$^3$). The total system efficiency is 96.5%. The PFC has an efficiency of 98.5% and the CLLLC is 98%.

#### 5.2.4.1 Efficiency

Efficiency data is provided in the following graphs with and without 12-V bias power. The bias supplies power to control, isolators, and gate drive. The graph in Figure 5-45 was taken under the following conditions:

- $V_{IN,RMS}$ = 240 V
- $V_{OUT}$ = 400 V
- Coolant Temperature: 20°C

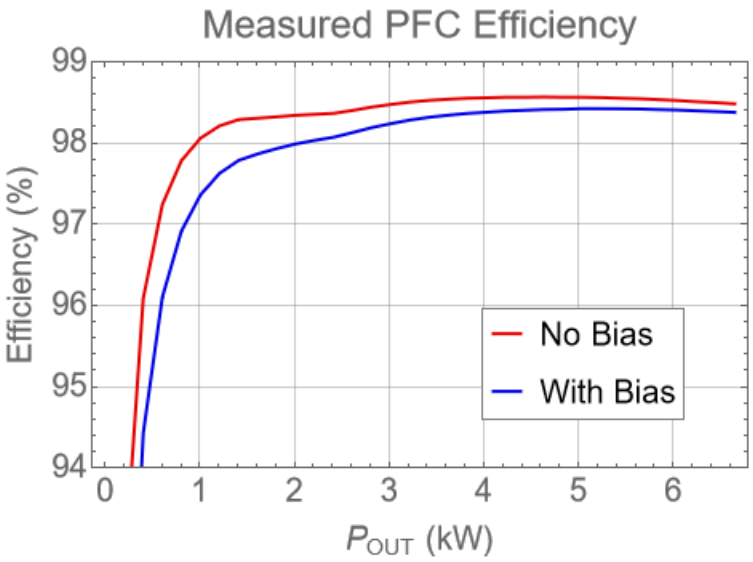



**Figure 5-45. PFC Efficiency**

The graph in Figure 5-46 was taken under the following conditions:

- $V_{IN}$ = 400 V
- $V_{OUT}$ = 350 V
- Coolant Temperature: 20°C

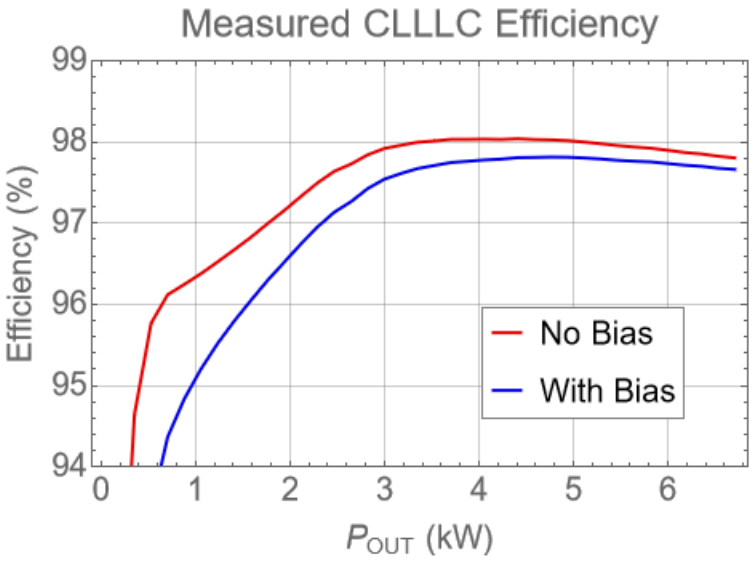



**Figure 5-46. CLLLC Efficiency**

The graph in Figure 5-47 was taken under the following conditions:

- $V_{IN,RMS}$ = 240 V
- $V_{OUT}$ = 350 V
- Coolant Temperature: 20°C

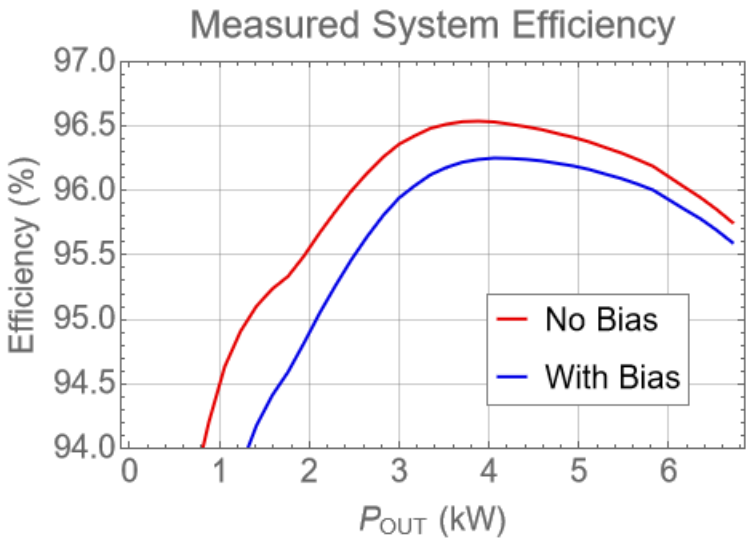



**Figure 5-47. System Efficiency**

### 5.2.4.2 System Performance

The following figures summarize the total system efficiency, system losses, total harmonic distortion (THD), and the normalized output voltage regulation accuracy.

The power density achieved in this design is 3.8 kW/L (62.5 W/in$^3$). This comes with a total system efficiency of 96.5%. Loads above 1.5 kW have a THD < 5% and the regulation accuracy of the output voltage is roughly within ±0.06%.

The graphs in Figure 5-48 use the following conditions:

- $V_{IN,RMS}$ = 240 V
- $V_{OUT}$ = 350 V
- Coolant Temperature: 20°C

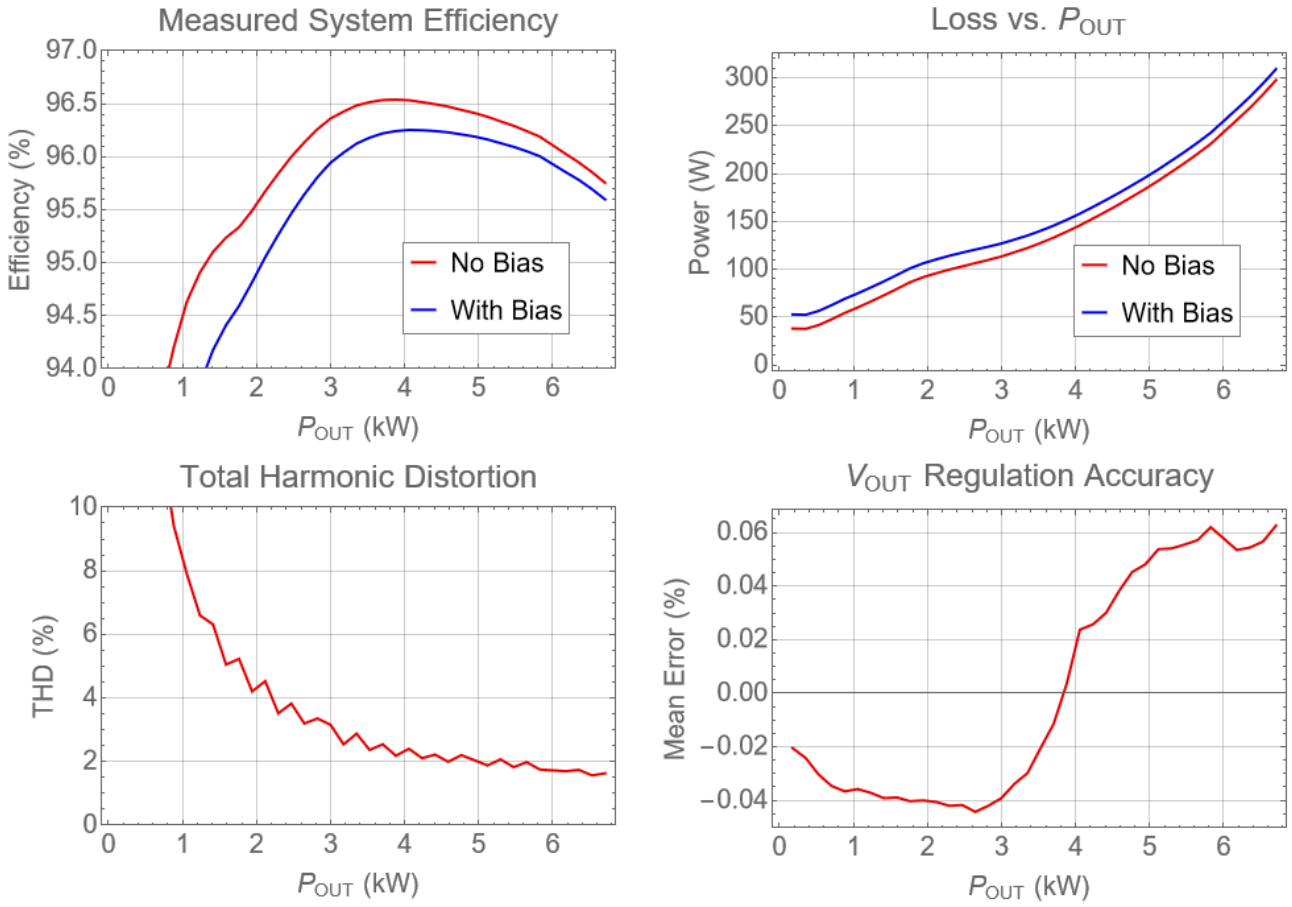
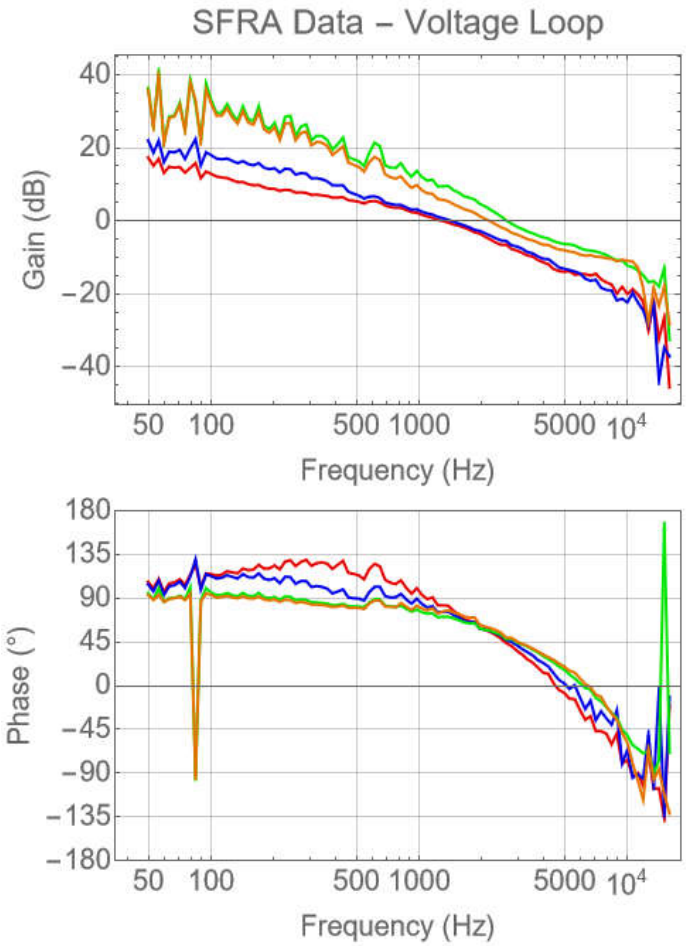



**Figure 5-48. System Performance**

### 5.2.4.3 Bode Plots

The following bode plots were acquired using the onboard software frequency response analyzer inside the TMS320F28388D microcontroller. The load used in the tests was configured as a constant current sink. The microcontroller is configured to regulate a constant output voltage. The bandwidth is roughly from 1 kHz to 2.5 kHz with a phase margin in excess of 45°C.




| | $V_{OUT}$ | $I_{OUT}$ | $f_{bw}$ | $\phi_m$ |
|---|---|---|---|---|
| — | 350 V | 18.8 A | 1.28 kHz | 82.5° |
| — | 350 V | 15 A | 1.42 kHz | 74.5° |
| — | 350 V | 5 A | 2.64 kHz | 47.4° |
| — | 350 V | 2.5 A | 2.18 kHz | 59.5° |

**Figure 5-49. Voltage Loop Bode Plot**

The following bode plots were acquired using the onboard software frequency response analyzer inside the TMS320F28388D microcontroller. The load used in the tests was configured as a constant voltage. The microcontroller is configured to regulate a constant output current. The bandwidth is roughly from 1 kHz to 2.5 kHz with a phase margin in excess of 60°C.

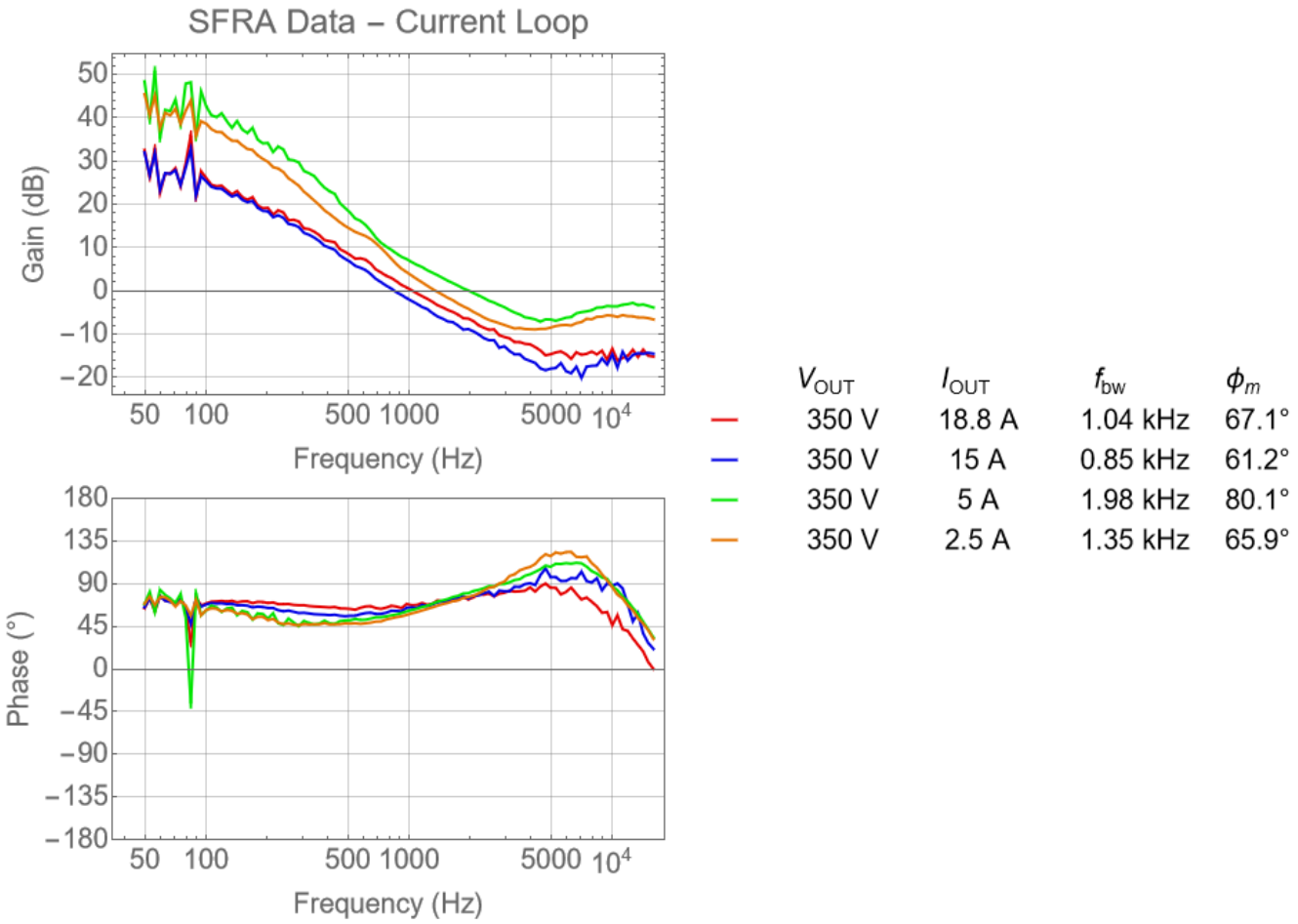



**Figure 5-50. Voltage Loop Bode Plot (Constant Voltage Load)**

### 5.2.4.4 Efficiency and Regulation Data

The following table shows efficiency and regulation data.

| $V_{OUT}$ (V) | $I_{OUT}$ (A) | $P_{OUT}$ (W) | $V_{IN}$ (V) | $I_{IN}$ (A) | $P_{IN}$ (W) | $V_{BIAS}$ (V) | $I_{BIAS}$ (A) | $P_{BIAS}$ (W) | Eff (%) No Bias | Eff (%) With Bias |
|---|---|---|---|---|---|---|---|---|---|---|
| 352.76 | 0.5 | 177.35 | 240.12 | 1.17 | 215.36 | 11.92 | 1.24 | 14.73 | 82.35 | 77.08 |
| 352.74 | 1 | 354.12 | 240.05 | 1.79 | 391.84 | 11.92 | 1.23 | 14.67 | 90.37 | 87.11 |
| 352.72 | 1.5 | 530.65 | 239.98 | 2.49 | 572.25 | 11.92 | 1.24 | 14.8 | 92.73 | 90.39 |
| 352.71 | 2 | 706.27 | 239.91 | 3.23 | 753.97 | 11.92 | 1.24 | 14.78 | 93.67 | 91.87 |
| 352.7 | 2.5 | 882.52 | 239.84 | 3.97 | 936.83 | 11.92 | 1.24 | 14.76 | 94.2 | 92.74 |
| 352.7 | 3 | 1058.9 | 239.76 | 4.72 | 1118.95 | 11.92 | 1.23 | 14.71 | 94.63 | 93.41 |
| 352.7 | 3.5 | 1235.3 | 239.69 | 5.48 | 1301.58 | 11.92 | 1.23 | 14.65 | 94.91 | 93.85 |
| 352.69 | 4 | 1411.74 | 239.62 | 6.23 | 1484.44 | 11.92 | 1.23 | 14.61 | 95.1 | 94.18 |
| 352.69 | 4.5 | 1587.99 | 239.55 | 6.99 | 1667.43 | 11.92 | 1.22 | 14.57 | 95.24 | 94.41 |
| 352.69 | 5 | 1763.62 | 239.47 | 7.75 | 1849.94 | 11.92 | 1.22 | 14.51 | 95.33 | 94.59 |
| 352.69 | 5.5 | 1939.96 | 239.4 | 8.51 | 2031.56 | 11.93 | 1.21 | 14.48 | 95.49 | 94.82 |
| 352.69 | 6 | 2116.85 | 239.32 | 9.27 | 2212.59 | 11.93 | 1.21 | 14.48 | 95.67 | 95.05 |
| 352.68 | 6.5 | 2293.18 | 239.25 | 10.02 | 2392.72 | 11.93 | 1.21 | 14.46 | 95.84 | 95.26 |
| 352.68 | 7 | 2469.51 | 239.17 | 10.78 | 2572.43 | 11.93 | 1.21 | 14.4 | 96 | 95.46 |
| 352.67 | 7.5 | 2645.17 | 239.1 | 11.53 | 2751.5 | 11.93 | 1.19 | 14.16 | 96.14 | 95.64 |
| 352.68 | 8 | 2821.8 | 239.02 | 12.29 | 2931.48 | 11.94 | 1.16 | 13.85 | 96.26 | 95.81 |
| 352.69 | 8.5 | 2998.24 | 238.95 | 13.04 | 3111.5 | 11.94 | 1.14 | 13.56 | 96.36 | 95.94 |
| 352.71 | 9 | 3174.78 | 238.87 | 13.8 | 3292.5 | 11.95 | 1.11 | 13.26 | 96.43 | 96.04 |
| 352.72 | 9.5 | 3350.46 | 238.79 | 14.56 | 3472.6 | 11.95 | 1.09 | 12.99 | 96.48 | 96.12 |
| 352.76 | 10 | 3527.33 | 238.71 | 15.33 | 3654.7 | 11.95 | 1.07 | 12.76 | 96.51 | 96.18 |
| 352.79 | 10.5 | 3704.19 | 238.63 | 16.09 | 3837.2 | 11.96 | 1.05 | 12.52 | 96.53 | 96.22 |
| 352.84 | 11 | 3881.08 | 238.54 | 16.87 | 4020.3 | 11.96 | 1.03 | 12.37 | 96.54 | 96.24 |
| 352.91 | 11.5 | 4058.4 | 238.46 | 17.64 | 4204.2 | 11.96 | 1.02 | 12.24 | 96.53 | 96.25 |
| 352.92 | 12 | 4235 | 238.37 | 18.42 | 4387.9 | 11.96 | 1.02 | 12.16 | 96.51 | 96.25 |
| 352.93 | 12.5 | 4411.3 | 238.28 | 19.2 | 4571.5 | 11.96 | 1.01 | 12.08 | 96.5 | 96.24 |
| 352.96 | 13 | 4588.2 | 238.18 | 19.98 | 4756 | 11.96 | 1 | 12 | 96.47 | 96.23 |
| 352.99 | 13.5 | 4765.1 | 238.09 | 20.77 | 4940.9 | 11.96 | 1 | 11.92 | 96.44 | 96.21 |
| 353 | 14 | 4941.9 | 237.99 | 21.55 | 5125.8 | 11.97 | 0.99 | 11.86 | 96.41 | 96.19 |
| 353.02 | 14.5 | 5118.9 | 237.89 | 22.34 | 5311.4 | 11.97 | 0.99 | 11.79 | 96.38 | 96.16 |
| 353.02 | 15 | 5294.8 | 237.79 | 23.13 | 5496.4 | 11.97 | 0.98 | 11.73 | 96.33 | 96.13 |
| 353.02 | 15.5 | 5471.6 | 237.68 | 23.92 | 5682.4 | 11.97 | 0.98 | 11.69 | 96.29 | 96.09 |
| 353.03 | 16 | 5648.3 | 237.62 | 24.71 | 5868.9 | 11.97 | 0.97 | 11.64 | 96.24 | 96.05 |
| 353.05 | 16.5 | 5825.2 | 237.51 | 25.51 | 6056.1 | 11.97 | 0.97 | 11.59 | 96.19 | 96 |
| 353.02 | 17.5 | 6176.6 | 237.26 | 27.13 | 6432.3 | 11.97 | 0.96 | 11.52 | 96.03 | 95.85 |
| 353.02 | 18 | 6353.2 | 237.14 | 27.94 | 6621.6 | 11.97 | 0.96 | 11.49 | 95.95 | 95.78 |
| 353.03 | 18.5 | 6529.8 | 237.01 | 28.76 | 6812.2 | 11.97 | 0.96 | 11.47 | 95.85 | 95.69 |
| 353.05 | 19 | 6706.8 | 236.87 | 29.59 | 7004.3 | 11.97 | 0.96 | 11.44 | 95.75 | 95.6 |

#### 5.2.4.5 Thermal Data

Figure 5-51 is taken under full load operation. All of the significant heat generation components are connected to the cold plate on the bottom side of the board. The hottest components visible in this image come from the common-mode inductors in the EMI filter. These parts have no access to the cold plate and receive all their cooling through the ambient air.

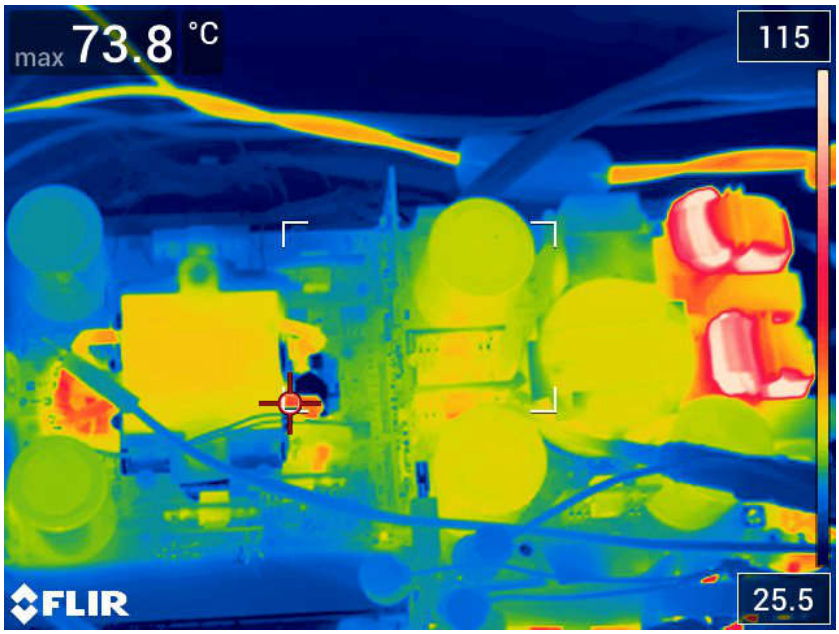



**Figure 5-51. Top Side Thermal Image**

GaN FET temperatures are provide by means of the onboard temperature sensors inside the LMG3522 devices. Under full load conditions, all FET temperatures are less than 75°C.

Table 5-5 lists the GaN FET temperature measurements under the following conditions:

- $V_{IN,AC}$: 240 V
- $V_{DC,LINK}$: 400 V
- Coolant temperature: 33°C

**Table 5-5. GaN FET Temperature Measurements**

| GaN FET | Temperature (°C) |
|---|---|
| **PFC** | 66.8 |
| **CLLLC Primary (350 V/19 A)** | 58.1 |
| **CLLLC Secondary (350 V/19 A** | 59.5 |
| **CLLLC Primary (300 V/19 A)** | 61.0 |
| **CLLLC Secondary (300 V/19 A)** | 74.0 |

Figure 5-52 shows the critical transformer temperatures under the following conditions:

- Coolant Temperature: 33°C
- Transformer temperature measurement locations
  - PRI 1 – Measured on the inside surface of the primary winding
  - PRI 2 – Measured on the outside surface of the primary winding
  - SEC 1 – Measured on the inside surface of the secondary winding
  - SEC 2 – Measured on the outside surface of the secondary winding
  - CORE 1 – Measured on the top of the core center leg
  - CORE 2 – Measured on the bottom of the core center leg
  - CORE 3 – Measured on the side of the core
  - CORE 4 – Measured on the top of the core

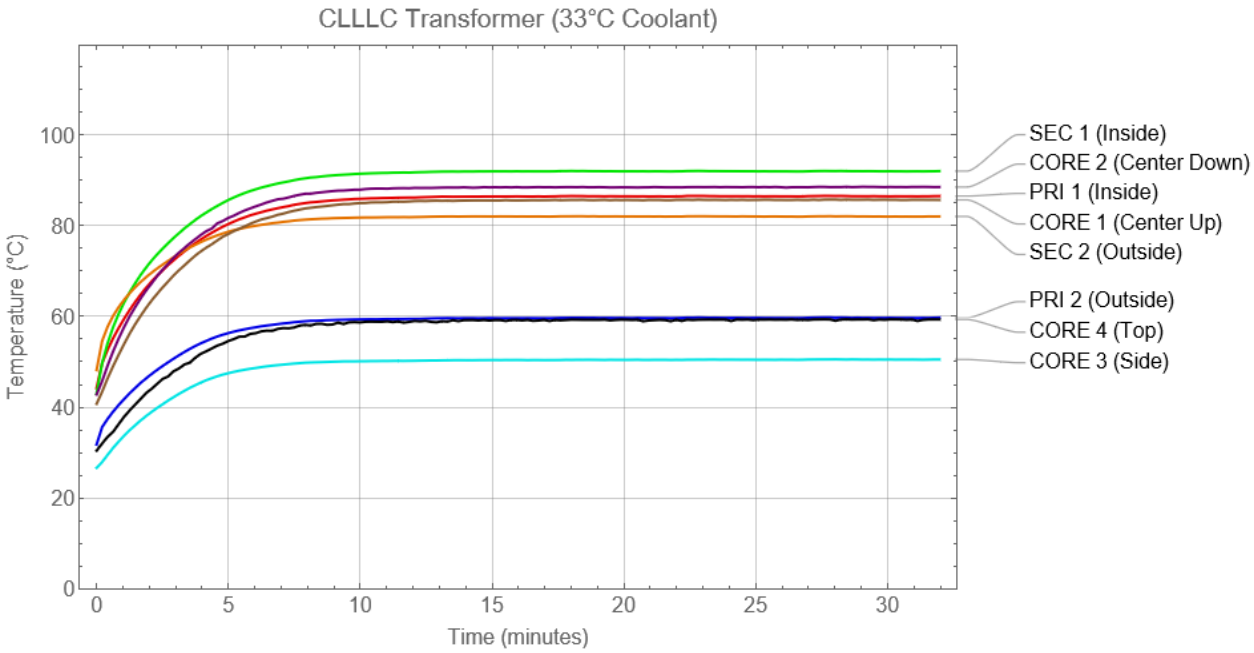



**Figure 5-52. CLLLC Transformer Temperatures**

### *5.2.4.6 PFC Waveforms*

Figure 5-53 shows the PFC input voltage and input current waveform measured at the following parameters:

- Traces
  - C2: $V_{IN}$
  - C4: $I_{IN}$
- Conditions
  - $V_{IN}$ = 208 V
  - $V_{OUT}$ = 400 V
  - $R_{OUT}$ = 43 Ω

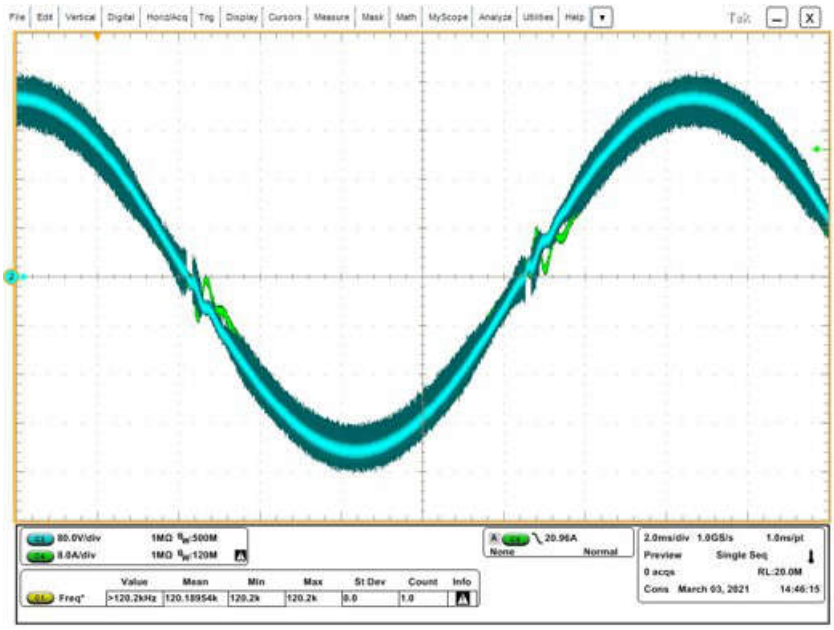


**Figure 5-53. PFC Input Voltage and Input Current**

Figure 5-54 shows the PFC GaN drain voltage waveform measured at the following parameters:

- Traces
  - C1: GaN Switch Node Drain Voltage
  - C2: $V_{IN}$
  - C4: $I_{IN}$
- Conditions
  - $V_{IN}$ = 208 V
  - $V_{OUT}$ = 400 V
  - $R_{OUT}$ = 43 Ω

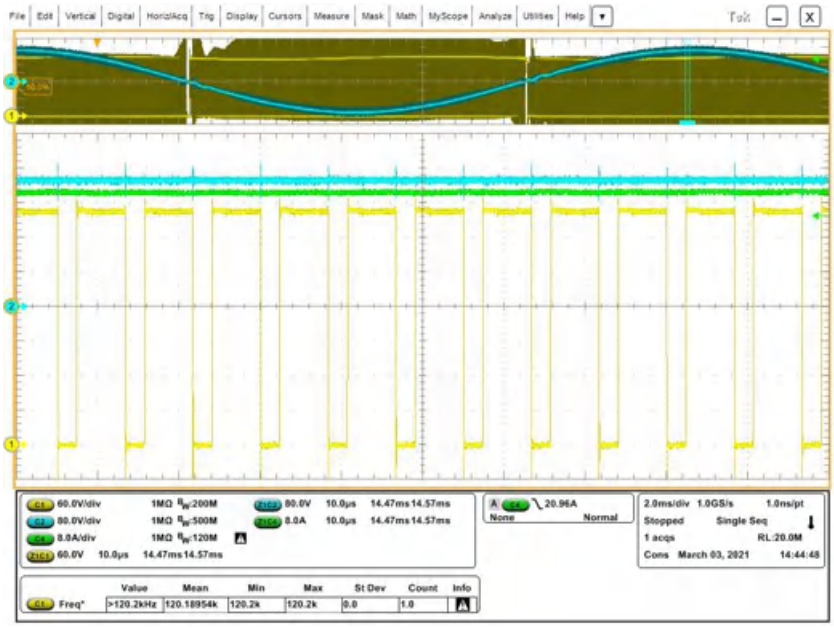


**Figure 5-54. PFC GaN Drain Voltage**

A zoom-in of the GaN switch drain-to-source voltage transition is shown to be approximately 20 ns in Figure 5-55. This rapid transition comes from the low $C_{OSS}$ of the LMG3522.

The waveform in Figure 5-55 was measured using the following parameters:

- Traces
  - C1: GaN Switch Node Drain Voltage
  - C2: $V_{IN}$
  - C4: $I_{IN}$
- Conditions
  - $V_{IN}$ = 208 V
  - $V_{OUT}$ = 400 V
  - $R_{OUT}$ = 43 Ω

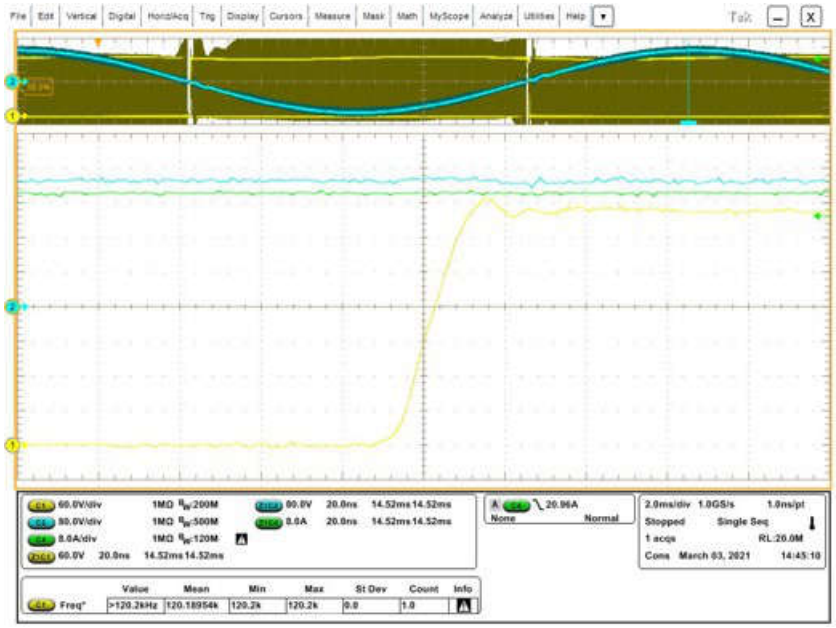


**Figure 5-55. PFC GaN Drain Voltage - Transition**

### 5.2.4.7 CLLLC Waveforms

Figure 5-56 shows CLLLC operation at 19 A (6.6 kW) under the following parameters:

- Traces
  - C1: GaN Primary Switch Node Drain Voltage
  - C2: GaN Secondary Switch Node Drain Voltage
  - C3: Transformer Primary Current
  - C4: Transformer Secondary Current
- Conditions
  - $V_{IN}$ = 400 V
  - $V_{OUT}$ = 350 V
  - $I_{OUT}$ = 19 A

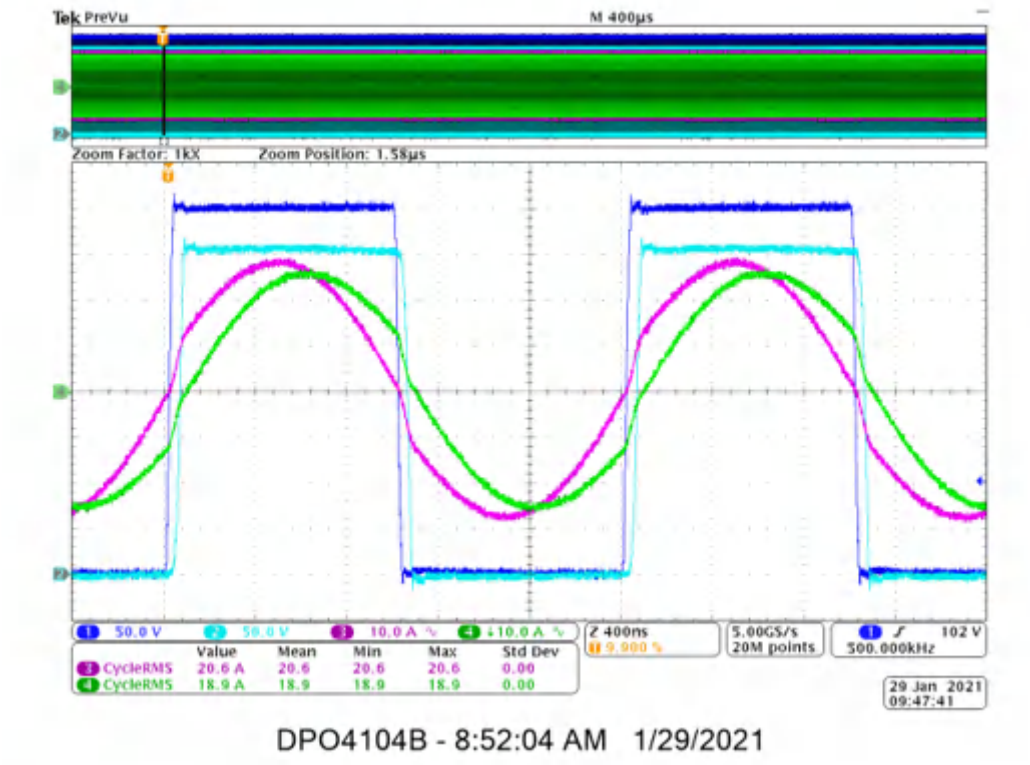



**Figure 5-56. CLLLC Operation 19 A (6.6 kW)**

Figure 5-57 shows CLLLC operation at 10 A and the following parameters:

- Traces
  - C1: GaN Switch Node Drain Voltage Leg 1
  - C2: GaN Switch Node Drain Voltage Leg 2
  - C3: Transformer Primary Current
  - C4: Transformer Secondary Current
- Conditions
  - $V_{IN}$ = 400 V
  - $V_{OUT}$ = 350 V
  - $I_{OUT}$ = 10 A

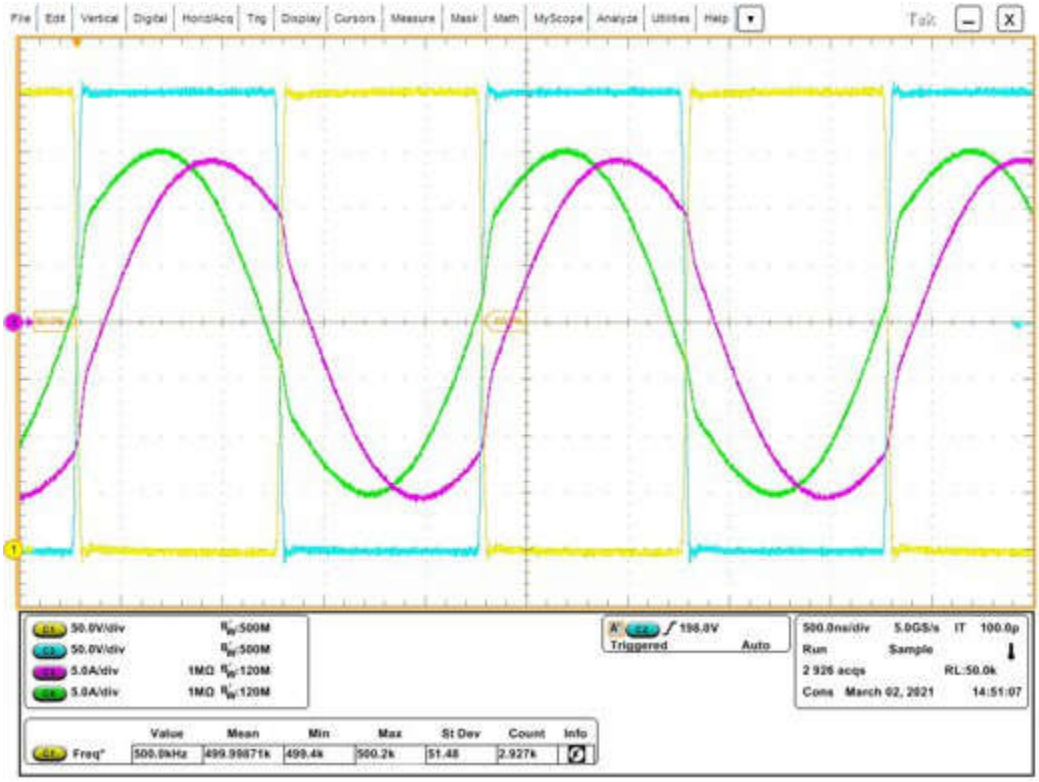



**Figure 5-57. CLLLC Operation 10 A**

A zoom-in of the GaN switch drain-to-source voltage transition is shown to be approximately 40 ns in Figure 5-58. This rapid transition comes from the low $C_{OSS}$ of the LMG3522.

The waveform in Figure 5-58 is measured using the following parameters:

- Traces
  - C1: GaN Switch Node Drain Voltage Leg 1
  - C2: GaN Switch Node Drain Voltage Leg 2
  - C3: Transformer Primary Current
  - C4: Transformer Secondary Current
- Conditions
  - $V_{IN}$ = 400 V
  - $V_{OUT}$ = 350 V
  - $I_{OUT}$ = 10 A

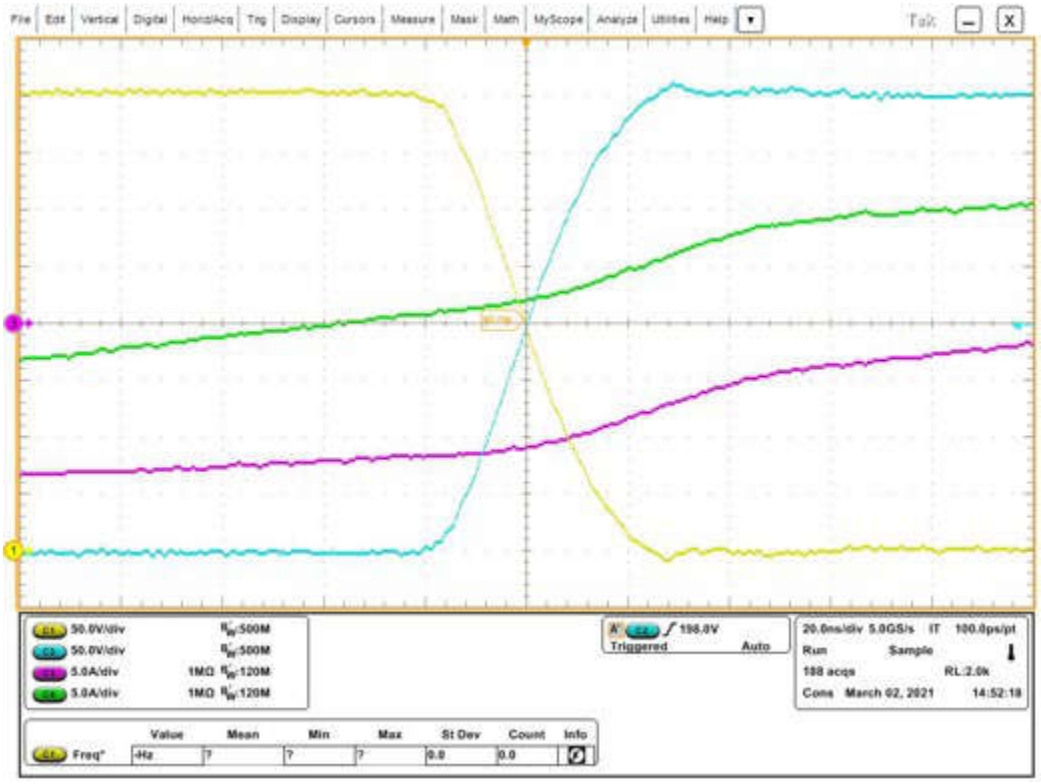



**Figure 5-58. CLLLC Operation 10 A - GaN FET Transitions**

The waveform in Figure 5-59 is measured using the following parameters:

- Traces
  - C1: GaN Switch Node Drain Voltage Leg 1
  - C2: GaN Switch Node Drain Voltage Leg 2
  - C3: Transformer Primary Current
  - C4: Transformer Secondary Current
- Conditions
  - $V_{IN}$ = 400 V
  - $V_{OUT}$ = 350 V
  - $I_{OUT}$ = 2 A

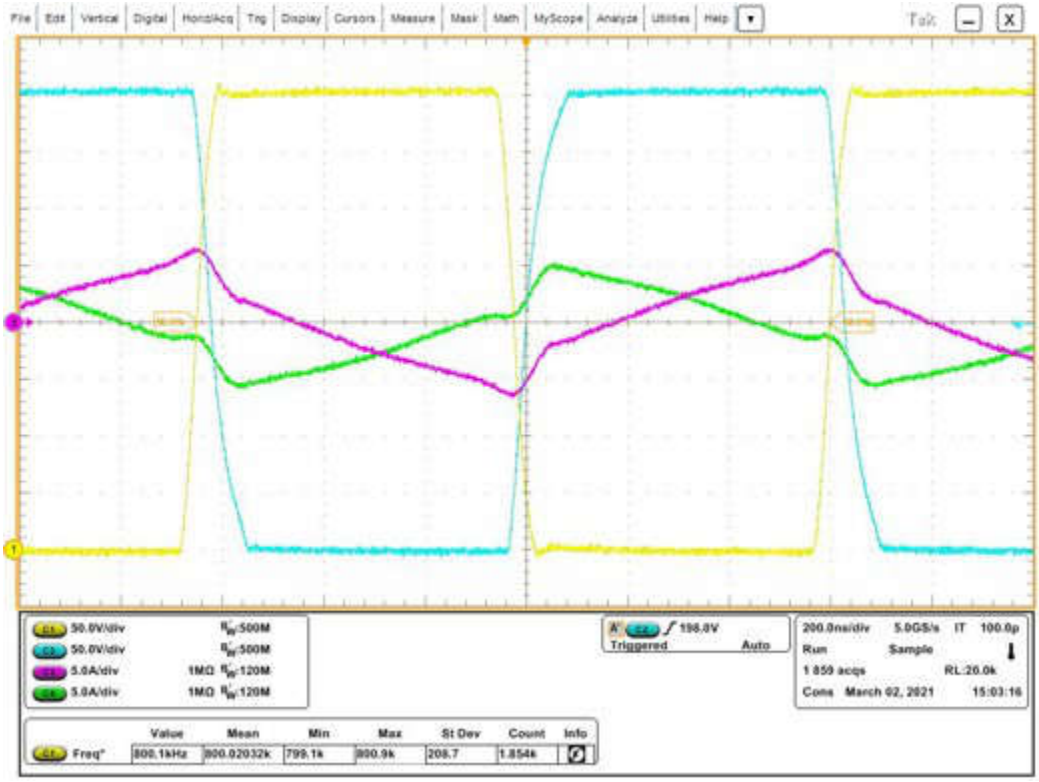



**Figure 5-59. CLLLC Operation 2 A**

A zoom-in of the GaN switch drain-to-source voltage transition is shown to be approximately 75 ns in Figure 5-60. This rapid transition comes from the low $C_{OSS}$ of the LMG3522. The slightly longer transition time in this image comes from the lighter load condition and the resulting reduced current flow.

The waveform in Figure 5-60 is measured using the following parameters:

- Traces
  - C1: GaN Switch Node Drain Voltage Leg 1
  - C2: GaN Switch Node Drain Voltage Leg 2
  - C3: Transformer Primary Current
  - C4: Transformer Secondary Current
- Conditions
  - $V_{IN}$ = 400 V
  - $V_{OUT}$ = 350 V
  - $I_{OUT}$ = 2 A

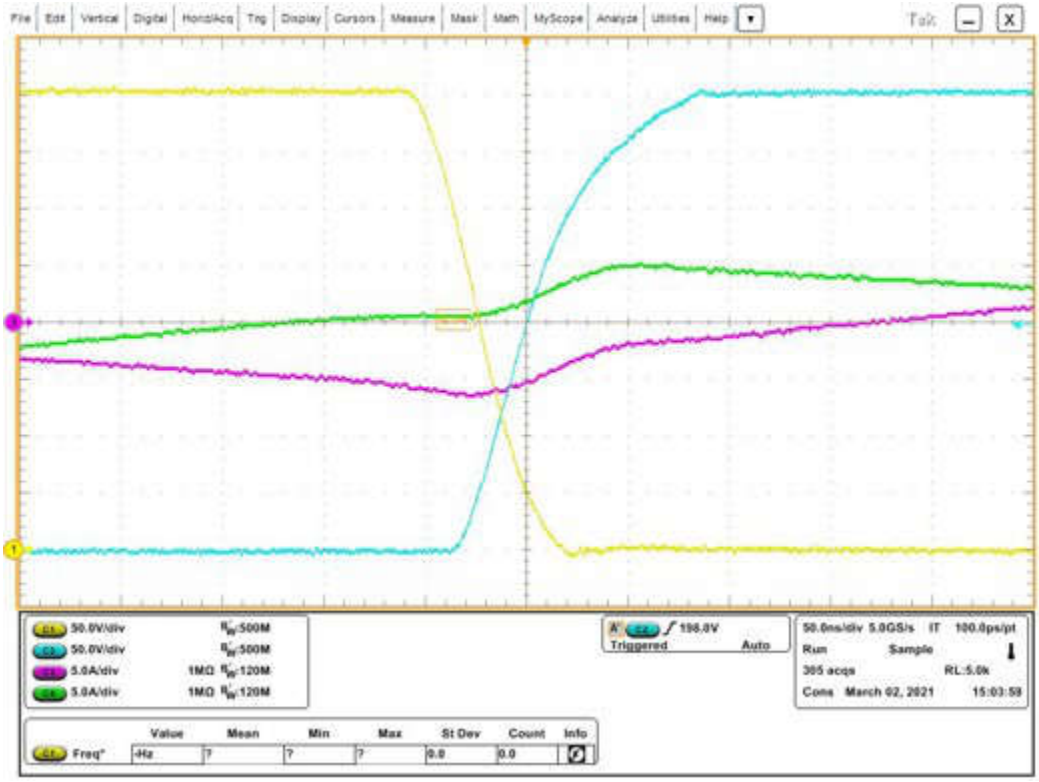



**Figure 5-60. CLLLC Operation 2 A - GaN FET Transitions**

# 6 Design Files

## 6.1 Schematics

To download the schematics, see the design files at TIDM-02013.

## 6.2 Bill of Materials

To download the bill of materials (BOM), see the design files at TIDM-02013.

## 6.3 Altium Project

To download the Altium Designer® project files, see the design files at TIDM-02013.

## 6.4 Gerber Files

To download the Gerber files, see the design files at TIDM-02013.

# 7 Software Files

To download the software for this reference design, go to the DigitalPower Software Development Kit (SDK) for C2000 MCUs site.

# 8 Related Documentation

1. Texas Instruments, *TMS320F28003x Real-Time Microcontrollers*, data sheet.
2. Texas Instruments, *C2000™ Software Frequency Response Analyzer (SFRA) Library and Compensation Designer in SDK Framework*, user's guide.
3. Zaka Ullah Zahid, Zakariya M. Dalala, Rui Chen, Baifeng Chen, and Jih-Sheng Lai, *Design of Bidirectional DC–DC Resonant Converter for Vehicle-to-Grid (V2G) Applications*, *IEEE Transactions on Transportation Electrification*, Vol. 1, No. 3, October 2015, pp. 232–244.
4. Zakariya M. Dalala, Zaka Ullah Zahid, Osama S. Saadeh, Jih-Sheng Lai, *Modeling and Controller Design of a Bidirectional Resonant Converter Battery Charger*, *IEEE Access*, Vol. 6, April 2018, pp. 23338–23350.
5. Biao Zhao, Qiang Song, Wenhua Liu, and Yandong Sun, *Overview of Dual-Active-Bridge Isolated Bidirectional DC–DC Converter for High-Frequency-Link Power-Conversion System*, *IEEE Transactions on Power Electronics*, Vol. 29, No. 8, August 2014, pp. 4091–4106.
6. Joel Turchi, J.T., Dhaval Dalal, D.D., Patrick Wang, P.T., and Laurent Lenck, L.L. (2014). *Power Factor Correction (PFC) Handbook: Choosing the Right Power Factor Controller Solution*. Revision 5, http://www.onsemi.com/pub/Collateral/HBD853-D.PDF
7. Texas Instruments, *Control Challenges in a Totem-Pole PFC*, analog applications journal.

## 8.1 Trademarks

# 9 Terminology

| ACRONYM | DEFINITION |
|---|---|
| BCM | Battery Charging Mode |
| BW | Bandwidth |
| CAN | Controller Area Network |
| CCM | Continuous Conduction Mode |
| CCS | Code Composer Studio |
| CLA | Control Law Accelerator |
| CLB | Configurable Logic Block |
| CLLLC | Capacitor Inductor Inductor Inductor Capacitor |
| CMPSS | Comparator Subsystem |
| CMTI | Common-Mode Transient Immunity |
| DAB | Dual Active Bridge |
| DAC | Digital-to-Analog Converter |
| DCM | Discontinuous Conduction Mode |
| DLOG | Data Logger |
| DT | Dead time |
| DUT | Design under test |
| eCAP | Enhanced Capture |
| ePWM | Enhanced Pulse Width Modulator |
| ERAD | Embedded Real-Time Analysis and Diagnostic |
| EV | Electric Vehicle |
| FET | Field Effect Transistor |
| FHA | First Harmonic Analysis |
| FSI | Fast Serial Interface |
| GaN | Gallium Nitride |
| HEV | Hybrid Electric Vehicle |
| HRPWM | High-Resolution Pulse Width Modulator |
| HSEC | High-Speed Edge Card |
| I2C | Inter-integrated Circuit |
| IDE | Integrated Development Environment |
| IGBT | Insulated-Gate Bipolar Transistor |
| ISR | Interrupt Service Routine |
| KCL | Kirchhoff's Current Law |
| KVL | Kirchhoff's Voltage Law |
| LIN | Local Interconnect Network |
| MOSFET | Metal Oxide Semiconductor Field Effect Transistor |
| OBC | Onboard Charger |
| PFC | Power Factor Correction |
| PGA | Programmable Gain Amplifier |
| PMBus | Power-Management Bus |
| SCI | Serial Communication Interface |
| SDFM | Sigma-Delta Filter Module |
| SFRA | Software Frequency Response Analyzer |
| SiC | Silicon Carbide |
| SPI | Serial Peripheral Interface |
| SRC | Series Resonant Converter |
| ZCS | Zero Current Switching |
| ZVS | Zero Voltage Switching |

## 10 About the Author

**Cody Watkins** is an application engineer with the C2000 Microcontrollers group at Texas Instruments. Cody received a degree in electrical engineering from the University of Cincinnati in 2015. Cody's interests are related to alternative energy sources, sustainability, and self-sufficient energy.

## 11 Revision History

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

| Changes from Revision * (October 2022) to Revision A (February 2024) | Page |
|---|---|
| • Software support with TMS320F28P65x microcontroller is added to this design | 1 |

# 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