Design Guide: TIDM-HV-1PH-DCAC
Grid Connected Inverter Reference Design

Description

This reference design implements single-phase inverter (DC/AC) control using a C2000™ microcontroller (MCU). The design supports two modes of operation for the inverter: a voltage source mode using an output LC filter, and a grid connected mode with an output LCL filter. High-efficiency, low THD, and intuitive software make this design attractive for engineers working on an inverter design for UPS and alternative energy applications such as PV inverters, grid storage, and micro grids. The hardware and software available with this reference design accelerate time to market.

Resources

TIDM-HV-1PH-DCAC	Design Folder
TIEVM-HV-1PH-DCAC	Orderable EVM Tool
TMS320F28377D	Product Folder
TMS320F280049C	Product Folder
AMC1304	Product Folder
OPA4350	Product Folder
UC3845	Product Folder

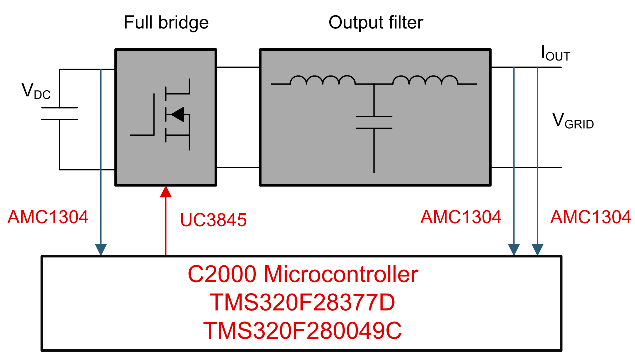
Search Our E2E™ support forums

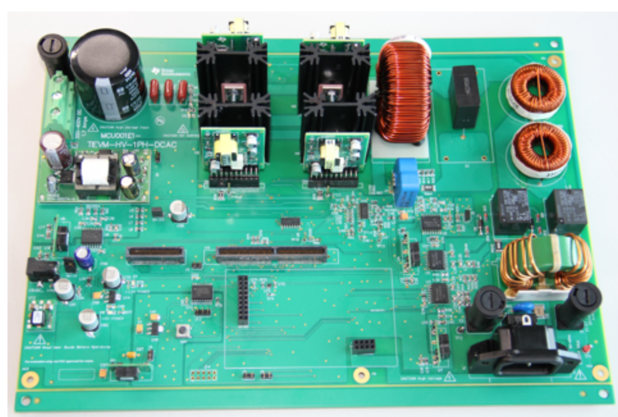
Features

- 380-DC V_{IN}, 110 V_{RMS}, 60 Hz, 400-VA Max Output, 20-kHz Switching
- Approximately 97% Efficiency
- <2% Total Harmonic Distortion at >50% Rated Power
- powerSUITE™ Support for Easy Adaptation of the Design for User Requirement
- SFRA and Compensation Designer for Ease of Tuning of Control Loops
- Supports TMS320F28377D and TMS320F280049C

Applications

- Photovoltaic Inverters
- Micro Grids
- Grid Storage
- Active Rectifier





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

1 System Description

WARNING

TI intends this 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 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 area 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.

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.

WARNING

Do not leave the design powered when unattended.

Grid connected inverters (GCI) are commonly used in applications such as photovoltaic inverters to generate a regulated AC current to feed into the grid. The control design of this type of inverter may be challenging as several algorithms are required to run the inverter. This reference design uses the C2000 microcontroller (MCU) family of devices to implement control of a grid connected inverter with output current control. A typical inverter comprises of a full bridge that is constructed with four switches that are modulated using pulse width modulation (PWM) and an output filter for the high-frequency switching of the bridge, as shown in Figure 1. An inductor capacitor (LCL) output filter is used on this reference design.

The design firmware is supported in the powerSUITE framework, which enables easy adaptation of the software and control design. All key algorithms such as phase locked loop (PLL) for grid synchronization and proportional resonant (PR) controllers provide good gain at selected frequencies. The adaptive notch filter actively dampens the resonance of the LCL filter that is implemented.

The high efficiency, low THD, and intuitive software of this reference design make it fast and easy to get started with the grid connected inverter design.

To regulate the output current, for example, the current feeds into the grid; voltages and currents must be sensed from the inverter. Sigma delta-based sensing provides easy isolation and superior sensing of these signals. Many C2000 MCUs have sigma-delta modulators to sense these parameters from the power stage. Sigma-delta modulators provide easy isolation and high quality reading of the physical variables, thus improving the overall quality of the control. Built-in sigma-delta demodulators on C2000 MCUs make using sigma delta-based sensing straight forward and easy to use.

Once the current and voltage parameters are sensed, the C2000 MCU runs the control algorithm to compute the modulation required for regulated operation. Compensation designer implements the model of the power stage, which makes the design of digital control loop simple. The software frequency response analyzer (SFRA) enables measurement of the frequency response in-circuit to verify the accuracy of the model and ensure accuracy of control.

1.1 Key System Specifications

Table 1 lists the key specifications of this reference design.

Table 1. Key System Specifications

PARAMETER	DESCRIPTION
Input voltage (V_{IN})	Typical 380-V DC, absolute max 400-V DC
Input current (I_{IN})	1.7 A max
Output voltage (V_{OUT})	Typical 110 V_{RMS}

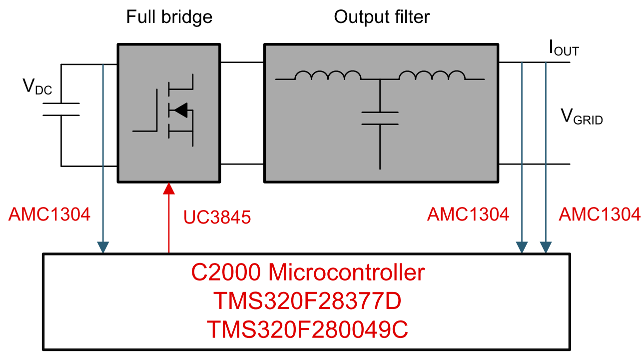
Table 1. Key System Specifications (continued)

PARAMETER	DESCRIPTION
Output current (I_{OUT})	Absolute RMS max 4.5 A, pulse max 10 A
VA rating	Absolute max 500 VA
THDi	<2% for greater than 50% rated load
Efficiency	At 110 V_{RMS} average is approximately 96%
Output inductor, Li	3 mH
Output capacitor, Cf	1 µF
Output grid side inductance, Lg	0.94 mH
Switching frequency	20 kHz

2 System Overview

2.1 Block Diagram

Figure 1. Typical Single Phase Inverter



2.2 System Design Theory

2.2.1 Modulation Scheme

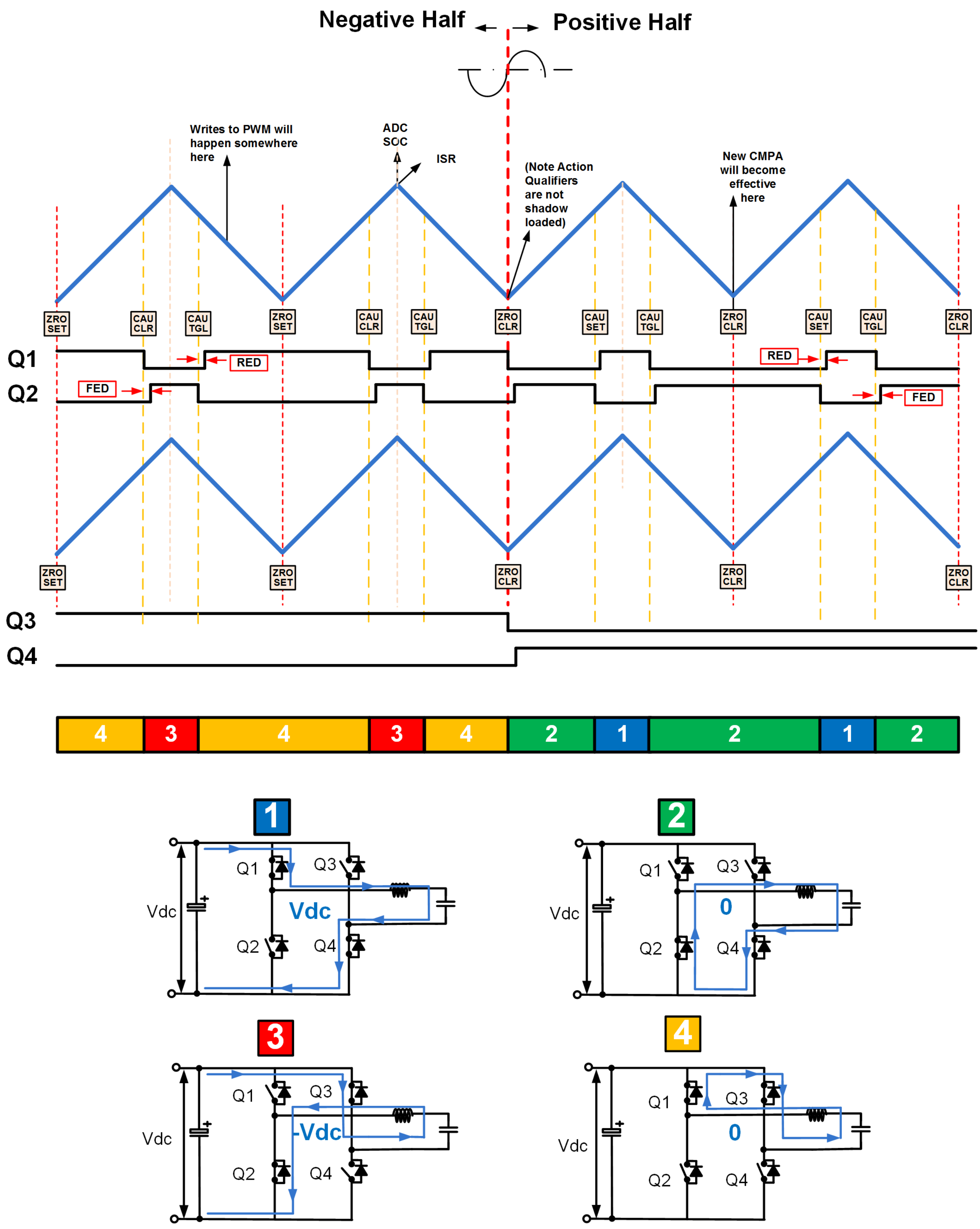
Popular modulation schemes for the PWM generation include bipolar modulation and unipolar modulation. This reference design uses a modified unipolar modulation in which switches Q1 and Q2 are switched at a high frequency and switches Q3 and Q4 are switched at a low frequency (frequency of the grid).

Table 2 lists the switching states of the inverter. The flexible PWM peripheral of the C2000 MCU enables generation of these signals easily. Figure 2 shows how the PWM peripheral is configured in this reference design. Ensure that the PWM waveform is symmetric around the zero crossing of the AC wave.

Table 2. Switching States Used in TIDM-HV-1PH-DCAC

CYCLE	Q1	Q2	Q3	Q4	VOLTAGE AT BRIDGE OUTPUT	STATE
Positive half cycle	ON	OFF	OFF	ON	V_{DC}	1
	OFF	ON	OFF	ON	0	2
Negative half cycle	OFF	ON	ON	OFF	$-V_{DC}$	3
	ON	OFF	ON	OFF	0	4

Figure 2. PWM Waveform Generation Using PWM Peripheral on C2000 MCU



2.2.2 Voltage and Current Sensing

To control the inverter stage for desired operation, voltage and current values are required to be sensed for processing by the digital controller. The design implements a sensing scheme based on ADCs and SDFMs. An Excel® sheet is also provided in the install package. In addition, this adapted solution's powerSUITE page can change the parameters of the sensing circuit. Observe how they effect the max sensed values.

For the grid connected mode, only the SDFM-based sensing is used in the software provided with the design.

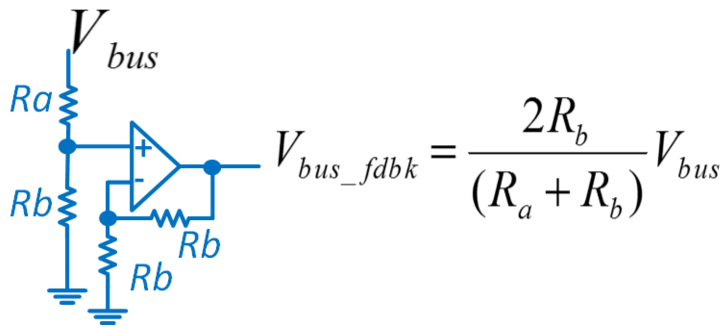
2.2.2.1 ADC-Based Sensing

In this reference design, the following signals are sensed using the on-chip ADC resource. The values shown here can also be entered through the powerSUITE configuration (CFG) page when ADC-based sensing is selected for the inverter.

2.2.2.1.1 DC Bus Sensing

The high-voltage DC bus is scaled down using a resistor divider. This resistor divider output can be directly fed into the ADC; however, this reference design uses an op amp stage to buffer this value as shown in Figure 3.

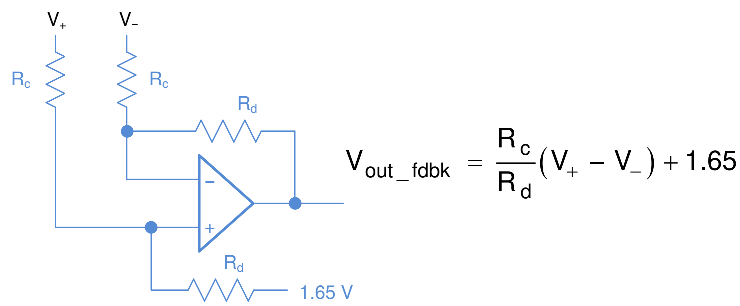
Figure 3. DC Bus Sensing Using Resistor Divider and Op Amp



2.2.2.1.2 AC Output Voltage Sensing

The AC output voltage is sensed differentially using resistor dividers and op amps, as shown in Figure 4. An offset voltage is added to the signal to enable measurement using the ADC, which can only convert positive voltages.

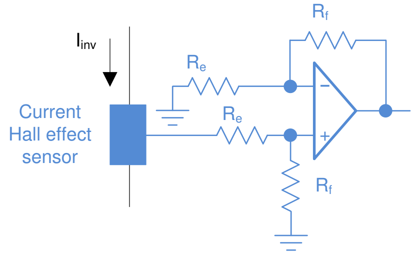
Figure 4. AC Output Voltage Differential Sensing Using Resistor Divider and Op Amp



2.2.2.1.3 Inductor Current Sensing

A Hall effect sensor is used to sense the current through the inductor. The Hall effect sensor has a built-in offset, and the range is different than what ADC can measure. As a result, the voltage is scaled to match the ADC range using the circuit shown in Figure 5.

Figure 5. Current Sense Using Hall Effect Sensor

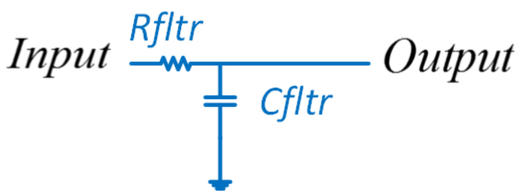


$$I_{inv\_fdbk} = \frac{R_f}{R_e}\left(I_{inv} \times \frac{V_{nominal}}{I_{nominal\_max}} + V_{offset}\right)$$

2.2.2.1.4 Sense Filter

An RC filter is used to filter the signals before being connected to the inverter. A common RC filter is used for all the sensing signals in this reference design, as shown in Figure 6.

Figure 6. RC Filter

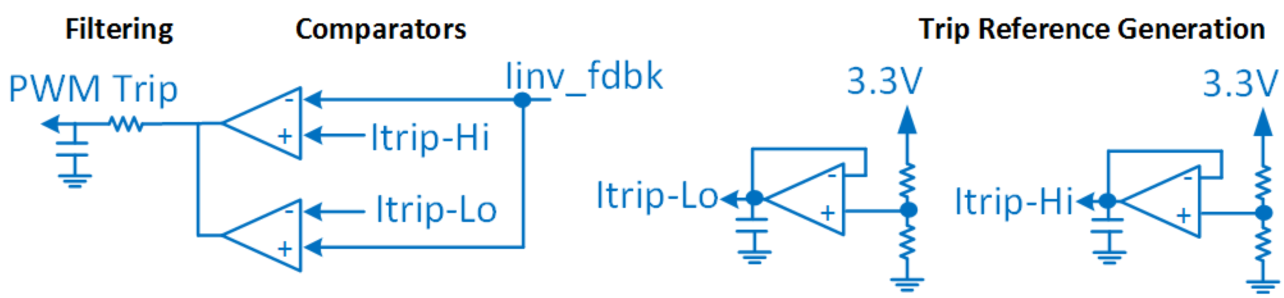


$$\frac{Output}{Input} = \frac{1}{1 + R_{fltr} C_{fltr} s}$$

2.2.2.1.5 Protection (Windowed Comparators)

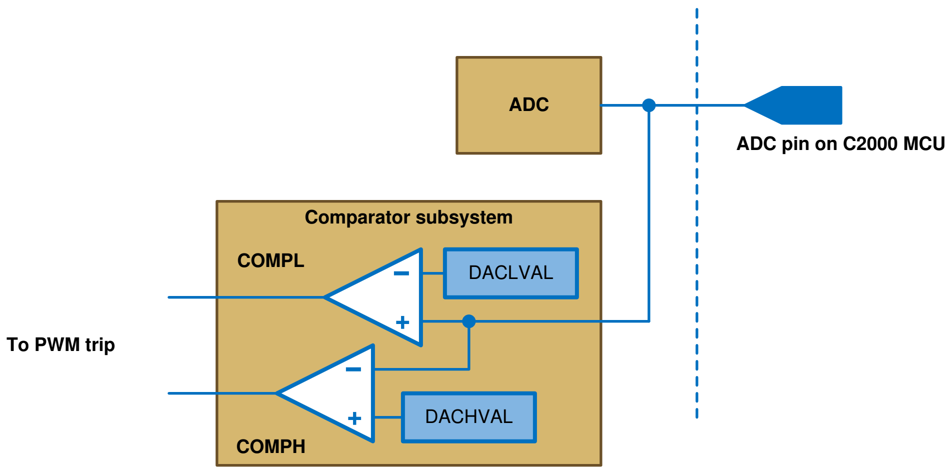
Most power electronics converters need protection from an overcurrent event. To protect from this event, multiple comparators are required, and references for the current and voltage trip must be generated, as shown in Figure 7.

Figure 7. Trip Generation for PWM Using Comparators and Reference Generators



All of this circuitry is avoided when using C2000 MCUs such as the TMS320F28377D, which has an on-chip windowed comparator internally connected to the PWM module that can enable fast tripping of the PWM. This connection saves board space, and cost in the end application as extra components can be avoid using on-chip resources. Figure 8 shows the comparator subsystem used for overcurrent protection.

Figure 8. Comparator Subsystem Used for Overcurrent Protection



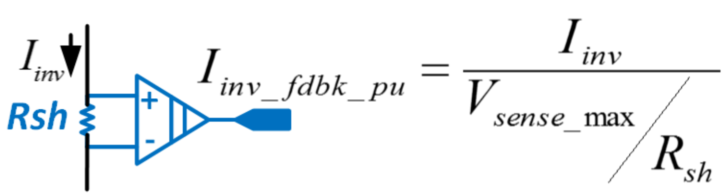
2.2.2.2 SDFM-Based Sensing

In this reference design, the following signals are sensed using the SDFM demodulator. The AMC1304 modulator generates the sigma-delta stream. The clock for the modulator is generated from the ECAP peripheral on the C2000 MCU. The AMC1304 modulator senses the signal in an isolated fashion and is very useful when designing inverters in which the controller needs to be on the isolated and cold side. The values shown here can also be entered through the powerSUITE page when SDFM-based sensing is selected for the inverter.

2.2.2.2.1 Isolated Output Current and Capacitor Current Sensing

A shunt resistor is used to sense the capacitor current and the output current on this reference design. The voltage across the shunt resistor is fed into the AMC1304 sigma-delta modulator, which generates the sigma-delta stream that is decoded by the SDFM demodulator present on the C2000 MCU, as shown in Figure 9. The inductor current is deuced from the capacitor and the output current readings.

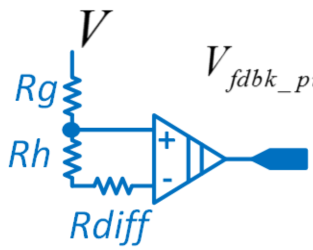
Figure 9. SDFM-Based Isolated Current Sensing Using a Shunt Resistor



2.2.2.2.2 Isolated Output Voltage and DC Bus Sensing

A resistor divide network senses the DC bus and output voltage using the SDFM. Account for the differential input resistance of the SDFM when interpreting the demodulated signals, as shown in Figure 10.

Figure 10. SDFM-Based Isolated Voltage Sensing for DC Bus and Output Voltage



$$V_{fdbk\_pu} = \frac{V}{V_{sense\_\max} \Big/ \dfrac{(R_h \parallel R_{diff})}{R_g + (R_h \parallel R_{diff})}}$$

2.2.2.2.3 Protection

In addition to the data filter, which can demodulate the SDFM stream generated by the modulator with specified oversampling rate (OSR) and filter order (SINC1, SINC2, SINC3), the SDFM has additional comparator filters that can be programmed with a much lower OSR and filter order to enable fast trip of the PWM.

2.2.2.2.4 SDFM Clock Generation

ECAP module generates the clock for the AMC1304 modulator. This clock is routed outside from the ECAP module using the OutputXbar and then routed back in to the SDFM CLK pins on the device. For details on using SDFM and CLK pins in this reference design, see Table 2.

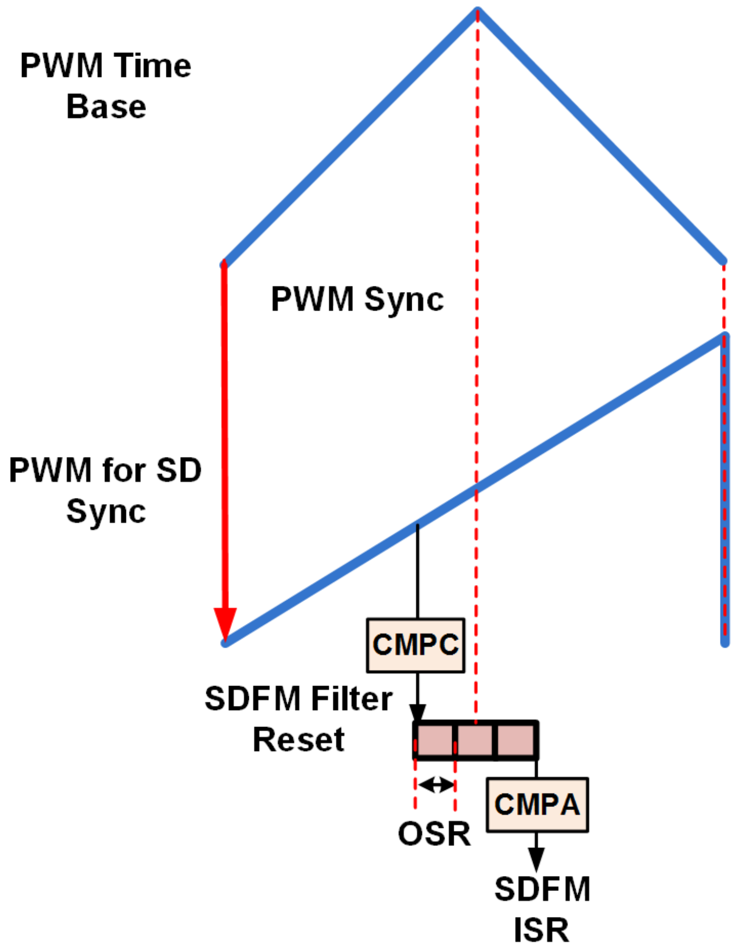
2.2.2.2.5 SDFM Filter Reset Generation and Syncing to Inverter PWM

SDFM provides a continuous stream of data. This data is then demodulated by the C2000 SDFM peripheral. Most control applications require the sampling of the data to be centered deterministically around the switching waveform (that is, the controlling PWM). The C2000 MCU provides a mechanism to generate this sync signal to the SDFM demodulator. The exact mechanism of the sync can be different on different devices. The following sections discuss the sync mechanism in several C2000 devices.

2.2.2.2.6 F2837x and F2807x

On these devices, the PWM11 is tied to the SDFM reset generation, hence the sync generation involves propagation of the sync from the inverter PWM to the PWM11 module. As the SDFM data is only valid 3 OSR time periods after the sync is provided, determine the time at which SDFM data must be read. Figure 11 shows the SDFM filter reset being generated from the PWM module and the ISR trigger to read the SDF registers.

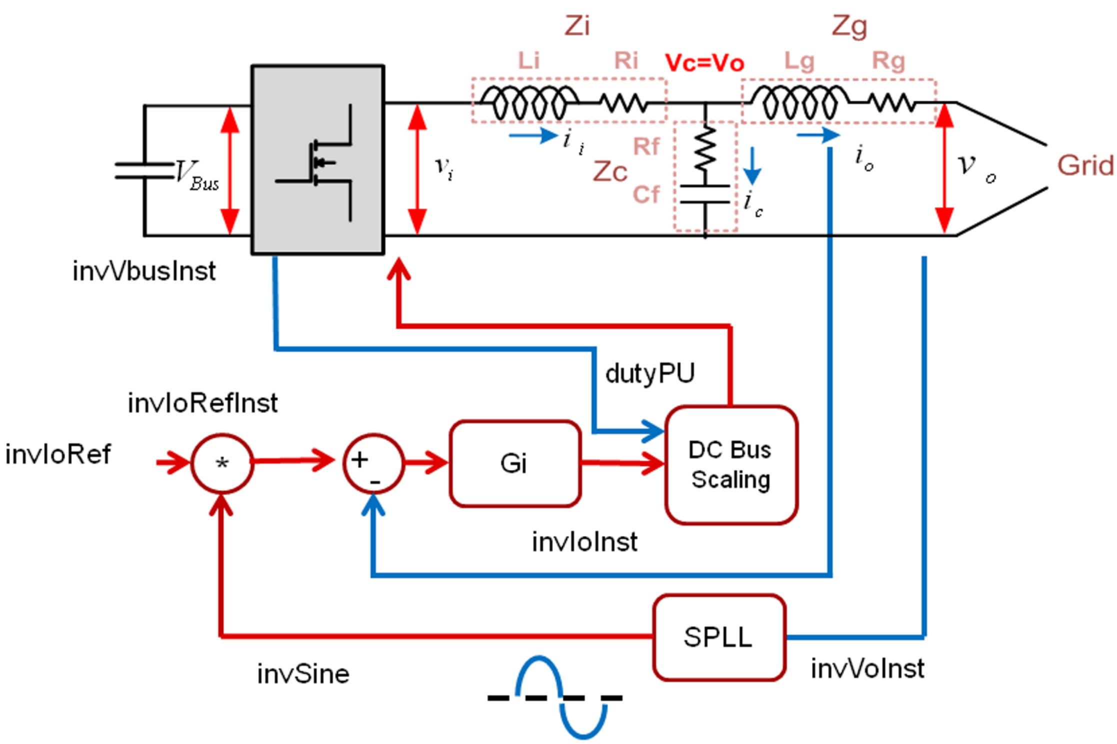
Figure 11. SDFM Filter Reset Being Generated From the PWM Module and ISR Trigger to Read SDF Registers



2.2.3 Control Scheme

The current control scheme is used for the grid inverter, as shown in Figure 12.

Figure 12. Control Scheme Used for Grid Connected Inverter Control



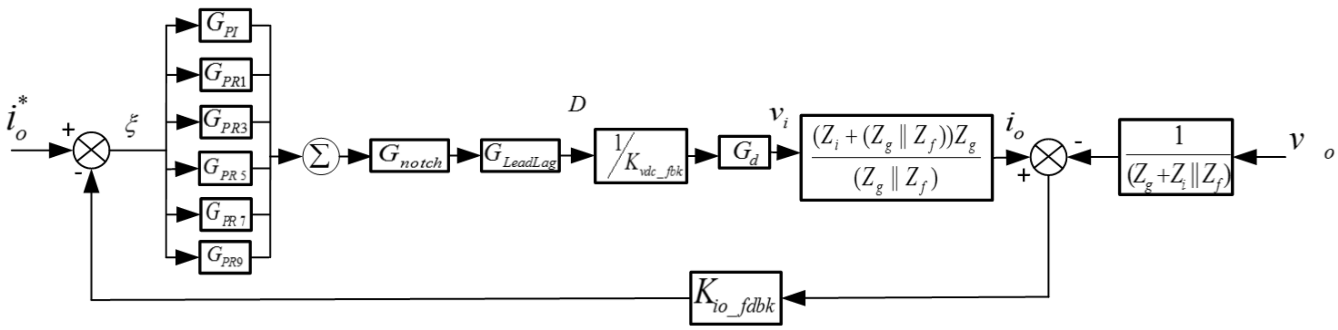
First, the grid voltage, V_O, is sensed in the variable *invVoInst*. A software PLL algorithm is then run to compute the angle and phase of the grid, *invSine*. This *invSine* value is then multiplied with the reference current command *invIoRef*, which generates the instantaneous current command reference *invIoRefInst*. This reference is then compared with the sensed output current, *invIoInst*, and the error fed into the current compensator, G_i, as shown in Equation 1. The goal of the current compensator is to zero the error between the reference and the measured value. A typical proportional integral (PI) controller can zero the error for the DC value; however, for a sinusoidal reference, the controller cannot reduce the error to zero. Thus, proportional resonant (PR) controllers are used as part of the current compensator G_i to zero the error at the AC frequency.

Additional resonant controllers are added to the current compensator to zero the error at harmonic frequency of the fundamental frequencies that are generated. A lead lag compensator is added to the current compensator to improve the phase margin in the design, and a PI controller is added to reduce startup current. Equation 1 shows the current compensator G_i.

$$G_i = \left(K_p \frac{(s+z0)}{s} + K_{pI\_1H} + \sum_{n=1}^{N} \frac{K_{iI\_nH}\, 2\,\omega_{rcI\_nH}\, s}{s^2 + 2\,\omega_{rcI\_nH}\, s + \omega_{o\_nH}^2} \right) G_{Lead\_Lag} \tag{1}$$

Figure 13 shows that the grid voltage acts as a disturbance in the system, which can be modeled as an impedance with regards to the current reference. The harmonic PR controllers help by increasing the impedance at harmonic frequencies, thus reducing distortion in the grid feed current where N is the total number of harmonic compensators added in the control loop. This reference design uses a total of five resonant compensators that compensate the first, third, fifth, seventh, and ninth harmonic. The compensation designer models the current loop plant and enables tuning of the compensator coefficients through the powerSUITE page.

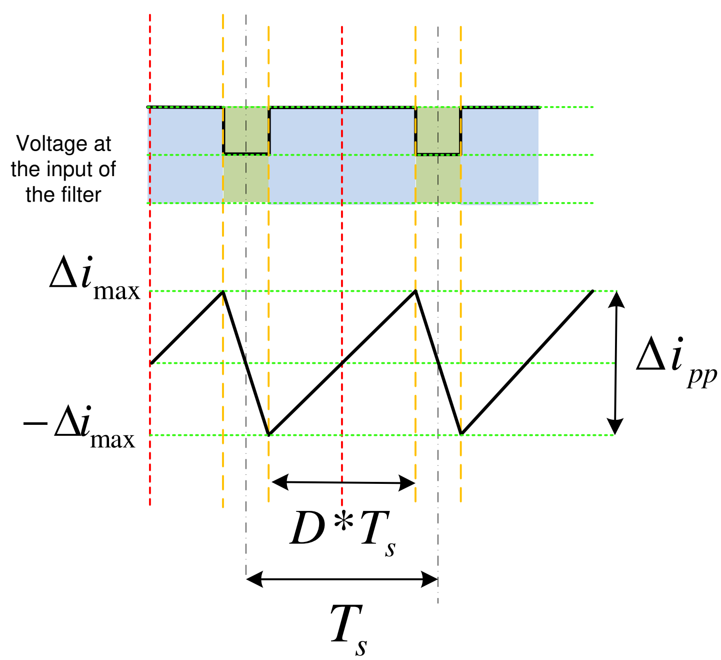
Figure 13. Control Diagram of Grid Connected Inverter



2.2.4 Inductor Design

The primary role of the inductor (Li) in the output filter is to filter out the switching frequency harmonics. Amongst other factors, the design of the inductor design depends calculating the current ripple and choosing a material for the core that can tolerate the calculated current ripple. Figure 14 shows one switching cycle waveform of the inverter output voltage V_i with regards to inductor current.

Figure 14. Current Ripple Calculation



The voltage across the inductor is given by:

$$V = L_i \times \frac{di}{dt} \tag{2}$$

For the full bridge inverter with an AC output, write the equation as:

$$=> (V_{Bus} - V_O) = L_i \times \frac{\Delta i_{pp}}{D \times T_s} \tag{3}$$

Where $T_s = \frac{1}{F_{sw}}$ is the switching period. Now, rearrange the current ripple at any instant in the AC waveform, given as:

$$=> \Delta i_{pp} = \frac{D \times T_s \times (V_{Bus} - V_O)}{L_i} \tag{4}$$

Assuming the modulation index to be m_a, the duty cycle is given as:

$$D(\omega t) = m_a \times \sin(\omega t) \tag{5}$$

The output of the inverter must match the AC voltage as it is safe to assume:

$$V_O = V_{DC} \times D \tag{6}$$

Therefore,

$$\Delta i_{pp} = \frac{V_{Bus} \times T_s \times m_a \times \sin(\omega t) \times (1 - m_a \sin(\omega t))}{L_i} \tag{7}$$

As seen in Equation 7, the peak ripple is a factor of where the inverter is in the sinusoidal waveform (for example, the modulation index). To find the modulation index where the maximum ripple is present, differentiate Equation 7 with regards to time to get Equation 8, and equate to zero.

$$\frac{d(\Delta i_{pp})}{dt} = K\{\cos(\omega t)(1 - m_a \sin(\omega t)) - m_a \sin(\omega t) * \cos(\omega t)\} = 0 \tag{8}$$

$$=> \sin(\omega t) = \frac{1}{2\, m_a} \tag{9}$$

Equation 9 then gives the modulation index for which the ripple is maximum, substituting back in Equation 7. The inductance value required to tolerate the ripple is shown in Equation 10 and Equation 11:

$$\Delta i_{pp}\Big|_{max} = \frac{V_{Bus} \times T_s}{4 \times L_i} \tag{10}$$

$$L_i = \frac{V_{Bus}}{4 \times F_{sw} \times \Delta i_{pp}\Big|_{max}} \tag{11}$$

For this reference design, the rating is 600 VA, the switching frequency is 20 kHz, and the bus voltage is 380 V. Assume that the ripple is 20% and is tolerable by the inductor core, and the minimum inductance required is calculated as:

$$L = \frac{380}{4 \times 20000 \times 5.45 \times 1.414 \times 0.20} = 3.08 \text{ mH} \tag{12}$$

These calculations are also provided inside an Excel sheet form for convenience located at: *C:\ti\c2000\C2000Ware_DigitalPower_SDK_<version>\solutions\tidm_hv_1ph_dcac\hardware\baseboard\calculation.xlsx sheet → UPS Li & Cf Sel*.

Select an appropriate core with these values in mind, and the inductor is designed to meet the inductance value.

$$L_i = \frac{V_{DC}}{4 \times F_{sw} \times \Delta i_{pp}\Big|_{max}} \tag{13}$$

$$L = \frac{380}{4 \times 20000 \times 5.45 \times 1.414 \times 0.20} = 3.08 \text{ mH} \tag{14}$$

For this reference design, the rating is 600 VA, the switching frequency is 20 kHz, and the bus voltage is 380 V. Assuming a 20% ripple is tolerable by the inductor core, the minimum inductance needed is calculated as:

<sdk_install_path>\solutions\tidm_hv_1ph_dcac\hardware\baseboard\calculation.xlsx sheet

sheet→Grid Conn. Li, cf, Lg Sel

sheet→LI Design

Select an appropriate core with these values in mind, and the inductor is designed to meet this inductance value.

2.2.5 Capacitance and Grid Side Inductance Selection

The output inductor and capacitor form a low-pass filter that filters out the switching frequency. As the inverter is connected to the grid, the capacitance determines the VAR power exchange when the inverter is not operating and is kept small, typically < 5% rated power. In addition, choose the grid side inductance such that the LCL resonance is greater than $F_{SW}/6$. See the following Excel sheet for a detailed calculation. The total switching attenuation is another aspect that determines the selection of the components.

<sdk_install_path>\solutions\tidm_hv_1ph_dcac\hardware\baseboard\calculation.xlsx sheet

sheet→Grid Conn. Li, Cf, Lg, Sel

3 Hardware, Firmware, Testing Requirements, and Test Results

3.1 Required Hardware and Firmware

3.1.1 Hardware

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

NOTE: This reference design is also available for order as TIEVM-HV-1PH-DCAC. Note the 15-V DC, 15-W power supply is not shipped with the design and must be arranged for by the user. A two-pronged power supply is recommended so it is truly floating and isolated. Cables, loads, oscilloscopes, and current probes must be arranged for by the user and connected to this EVM according to the user guide instructions and observing local compliance and standards for wiring. Only use isolated power supplies.

Also, the shipped EVM is configured in voltage source mode and the user will need to depopulate the C1 20-µF capacitor and populate it with a C1 1-µF capacitor, which is provided in the EVM box. In addition to this, the L2 and L2N, which are jumper wired on the voltage source inverter, are populated in this reference design. L2 and L2N are also provided in the EVM box but need to be soldered on by the user.

3.1.1.1 Base Board Settings

The design follows a HSEC control card concept and any device for which HSEC control card is available from the C2000 MCU product family can be used on the design. Table 3 lists the key resources used for controlling the power stage on the MCU. Figure 15 shows the key power stage and connectors on the reference design, and Table 4 lists the key connectors and their functions. To get started:

1. Make sure no power source is connected to the reference design.
2. Ensure that the output filter is correct for the mode that is desired to run the design. For example, for the grid connected mode, an LCL filter is used. L2 and L2N must be populated with the 470-mH inductor; this inductor is provided in the EVM box, and the part number can also be identified from the BOM. The BOM is for voltage source inverter; the L2 and L2N are listed as DNP, but the part number is provided. The capacitor C1 must be populated with the 1-µF film capacitor (250-V AC, 630-V DC, Polypropylene (PP), Metallized Radial 1.240" L × 0.532" W), which is also provided in the EVM box. Figure 15 shows the board picture when configured in grid connected mode.
3. Insert the control card in the J15-J16 slot.
4. Connect a 15-V DC, 1-A power supply at J2.
5. Insert a jumper at J4 if not already populated. The LED lights on the base board and control card will light up, indicating that the device is powered up.
6. Connect a USB cable from the control card to a host computer to connect JTAG.
7. Optional: Connect an isolated high-voltage DC source to the J17, but do not apply power at this point.
8. Connect a resistive load of approximately 100 Ω to the output from J1.

Figure 15. Board Overview

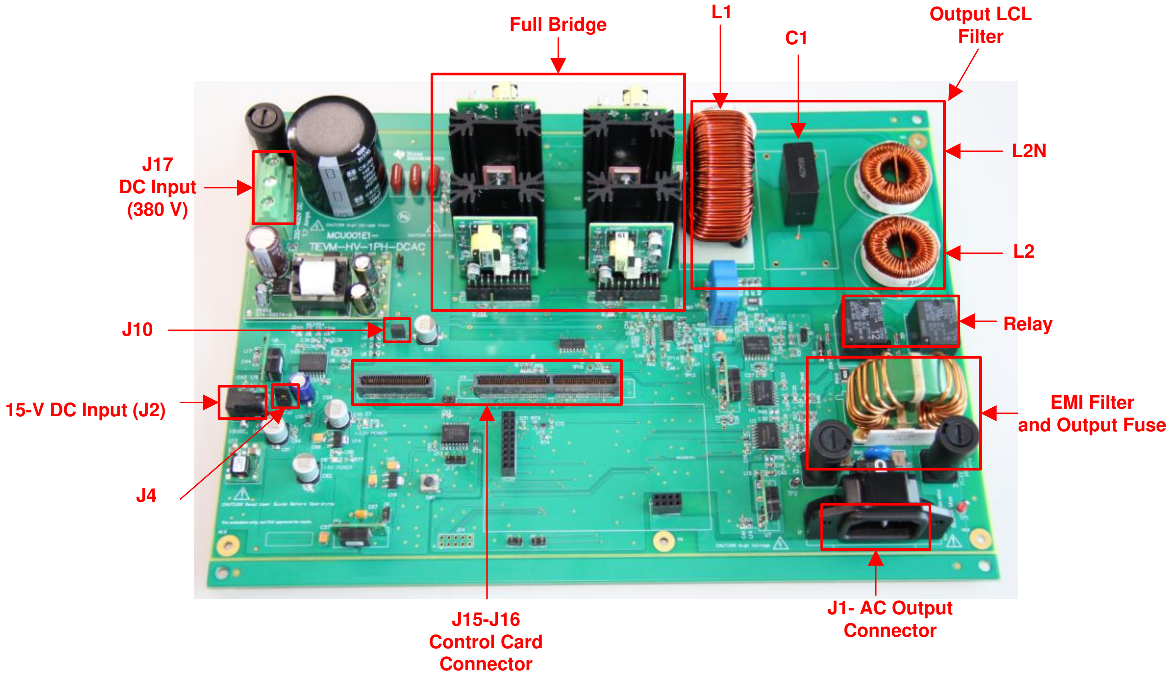


Table 3. Key Controller Peripherals Used to Control Bridge on Board

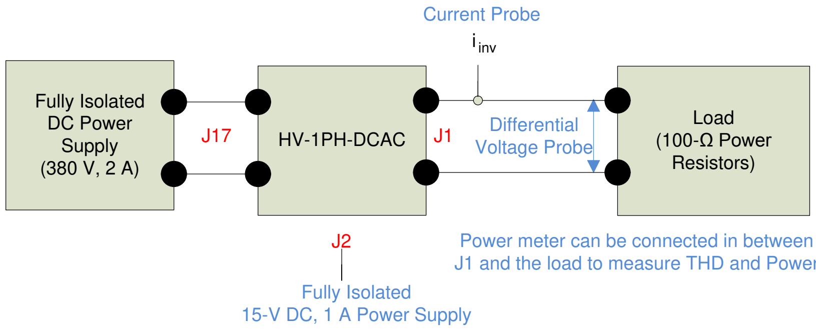
SIGNAL NAME	HSEC PIN NUMBER	FUNCTION
PWM–1A	49	PWM: Inverter drive
PWM–1B	51	PWM: Inverter drive
PWM–2A	53	PWM: Inverter drive
PWM–2B	55	PWM: Inverter drive
I.inv	15	ADC: Inductor current measurement
1.65 V	17	ADC: Reference voltage generated on the board
Bus.V	21	ADC: DC bus sensed on the board
Line.V	25	ADC: AC voltage sensing
PLC_RX	27, 12	ADC: PLC ADC pin
SD_Data_CapI	99	SDFM: Data from the SDFM modulator for the capacitor current feedback
SD_Data_GridI	103	SDFM: Data from the SDFM modulator for output current
SD_CLK_GridV, SD_CLK_GridI, SD_CLK_CapI	50, 101, 105, 109	SDFM: Clock from the SDFM Modulator (common clock is used for grid voltage, current and capacitor current SDFM) . This clock is generated from ECAP1 module which is brought out using the *Output XBar*
SD_CLK_Vbus	102, 54	SDFM: Clock from the SDFM modulator used for Vbus measurement. This clock is generated from ECAP1 module which is brought out using the *Output XBar*
OPRLY	52	GPIO: Relay GPIO output
SW–ON	56	GPIO: Switch GPIO input

Table 4. Key Connectors and Their Function

CONNECTOR NAME	FUNCTION
J17	Used to connect the high-voltage DC bus at the input
J2	Supplies the bias power supply for the control card and the circuitry for sensing on the base board
J4	Switch to connect disconnect the DC bias of the board
J1	AC connector to connect the output to load
J15-J16	HSEC control card slot
J10	Supplies the DC bias power supply to the isolated gate drivers; must be populated

Figure 16 shows the hardware setup to run software for Build Level 1.

Figure 16. Hardware Setup to Run Software for BUILD Level 1



3.1.1.2 Control Card Settings

Certain settings on the device control card are needed to communicate over JTAG, use the isolated UART port, and provide a correct ADC reference voltage. Follow these steps on revision 1.1 of the TMS320F28377D control card. Refer to the info sheet located inside C2000Ware at *<sdk_install_path>\c2000ware\boards\controlcards\TMDSCNCD28377D* .

1. Set both ends of A:SW1 on the control card to the *ON* (up) position to enable the JTAG connection to the device and the UART connection for the SFRA GUI. If this switch is *OFF* (down), the user cannot use the isolated JTAG built in on the control card, nor can the SFRA GUI communicate with the device.
2. Connect the USB cable to A:J1 to communicate with the device from a host PC on which Code Composer Studio™ (CCS) runs.
3. For the control loop, set the appropriate jumpers to provide a 3.3-V reference externally to the on-chip ADC.

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 settings are required for revision A of the TMS320F280049C control card. Refer to the info sheet located at *<sdk_install_path>\c2000ware\boards\controlcards\TMDSCNCD280049C*.

1. Set both ends of S1:A on the control card to the *ON* (up) position to enable JTAG connection to the device and UART connection for SFRA GUI. If this switch is *OFF* (down), the user cannot use the isolated JTAG built in on the control card, nor can the SFRA GUI communicate with the device.
2. Connect the USB cable to J1:A to communicate with the device from a host PC on which CCS runs.
3. A 3.3-V reference is desired for the control loop tuning on this design. Internal reference of the TMS320F28004x is used and for this S8 switch must be moved to the left (that is, pointing to VREFHI).
4. For the best performance of this reference design, remove the capacitor connected between the isolated grounds on the control card, C26:A.
5. GPIO24 through GPIO27 are muxed on the TMS320F280049C control card. To route them to the correct control card pins for the SDFM, flip all the switches on SW5 to *OFF* (down) and all the switches on SW6 to *ON* (up).

3.1.1.3 Tips to Connect JTAG USB Cable

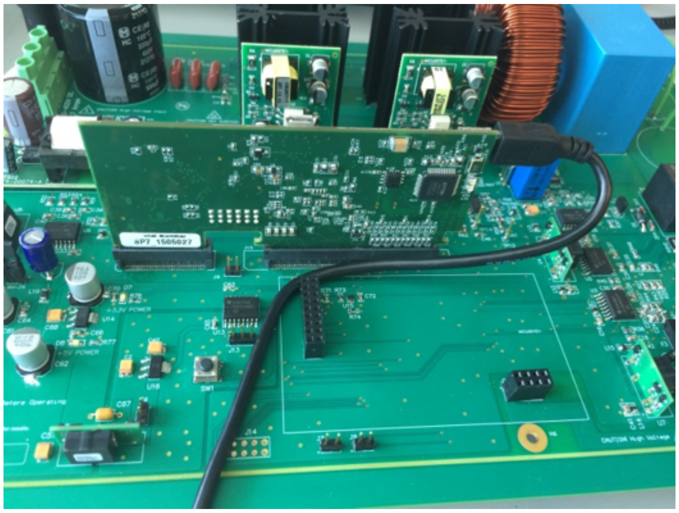
High-voltage boards can generate high EMI due to switching action. Even though the JTAG is isolated, some coupling can still occur due to radiated EMI. This coupling can result in a loss of JTAG frequently. Follow these suggestions to avoid this from happening:

1. Wind the USB cable around a ferrite bead as shown in Figure 17.

Figure 17. USB Cable Around Ferrite Bead

2. Do not cross the USB cable directly over the high-voltage section by the following connection of the USB cable.

Figure 18. USB Connection on Board



3.1.2 Firmware: powerSUITE and Incremental Build Software

NOTE: The firmware for this reference design is supported on both the TMS320F283779D and TMS320F280049C devices.

3.1.2.1 Opening the Project Inside Code Composer Studio™

To start:

1. Install CCS (version 9.!~3!~2.0 or above).
2. Install C2000Ware DigitalPower SDK.

NOTE: powerSUITE is installed with DigitalPower SDK in the default install.

---

3. Open CCS and create a new workspace.
4. Go to *View → Resource Explorer*.
5. Under the TI Resource Explorer, go to *Software → C2000Ware DigitalPower SDK - <version>*.

The software of this design can be opened in two modes.

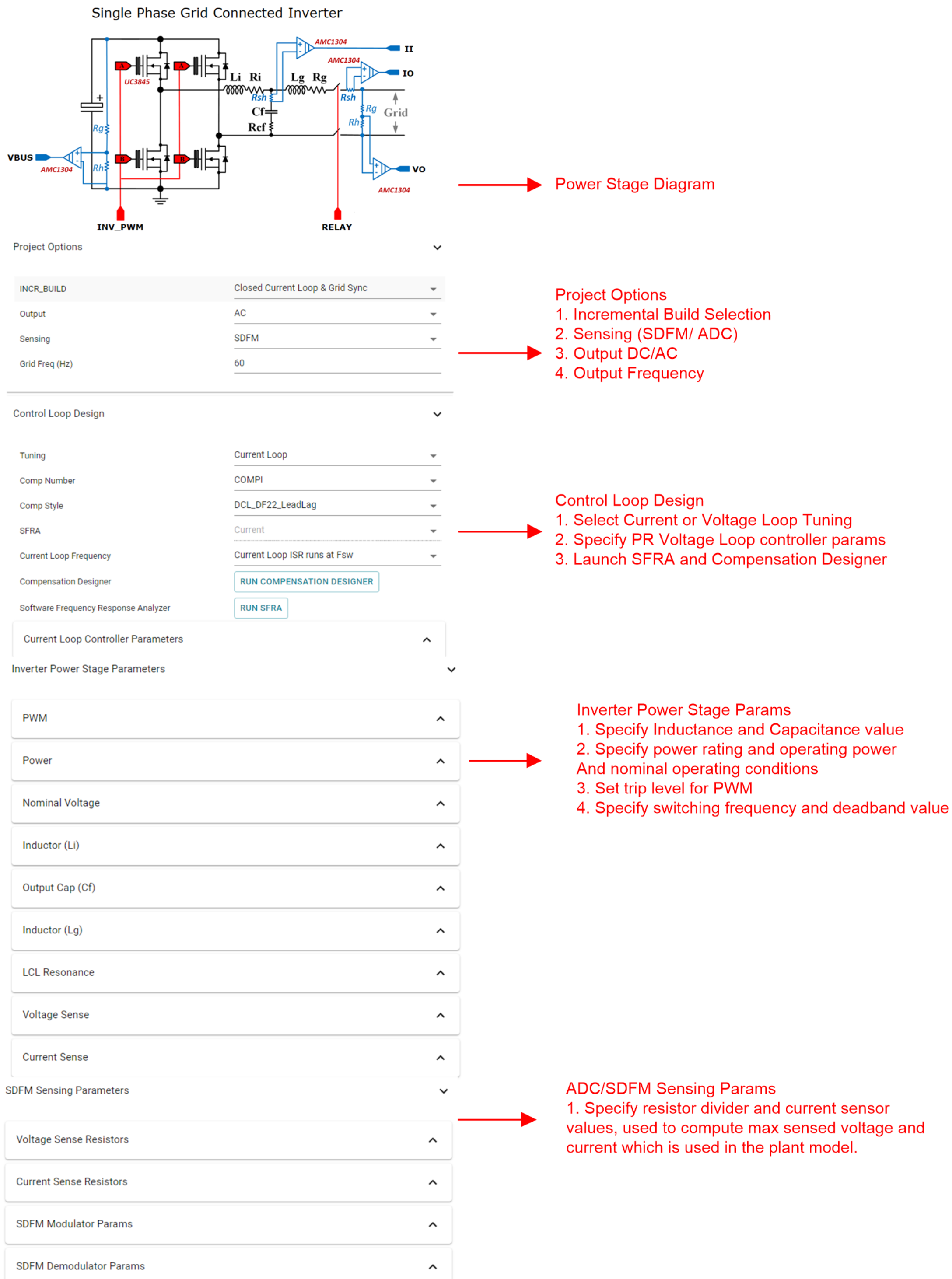
3.1.2.1.1 Open TI Design Software for Adaptation

The software opens the firmware as it was run on this design and hardware. The user can modify power stage parameters used to create the model of the power stage in the compensation designer. The user can also modify scaling values for voltages and currents.

1. Under C2000Ware DigitalPower SDK, select *Solution Adapter Tool → Single Phase Inverter: Grid Connected Inverter*.
2. The development kit and designs page appears. This page to browse all of the information on the design, including this design guide, test reports, hardware design files, and more.
3. Click *Import <devicename> Project*.
4. The project is imported into the workspace environment. A CFG page with a GUI similar to Figure 19 appears.
5. If necessary, use the GUI to change the parameters for an adapted solution, such as power rating, inductance, capacitance, sensing circuit parameters, and more.

Figure 19 shows the powerSUITE page for the grid connected inverter solution.

Figure 19. powerSUITE Page for the Grid Connected Inverter Solution



3.1.2.2 Project Structure

Once the project is imported, the project explorer appears inside CCS, as shown in Figure 20.

NOTE: Figure 20 shows the project for F28377D; however, if a different device is chosen from the powerSUITE page, the structure will be similar.

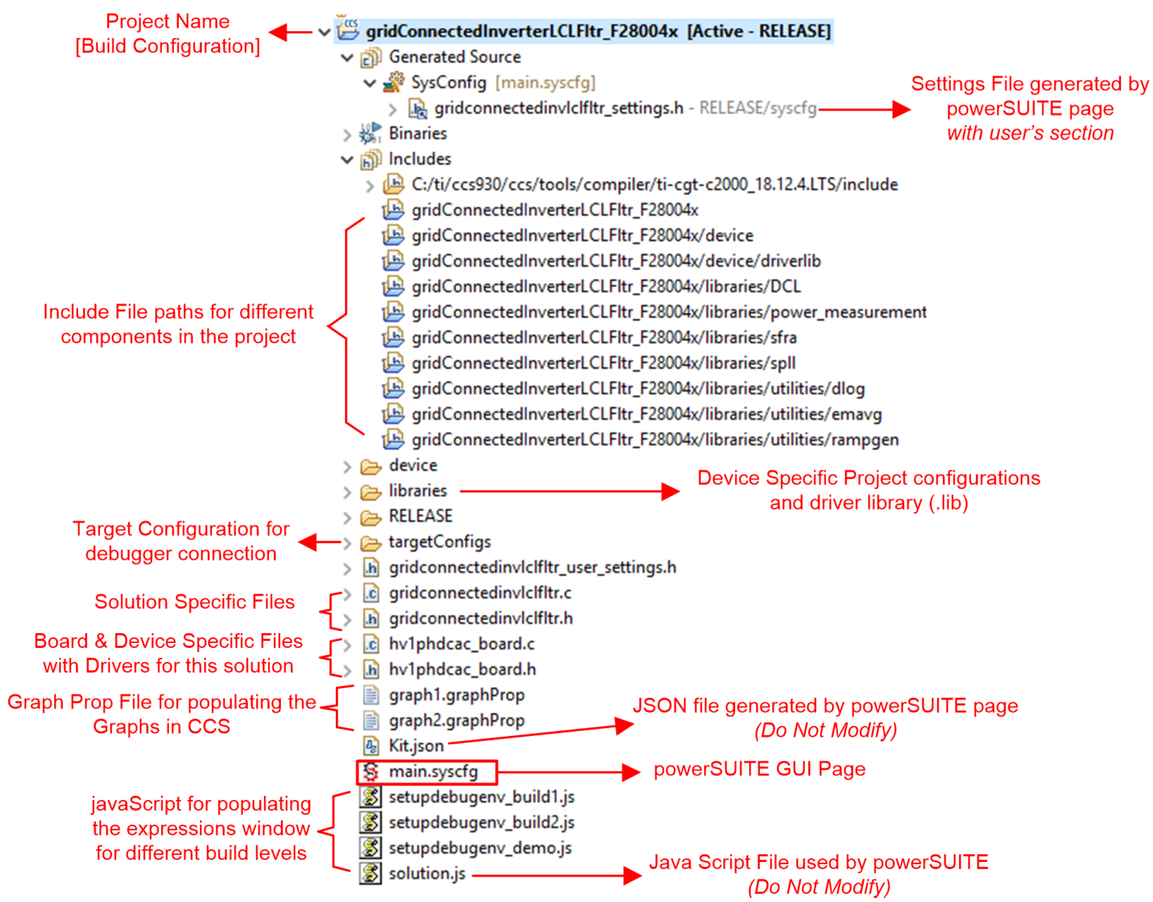
Solution specific and device independent files are *gridconnectedinvlclfltr.c/h*. This file consists of the main.c file of the project and is responsible for the control structure of the solution.

Board specific and device specific files are *hv1phdcac_board.c/h*. This file consists of device specific drivers to run the single-phase inverter.

The powerSUITE page can be opened by clicking on the *main.syscfg* file, listed under the project explorer. The powerSUITE page generates the *gridconnectedinvlclfltr_settings.h* file. This file is the only file used in the compile of the project that is generated by the powerSUITE page. Pin mapping and other user defined settings are located in *gridconnectedinvlclfltr_user_settings.h*.

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

Figure 20. Project Explorer View of Solution Project



The project consists of an interrupt service routine which is called every PWM cycle called *inverterISR()* where the control algorithm is executed. In addition, there are background tasks A0–A4, B0–B4, and C0–C4 which are called in a polling fashion, and may be used to run slow tasks for which absolute timing accuracy is not required.

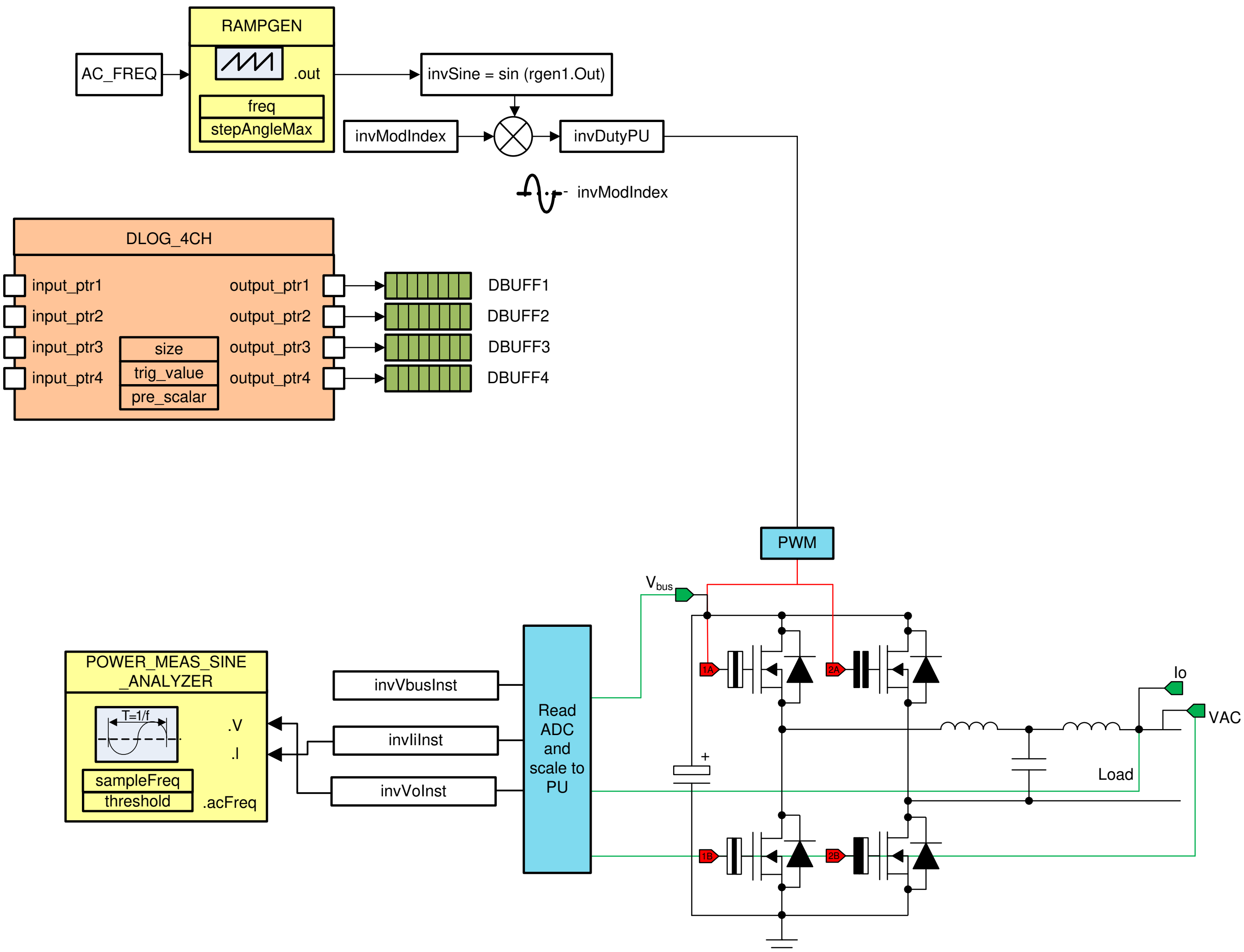
3.1.2.3 Running the Project

The software of this reference design is organized in two incremental builds and a few options to test the control loop design. The incremental build process simplifies the system bring-up and design. This process is outlined in Section 3.1.2.3.1. If using the hardware of this reference design, make sure the hardware setup is completed as outlined in Section 3.1.1.

3.1.2.3.1 Build Level 1—Open Loop

In this build, the inverter is excited in open loop fashion with a fixed modulation index, as shown in Figure 21. First, a ramp generator generates the theta angle, which is then used to compute the sine value. This sine value is multiplied with the *invModIndex* variable, which gives the duty cycle *invDutyPU* with which the inverter full bridge is modulated. Check the modulation scheme and feedback values from the power stage in this build to ensure they are correct and there are no hardware issues.

Figure 21. Build Level 1 Control Diagram: Open Loop Project



3.1.2.3.1.1 Setting Software Options for Build 1

1. Make sure the hardware is setup as shown in Figure 16. Do not supply any high-voltage power to the board yet.
2. On the powerSUITE page, select under the project options section:
   - Select *Open Loop* for the build level.
   - Select *AC* for the *Output*.
   - In this mode, *SDFM* is the only sensing method supported.
   - Enter the output frequency as 60 Hz.
   - Update the *ADC Sensing Parameters* and/or *SDFM Sensing Parameters* with the sensing resistors used in each case if they differ from the ones provided.
   - Specify the switching frequency.
   - Specify the dead band and the power rating.
   - Save the page.

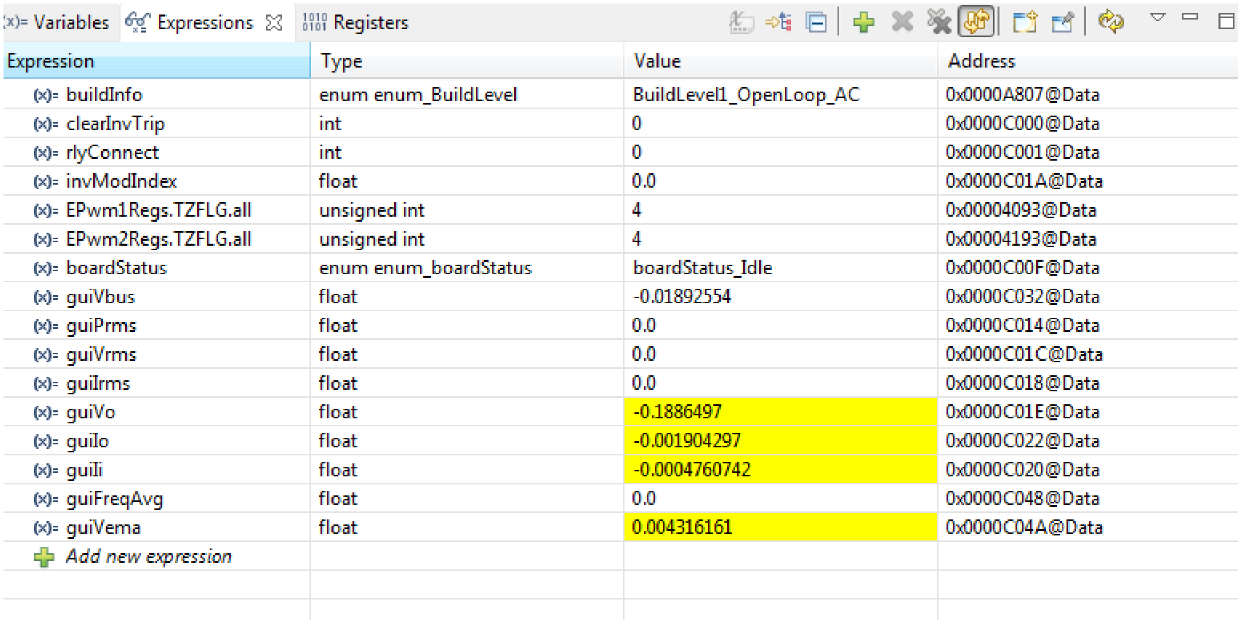
3.1.2.3.1.2 Building and Loading the Project

1. Right-click on the project name.
2. Click *Rebuild Project*.
3. In the *Project Explorer*, set the correct target configuration file as *Active*, as shown in Figure 20.
4. Make sure the board has bias power; for example, the 15-V supply is connected and the switch S1 is *ON*. This power is confirmed by the LEDs lighting up on the base board and on the control card.
5. Connect a USB cable from the control card to the host machine on which CCS is running.
6. Click *Run* → *Debug*. A debugging session launches, and a window can appear to select the CPU on which the debug must be performed.
7. In this case, select *CPU1*.
8. The project loads on the device and the CCS debug view becomes active.
9. The code halts at the start of the main routine.

3.1.2.3.1.3 Setup Debug Environment Windows

1. To add variables in the watch and expressions widow, click *View* → *Scripting Console* to open the scripting console dialog box.
2. Click *Open* on the upper right corner of the console to browse to the *setupdebugenv_build1.js* script file located inside the project folder.
3. The watch window populates with the appropriate variables required to debug the system, as shown in Figure 22.
4. Click on the *Continuous Refresh Button* on the watch window to enable the continuous update of values from the controller.
5. Figure 22 shows how the watch window appears.

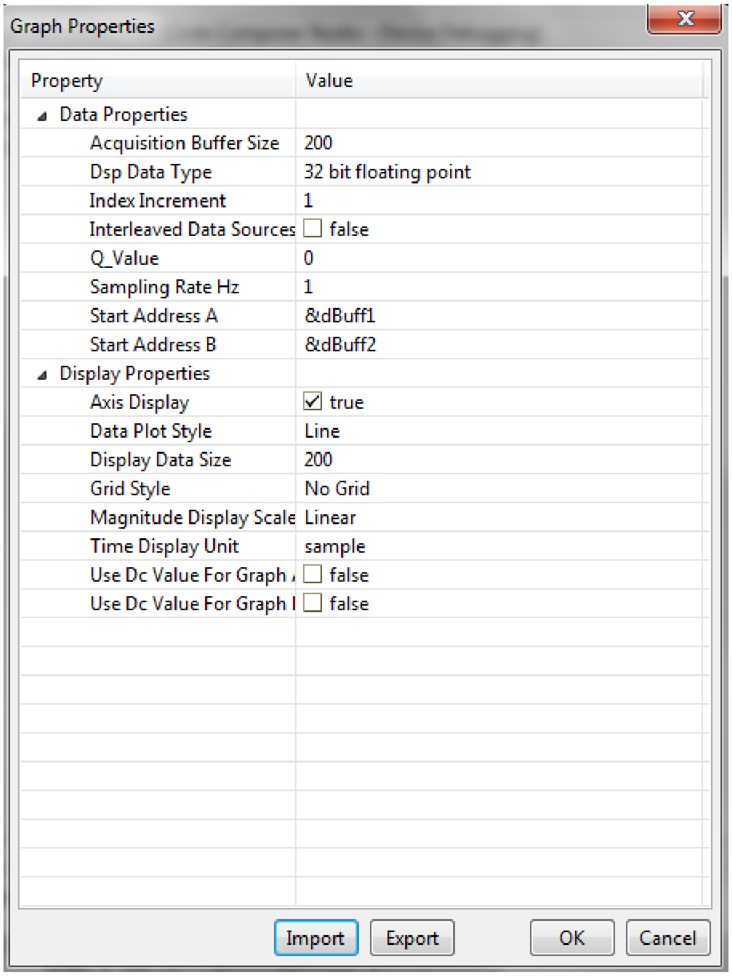
Figure 22. Build Level 1 Expressions View



Expression	Type	Value	Address
buildInfo	enum enum_BuildLevel	BuildLevel1_OpenLoop_AC	0x0000A807@Data
clearInvTrip	int	0	0x0000C000@Data
rlyConnect	int	0	0x0000C001@Data
invModIndex	float	0.0	0x0000C01A@Data
EPwm1Regs.TZFLG.all	unsigned int	4	0x00004093@Data
EPwm2Regs.TZFLG.all	unsigned int	4	0x00004193@Data
boardStatus	enum enum_boardStatus	boardStatus_Idle	0x0000C00F@Data
guiVbus	float	-0.01892554	0x0000C032@Data
guiPrms	float	0.0	0x0000C014@Data
guiVrms	float	0.0	0x0000C01C@Data
guiIrms	float	0.0	0x0000C018@Data
guiVo	float	-0.1886497	0x0000C01E@Data
guiIo	float	-0.001904297	0x0000C022@Data
guiIi	float	-0.0004760742	0x0000C020@Data
guiFreqAvg	float	0.0	0x0000C048@Data
guiVema	float	0.004316161	0x0000C04A@Data
Add new expression			

6. Verify the inverter current and voltage measurements by viewing the data in the graph window. These values are logged in the *inverterISR() routine*.
7. Go to *Tools* → *Graph* → *DualTime*.
8. Click on *Import*.
9. Point to the *graph1.GraphProp* file inside the project folder, and the graph properties window populates.
10. Alternatively, the user may enter the values as shown in Figure 23.
11. When the entries are verified, click *OK*.
12. Two graphs appear in CCS.
13. Click *Continuous refresh* on these graphs.
14. If desired, add a second set of graphs by importing the *graph2.GraphProp* file.

Figure 23. Graph Settings



3.1.2.3.1.4 *Using Real-Time Emulation*

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

1. Enable real-time mode by hovering the cursor on the buttons on the horizontal toolbar and clicking the  button.
2. If a message box appears, select *YES* to enable debug events.
3. Set bit 1 (DGBM bit) to 0, and the memory and register values can be passed to the host processor for updating the debugger windows.

3.1.2.3.1.5 Running the Code

1. Run the project by clicking .
2. In the watch view, check if the *guiVbus*, *guiIi*, and *guiVo* variables are updating periodically.
3. These variables are close to zero because no high voltage has been applied to the power input yet.
4. Set the value of *rylConnect* to 1, which connects the relay and a clicking sound is audible.
5. Set the *clearInvTrip* variable to 1.
6. Set *EPwm1Regs.TZFLG.all* to zero.
7. The *boardStatus* updates to *boardStatus_NoFault*.
8. Set the *invModIndex* variable to 0.5.
9. With a resistance of 100 Ω connected at the output, raise the input DC bus slowly up to 50 V.
10. Observe the AC waveform on the oscilloscope for the voltage and current to see a clean AC waveform (expect owing to the low-frequency switching a sharp pulse around the zero crossing).
11. Now raise the DC bus voltage slowly up to 380 V.
12. Verify that the variable *guiFreqAvg* closely matches the value set on the powerSUITE page (that is, 60 Hz for the default code).
13. Confirm that the AC measurement is correct by viewing the *gui_Vrms* and *gui_Irms* values.
14. For the 380-V DC bus and 100-Ω output with a 0.5-inverter modulation index, the output voltage is close to 126 V_{RMS} and the current is 1.26 I_{RMS}.
15. If there are inconsistencies in the measured valued and actual values, confirm the hardware and enter the correct values on the powerSUITE page for the scaling and voltage currents.
16. Figure 24 shows the watch expressions window under these conditions.

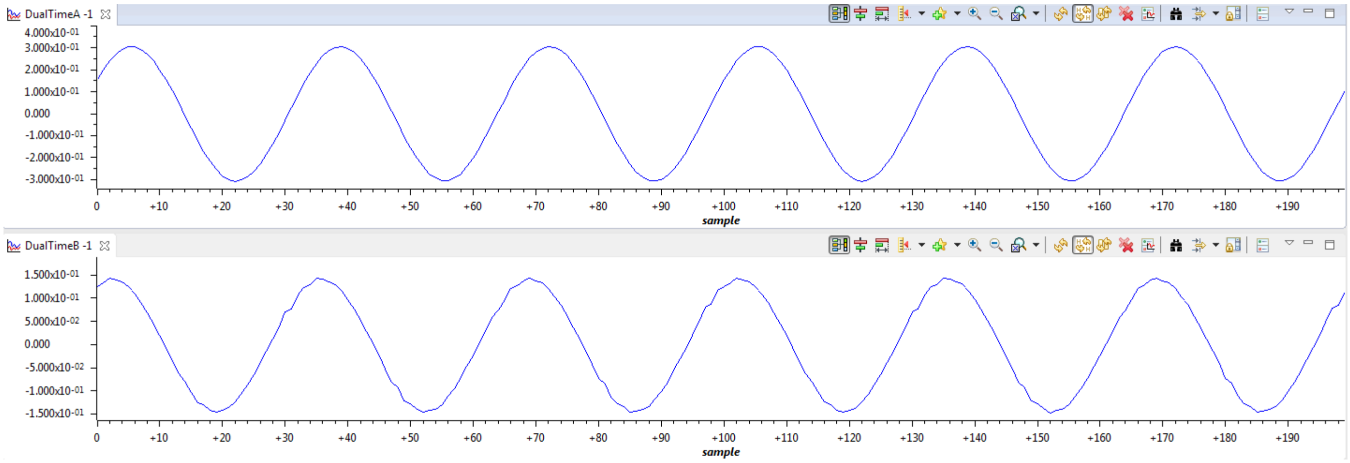
Figure 24. Build Level 1 Expressions View With Power Measurement

(x)= Variables | Expressions | Registers

Expression	Type	Value	Address
(x)= buildInfo	enum enum_BuildLevel	BuildLevel1_OpenLoop_AC	0x0000A807@Data
(x)= clearInvTrip	int	0	0x0000C000@Data
(x)= rlyConnect	int	1	0x0000C001@Data
(x)= invModIndex	float	0.5	0x0000C01A@Data
(x)= EPwm1Regs.TZFLG.all	unsigned int	0	0x00004093@Data
(x)= EPwm2Regs.TZFLG.all	unsigned int	0	0x00004193@Data
(x)= boardStatus	enum enum_boardStatus	boardStatus_Idle	0x0000C00F@Data
(x)= guiVbus	float	380.5925	0x0000C032@Data
(x)= guiPrms	float	164.1661	0x0000C014@Data
(x)= guiVrms	float	126.2137	0x0000C01C@Data
(x)= guiIrms	float	1.302251	0x0000C018@Data
(x)= guiVo	float	-144.5417	0x0000C01E@Data
(x)= guiIo	float	-1.527148	0x0000C022@Data
(x)= guiIi	float	-0.2155396	0x0000C020@Data
(x)= guiFreqAvg	float	60.01892	0x0000C048@Data
(x)= guiVema	float	113.3325	0x0000C04A@Data
Add new expression			

17. Use the graph to view the AC waveforms, as shown in Figure 25.

Figure 25. Build Level 1 Graph1.GraphProp File Showing Measured Per-Unit Voltage and Current Values



The following variables are plotted in Build 1:

- dVal1 = spll1.sin; → dBuff1
- dVal2 = invIoInst; → dBuff2
- dVal3 = invVoInst; → dBuff3
- dVal4 = invDuty; → dBuff4

Use these graphs to verify if the sensing on the board for voltages and currents is accurate. If nothing is observed in the graph , put dlog1.status in the Expression window in CCS and if it is 0 set it to 1. Also if multiple AC cycles need to be observed, enter dlog1.prescalar to be greater than 1; for example, it can be set to 5.

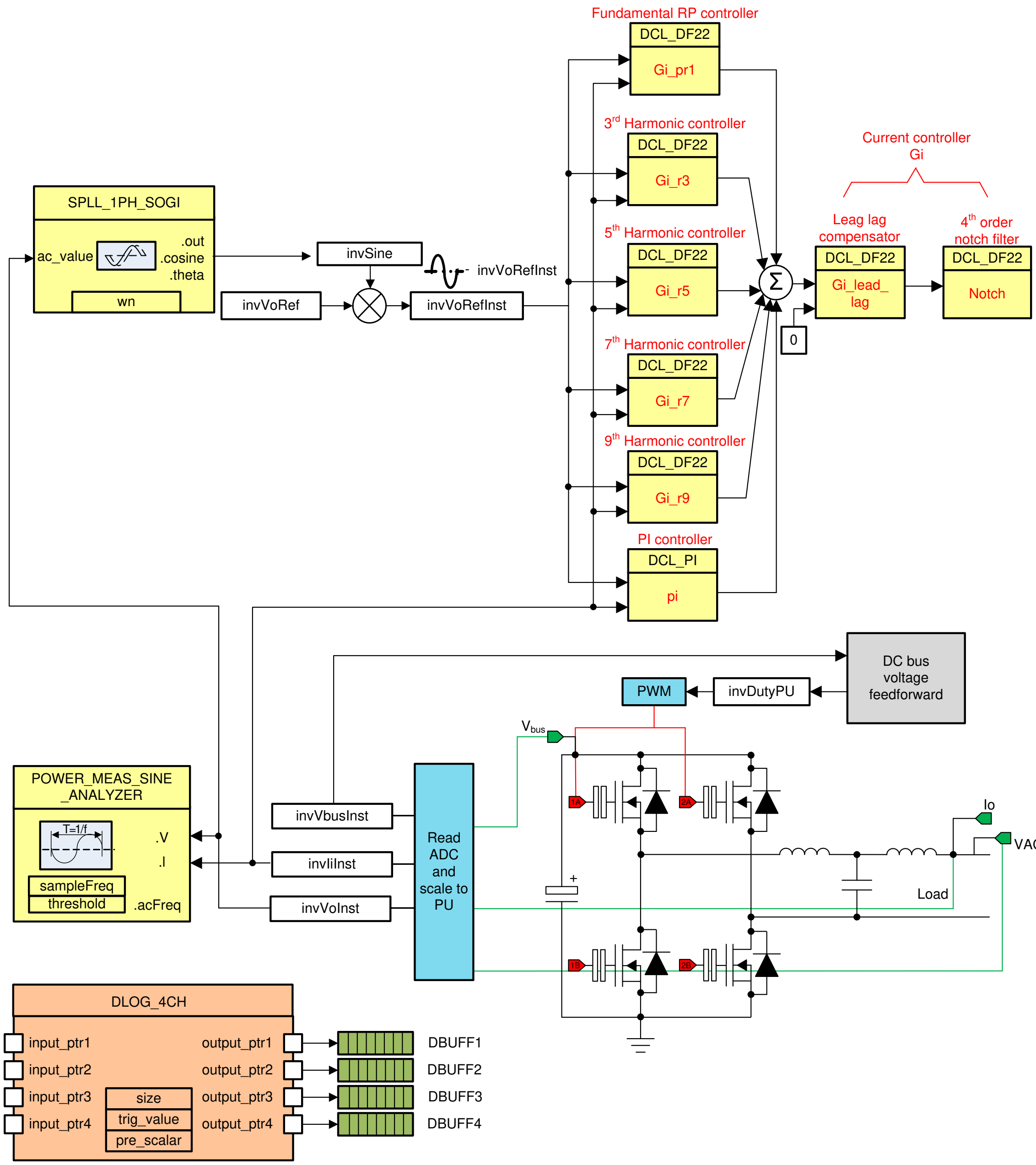
18. The AC voltage may be further modulated by changing the modulation index through the watch window.
19. Ensure that the VA rating of the inverter is never exceeded while changing this modulation.
20. The check for this build is completed, the following items are verified upon successful completion of this build:
    - Inverter modulation scheme and generation of correct AC waveform.
    - Sensing of voltages, currents, and scaling are correct.
    - Interrupt generation and execution of the Build 1 code in the inverter ISR.
21. To power down, set the *invModIndex* to zero.
22. Set *rlyConnect* to zero
23. Slowly decrease the DC bus voltage to 0 V.
24. Fully halting the MCU when in real-time mode is a two-step process:
    a. First, halt the processor by clicking the halt button on the toolbar, , or by using *Target* → *Halt*.
    b. Click  to take the MCU out of real-time mode.
25. Click  to reset the MCU.
26. Click the *Terminate Debug Session* button,  to close the CCS debug session, or go to *Target* → *Terminate all*.

3.1.2.3.2 Build Level 2—Close Current Loop: DC Check

In Build 1, the open loop operation of the inverter is verified. In Build 2, the current loop is closed; for example, the output current is controlled using a current compensator Gi. The Gi is comprised of multiple resonant controllers to provide gain at the fundamental AC frequency and harmonics, a notch filter to damp the LCL filter resonance, and a lead lag compensator to improve the phase margin. DC bus voltage feedforward is applied to the output of this current compensator to generate the duty cycle of the inverter, as shown in Equation 15. This equation makes the plant for the current compensator independent of the DC bus voltage. Figure 26 shows the control diagram of the closed current loop of Build Level 2.

$$\text{invDuty PU} = (\text{invIoRefInst} - \text{invIoInst}) \times G_i/\text{invVbusInst} \quad (15)$$

Figure 26. Build Level 2 Control Diagram: Closed Current Loop

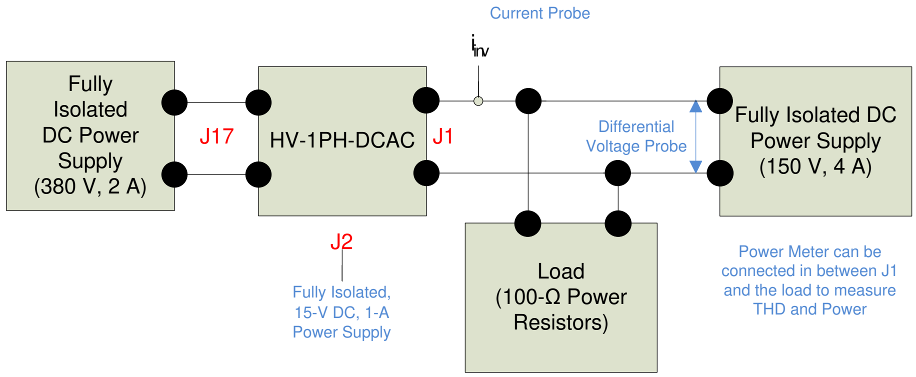


The design of the compensator is checked by operating the inverter in DC.

3.1.2.3.2.1 Setting Software Options for Build 2: DC Check

1. De-energize all sources.
2. Make sure the hardware is setup as shown in Figure 27.

Figure 27. Hardware Setup to Run Software for Build Level 2: DC Check

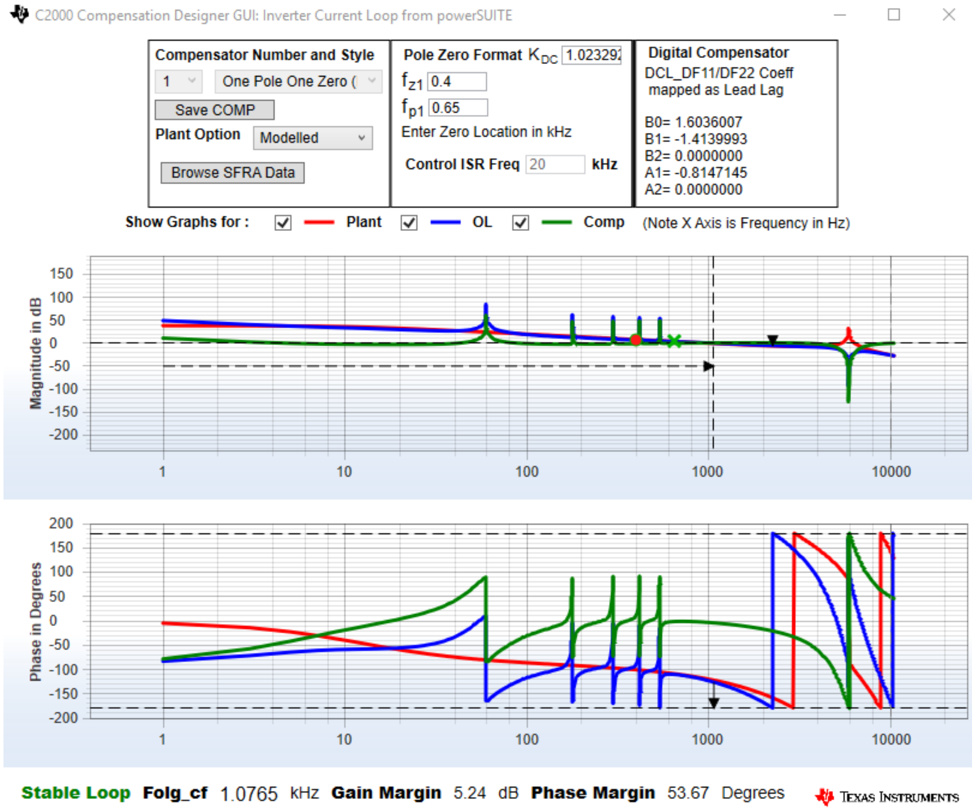


3. Do not supply any high-voltage power to the board yet.
4. Supply an isolated 15 V through J2.
5. On the powerSUITE page, select under the *Project Options* section:
   - *Closed Current Loop and Gris Sync* for the build level.
   - *DC* for the *Output*.
   - In this mode, *SDFM* is the only sensing method supported.
   - Enter the output frequency as 60 Hz as this is a DC build it will not matter.
6. If this is an adapted solution, edit the setting accordingly by specifying the switching frequency, the dead band, and the power rating.
7. Save page.

3.1.2.3.2.2 Designing the Current Loop Compensator

1. Click the compensation designer icon from the powerSUITE page to launch the compensation designer.
2. The plant model for the inverter for the output current loop is created using the parameters specified on the powerSUITE page.
3. The compensation designer GUI enables editing the lead lag compensator, which is also a part of Gi
4. If a change is required to resonant and PI controller, close the compensation designer and edit the values on the powerSUITE page.
5. Save the page.
6. Relaunch the GUI.
7. Verify the stability of the system by observing the gain and phase margins on the open loop transfer function plot in the compensation designer, as shown in Figure 28.

Figure 28. Current Loop Design Using Compensation Designer



8. Once satisfied with the compensation design, click *Save COMP*.
9. Save the compensator values into the project (for example, when the project is recompiled, it will use the new coefficients for the compensator).

NOTE: If the project is not selected from the solution adapter, the compensator cannot be changed. For a unique design, select the solution through the solution adapter.

10. Close the compensation designer and return to the powerSUITE page.

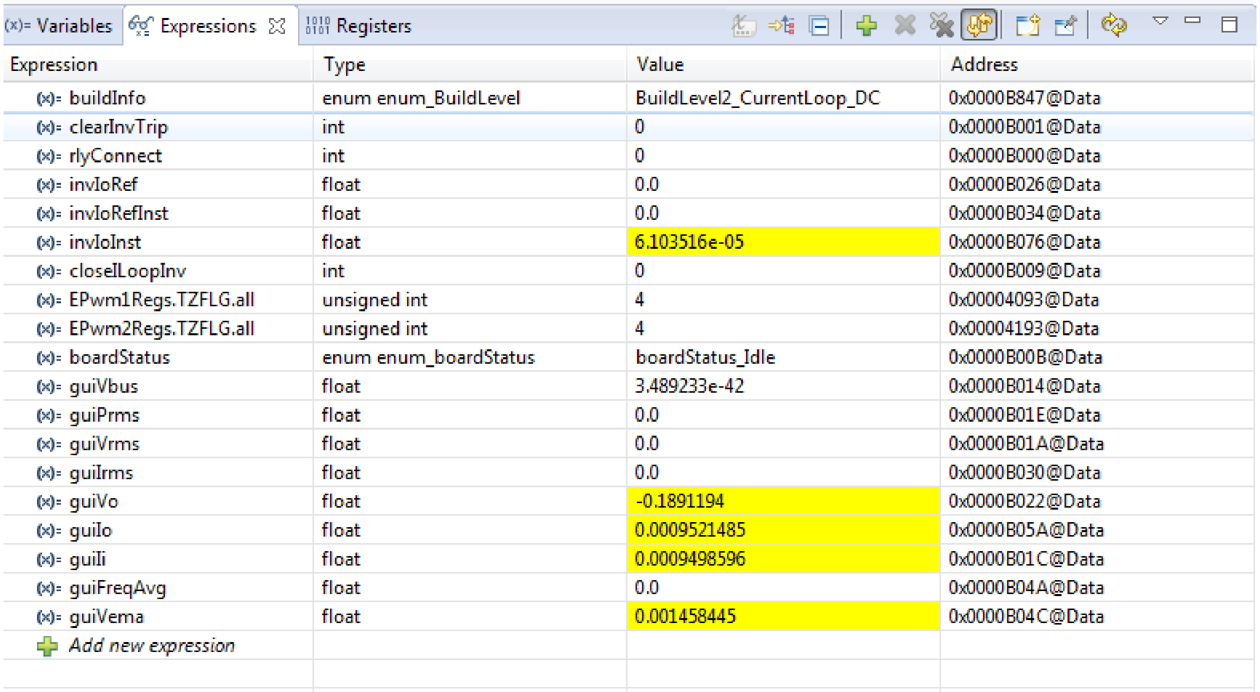
3.1.2.3.2.3 Building and Loading the Project and Setting up Debug

1. Right-click on the project name.
2. Click *Rebuild Project*.
3. The project builds successfully.
4. Click *Run* → *Debug*, a debugging session launches.
5. Select the CPU in the window that appears to perform the debug in case of dual PC devices.
6. In this case, select *CPU1*.
7. The project loads on the device, and the CCS debug view is active.
8. The code halts at the start of the main routine.
9. Click *View* → *Scripting Console* to add variables in the watch and expressions window to open the scripting console dialog box.
10. Click *Open* to browse to the *setupdebugenv_build2.js* script file located inside the project folder.
11. The watch window populates with the appropriate variables required to debug the system, as shown in Figure 29.
12. Click on the *Continuous Refresh* button, , on the watch window to enable the continuous update of

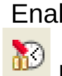
values from the controller.

13. Figure 29 shows how the watch window appears.

Figure 29. Build Level 2—DC Check Expressions Window



Expression	Type	Value	Address
buildInfo	enum enum_BuildLevel	BuildLevel2_CurrentLoop_DC	0x0000B847@Data
clearInvTrip	int	0	0x0000B001@Data
rlyConnect	int	0	0x0000B000@Data
invIoRef	float	0.0	0x0000B026@Data
invIoRefInst	float	0.0	0x0000B034@Data
invIoInst	float	6.103516e-05	0x0000B076@Data
closeILoopInv	int	0	0x0000B009@Data
EPwm1Regs.TZFLG.all	unsigned int	4	0x00004093@Data
EPwm2Regs.TZFLG.all	unsigned int	4	0x00004193@Data
boardStatus	enum enum_boardStatus	boardStatus_Idle	0x0000B00B@Data
guiVbus	float	3.489233e-42	0x0000B014@Data
guiPrms	float	0.0	0x0000B01E@Data
guiVrms	float	0.0	0x0000B01A@Data
guiIrms	float	0.0	0x0000B030@Data
guiVo	float	-0.1891194	0x0000B022@Data
guiIo	float	0.0009521485	0x0000B05A@Data
guiIi	float	0.0009498596	0x0000B01C@Data
guiFreqAvg	float	0.0	0x0000B04A@Data
guiVema	float	0.001458445	0x0000B04C@Data
Add new expression			

14. Do not use the graph window as this is a DC check.
15. Enable real-time mode by hovering the cursor on the buttons on the horizontal toolbar and clicking the  button.

3.1.2.3.2.4 Running the Code

1. Click  to run the project.
2. Set the *clearInvTrip* variable to 1 to clear the inverter trip.
3. Set the *rylconnect* variable to 1 to connect the relay.
4. Slowly raise the input DC bus to approximately 50 V.
5. Observe that the expressions window shows the correct value for *guiVbus*.
6. Slowly increase the voltage of the DC source connected at the output of the inverter to 20 V.
7. A resistive load is connected in parallel so a small current is drawn from the DC source.
8. Watch for any inadvertent events, such as the DC bus of the inverter input rising or a very high current draw.
9. These events may point to a problem in the setup of the inverter.
10. Revisit Build Level 1, and verify that everything is correct before you proceed.
11. Observe that at this stage, the resistive load is supplied with power from the DC source connected at the output.
12. The output voltage is 20 V, and for a resistive load, the current with a 100-Ω load is approximately 0.2 A.

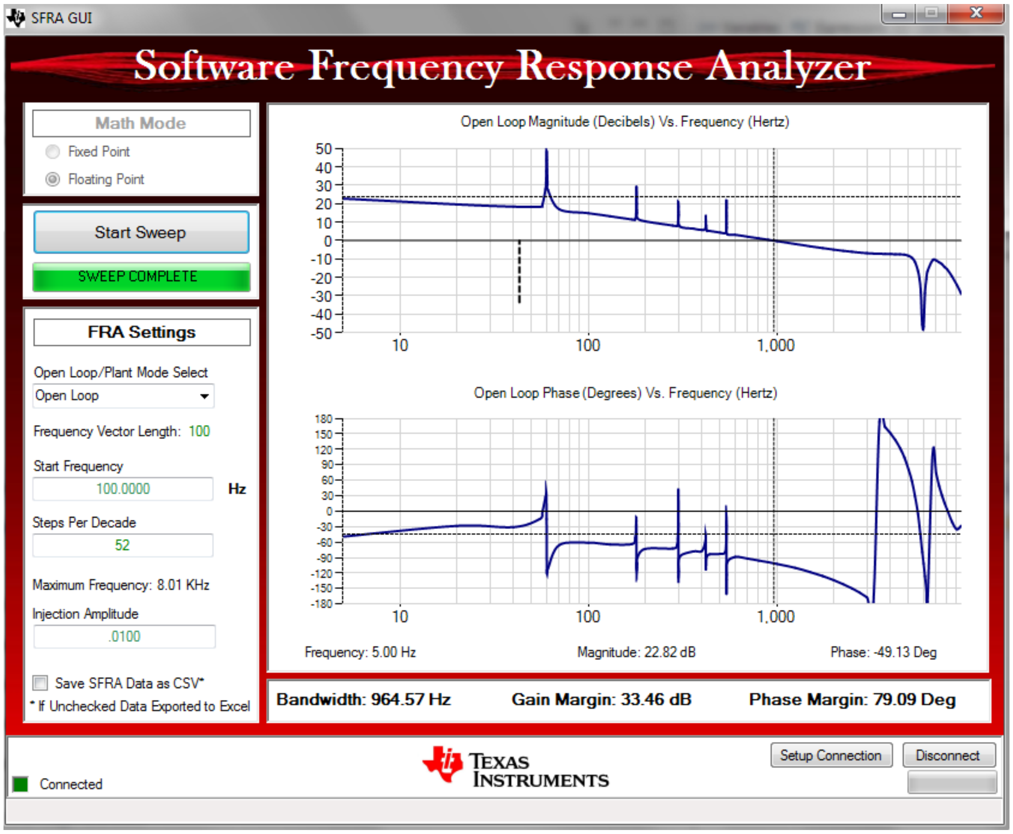
NOTE: Verify the sign of the output voltage guiVo is also correct (that is, 20 V). If not, reduce the DC source connected to the output from 20 V to 0 V, reduce the input DC source voltage to 0, and swap the connection terminals of the output terminal. A re-load of the code may be necessary. Resume the debug and verify that now the guiVo is read as 20 V.

13. Increase *invIoRef* to 0.01 to check the output current control of the inverter. This PU reference for this design corresponds to approximately 0.17 A. Additionally, the variable invIoInst is monitored in the watch window and can be used to check closed loop operation. This variable is the actual output current measured and will follow the reference current set in invIoRef.
14. Observe that the current supplied by the DC source at the output decreases, and the inverter supplies the rest of the DC current.

NOTE: As this is DC operation, the inverter operates in buck mode.

15. Increase the DC bus to 380 V. Maintain the closed loop operation as the user raises the DC bus. The feedforward term of the DC bus ensure that the closed loop performance remains the same.
16. Increase the DC voltage of the power supply connected at the output to approximately 120 V.
17. Increase the *invIoRef* further to 0.07 in steps of 0.01.
18. Observe that with each increment, more current is sourced from the inverter and less current is supplied by the DC source connected at the output.
19. Make sure that the DC current from the supply connected at the output is never zero. This verifies that the output current control is working to verify robustness.
20. SFRA is integrated in the software of this build which may be used to verify the designed compensators, and check if enough gain and phase margin are present.
21. Keep the project running to run the SFRA from the cfg page.
22. Click on the *SFRA* icon.
23. The *SFRA GUI* appears.
24. Select the options for the device on the GUI. For example, select *floating point* for F28377D.
25. Click *setup connection*.
26. Select an appropriate *COM port*.
27. Click *OK*.
28. Return to the *SFRA GUI*.
29. Click *Connect*.
30. The *SFRA GUI* connects to the device.
31. Click *Start Sweep* to begin an SFRA sweep.
32. The complete SFRA sweep takes around 5 minutes to finish.
33. Monitor activity by watching the progress bar on the SFRA GUI and by checking the flashing of the blue LED light on the back of the control card that indicates UART activity. Alternatively, the user can enter "SFRA1". The FreqIndex variable in the watch window tells where the SFRA is in the frequency sweep. The sweep is of 100 points in this software and will complete when this variable reached a value of 100.
34. Once complete, a graph with the open loop plot appears, as shown in Figure 30.
35. The plot verifies that the designed compensator is stable.

Figure 30. SFRA Run on Closed Current Loop



NOTE: The SFRA GUI is designed for log step frequency points hence in case of this project where this is not valid the SFRA GUI can crash and not display the complete waveform. Under such a situation, the following workaround is recommended:

1. Close the SFRA GUI, go to *<sdk_install_path>/libraries/sfra/GUI* and launch the SFRA.exe file from here.
2. Connect the SFRA GUI and make sure "Save SFRA Data as CSV" is not checked. This forces the SFRA data to be saved into an Excel sheet. Even when the SFRA GUI reports an error after the retrieval of the data, select continue and the Excel sheet will be populated with all the frequency data. This data can then be used in MATLAB or saved as a .CSV file and used in compensation designer.
3. If the SFRA GUI errors out, skip the following steps from 36 to 45.

36. The frequency response data is also saved in the project folder, under an SFRA Data Folder, and is stamped with the time of the SFRA run.
37. Go back to the powerSUITE page.
38. Open *Compensation Designer* once the sweep is complete.
39. Click on *Compensation Designer*.
40. Choose *SFRA Data* for the plant option on the GUI.
41. The *SFRA Data* uses the measured plant information to designer the compensator. This option may be used to fine-tune the compensation.
42. By default, the compensation designer points out the latest *SFRA run*.
43. If a previous *SFRA run* plant information is required to be used, select the *SFRAData.csv* file and click *Browse SFRA DATA*.
44. Close the compensation designer to return to the configuration page. Figure 31 shows the compensation designer with measured plant frequency response data.

Figure 31. Compensation Designer With Measured Plant Frequency Response Data

45. This verifies the current compensation design, both modeled and measured to match closely. The compensator may further be adjusted to achieve the best performance from the system.
46. Set the *invIoRef* to zero.

47. Set *rlyConnect* to zero to disconnect the relay.
48. Reduce the DC bus of the supply connect at the input to zero.
49. Reduce the output voltage of the supply connected at the output of the inverter to zero.
50. Fully halting the MCU in real-time mode is a two-step process.
51. Halt the processor by using the *Halt button* on the toolbar, , or by using *Target* → *Halt*.
52. Click on to take the MCU out of real-time mode.
53. Click to reset the MCU.
54. Click on the *Terminate Debug Session button*, , or go to *Target* → *Terminate All* to close the CCS debug session.

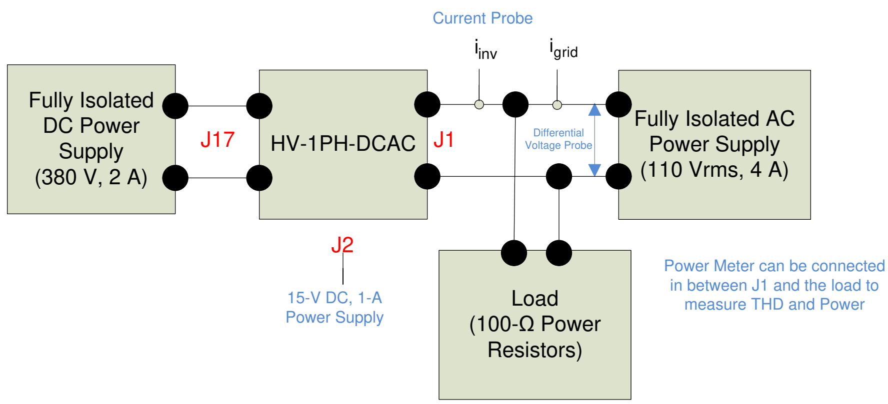
3.1.2.3.3 Build Level 2—Close Current Loop: AC Check

In Section 3.1.2.3.2, the output current closed loop operation of the inverter is verified. In this build, the current loop is closed with AC output voltage.

3.1.2.3.3.1 Setting Software Options for Build 2: AC Check

Set up the hardware as shown in Figure 32. Do not supply any high-voltage power to the board yet. TI recommends to use a controlled source at the output, such as an AC power supply to verify grid connected operation. Once the operation is verified, check the functioning of the inverter with direct grid connection. Bias supply to the board is provided by an isolated 15-V supply connected to J2 and S1 in the *ON* position.

Figure 32. Setup to Run Software for Build Level 2: AC Check



On the powerSUITE page, select under the *Project Options* section:

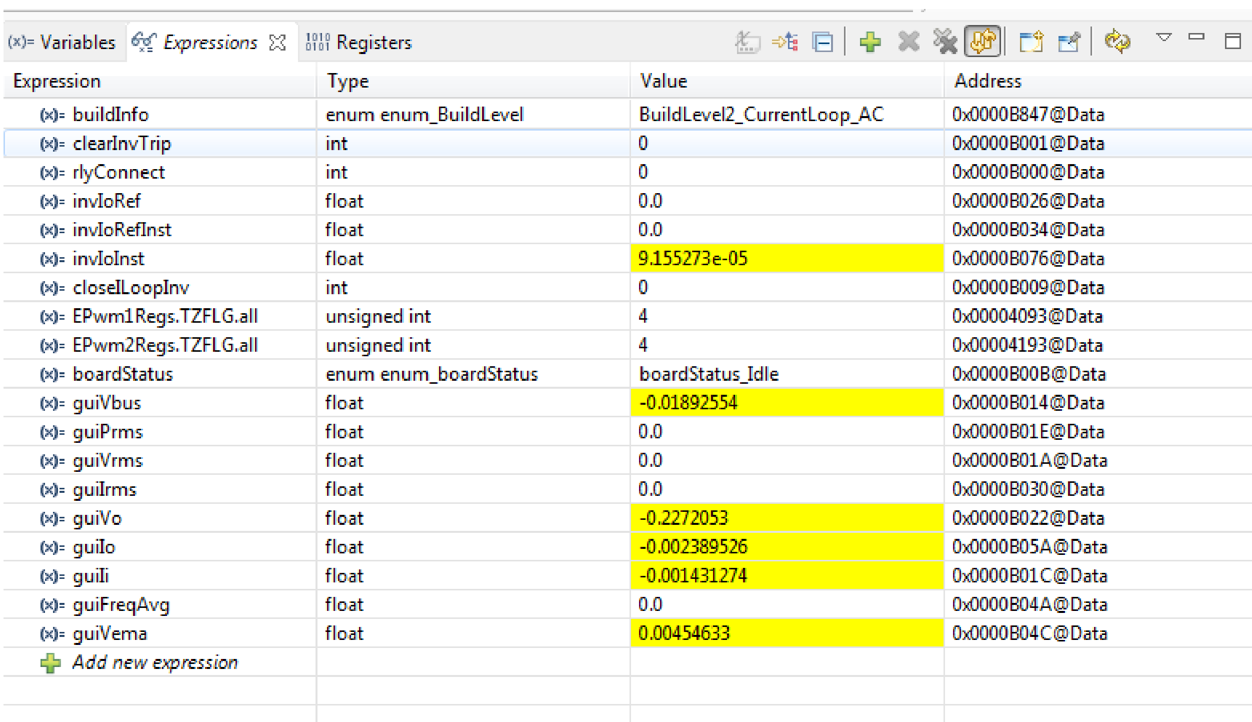
- *Closed Current Loop and Grid Sync* for the *build level*.
- *AC* for the *Output*.
- In this mode, *SDFM* is the only sensing method supported.
- Enter 60 Hz for the *output frequency*.

  If this is an adapted solution, edit the setting accordingly and specify the switching frequency, the dead band, and the power rating.
- Save the page.

3.1.2.3.3.2 Building and Loading the Project and Setting Up Debug

1. Right-click on the project.
2. Click *Rebuild Project*.
3. The project builds successfully.
4. Click *Run* → *Debug*, and a debugging sessions launches.
5. Select the CPU when the window appears to perform the debug in case of dual PC devices.
6. Select *CPU1* in this case.
7. The project loads the device and the CCS debug view is active.
8. The code halts at the start of the main routine.
9. Click *View* → *Scripting Console* to open the scripting console dialog box, and to add the variables in the watch and expressions window.
10. Click on *Open to browse* on the upper right corner of the console to browse to the *setupdebugenv_build2.js* script file, located inside the project folder.
11. The watch window populates with the appropriate variables required to debug the system.
12. Click on the *Continuous Refresh* button, , on the watch window to enable a continuous update of values from the controller.
13. Figure 33 shows how the watch window appears.

Figure 33. Build Level 2: AC Check Expressions Window



Expression	Type	Value	Address
buildInfo	enum enum_BuildLevel	BuildLevel2_CurrentLoop_AC	0x0000B847@Data
clearInvTrip	int	0	0x0000B001@Data
rlyConnect	int	0	0x0000B000@Data
invIoRef	float	0.0	0x0000B026@Data
invIoRefInst	float	0.0	0x0000B034@Data
invIoInst	float	9.155273e-05	0x0000B076@Data
closeILoopInv	int	0	0x0000B009@Data
EPwm1Regs.TZFLG.all	unsigned int	4	0x00004093@Data
EPwm2Regs.TZFLG.all	unsigned int	4	0x00004193@Data
boardStatus	enum enum_boardStatus	boardStatus_Idle	0x0000B00B@Data
guiVbus	float	-0.01892554	0x0000B014@Data
guiPrms	float	0.0	0x0000B01E@Data
guiVrms	float	0.0	0x0000B01A@Data
guiIrms	float	0.0	0x0000B030@Data
guiVo	float	-0.2272053	0x0000B022@Data
guiIo	float	-0.002389526	0x0000B05A@Data
guiIi	float	-0.001431274	0x0000B01C@Data
guiFreqAvg	float	0.0	0x0000B04A@Data
guiVema	float	0.00454633	0x0000B04C@Data
Add new expression			

14. Use the graph window to observe the variables as this is a check.
15. Enable real-time mode by hovering the cursor on the buttons on the horizontal toolbar and clicking .

3.1.2.3.3.3 Running the Code

1. Click  to run the project.
2. Raise the input DC bus to 50 V.
3. Set the AC voltage of the supply connected to the output of the inverter to 20 V_{RMS} and 60 Hz. This supply provided the current to the resistive load connected at the output of the inverter.
4. Set *rlyConnect* to 1 to connect the relay.
5. Set *invIoRef* to 0.01.
6. Set *clearInvTrip* to 1. The inverter provides some current to the resistive load, and the current from the AC power supply drops slightly.
7. Raise the DC bus to 380 V.

NOTE: As the inverter is operating at very low power, observe any spikes around the zero crossing. Ignore these spikes for now as they reduce once operating at rated current and voltage. An audible buzz may also be heard.

---

8. Increase the output AC voltage to 110 V_{RMS}.
9. Increase the current command so that more current is provided by the inverter. Increase the step size by 0.01 until approximately 0.07 per unit scaling.
10. Make sure that the current for the AC source *never goes to zero*, as many AC sources do not accept four quadrant operation.
11. Watch for any inadvertent events, such as the DC bus of the inverter input rising, or very high current draw. These events may point to a problem in the setup of the inverter.
12. Revisit Section 3.1.2.3.1 and Section 3.1.2.3.2 to verify that everything is correct before proceeding.
13. Observe the current that is shared on the load by the inverter, and the AC source. Spiking around the zero crossing can occur. These spikes may be mitigated by the user by selecting a different inverter configuration, or using a different modulation scheme.
14. The verification of the grid connected mode of operation is complete.
15. Disengage the relay setting by setting *rlyconnect* to zero to stop the inverter.
16. Reduce the DC bus and the AC source to zero to completely de-energize the hardware setup.
17. Fully halting the MCU when in real-time mode is a two-step process:
    a. Click the *Halt Button* on the toolbar, , or use *Target* → *Halt* to stop the processor.
    b. Click  to take the MCU out of real-time mode.
18. Click  to reset the MCU.
19. Click the *Terminate Debug Session* button, , or go to *Target* → *Terminate All* to close the CCS debug session.

3.1.2.3.4 Build Level 2—Close Current Loop: Demo

If the same hardware as this reference design is available, then run a demo mode as outlined in this section. In this build, a state machine is invoked to take care of the following checks:

- Check if the grid voltage and frequency is within a universal grid value rang. If these are exceeded, trip the inverter.
- Check if the DC bus is greater than the grid voltage max to ensure that power may be fed from the inverter to the grid.
- Tune the PR controller according to the measured frequency of the grid on the controller.
- Safely start and stop the inverter by zeroing the controller history and setting the current command appropriately.

3.1.2.3.4.1 Setting Software Options for BUILD 2: AC Check

Make sure the hardware is set up as shown in Figure 32. Do not supply any high-voltage power to the board yet. TI recommends to use a controlled source at the output, such as an AC power supply to verify grid connected operation. Once the operation is verified, check the functioning of the inverter with direct grid connection.

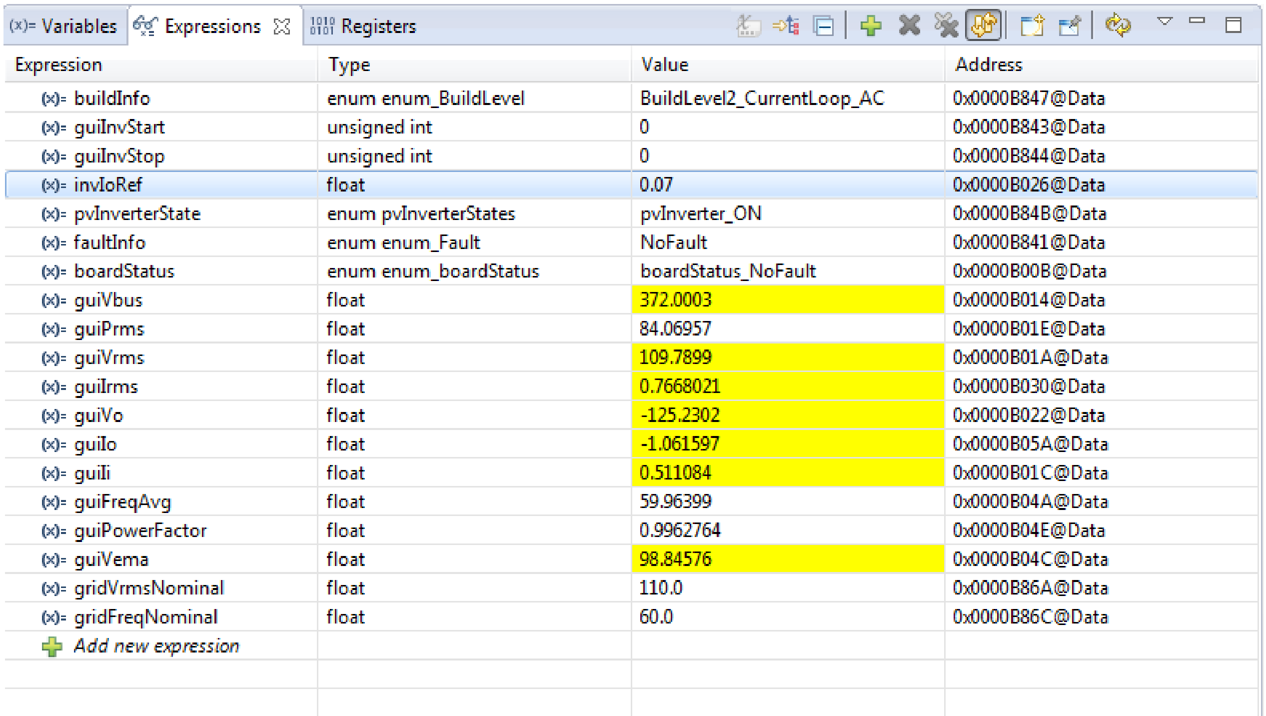
On the powerSUITE page, select under the *Project Options* section:

- *Closed Current Loop and Grid Sync* for the build level.
- *AC* for the *Output*.
- In this mode, *SDFM* is the only sensing method supported.
- Enter 60 Hz for the output frequency.

3.1.2.3.4.2 Building and Loading the Project and Setting Up Debug

1. Right-click on the *project name*.
2. Click *Rebuild Project*.
3. The project builds successfully.
4. Click *Run* → *Debug* and a debugging sessions launches.
5. Select the CPU when the window appears to perform the debug in the case of dual CPU devices.
6. Select *CPU1* in this case.
7. The project loads on the device, and the CCS debug view becomes active.
8. The code halts at the start of the main routine.
9. Click *View* → *Scripting Console* to add the variables in the watch and expressions window and to open the scripting console dialog box.
10. Click *Open to browse* on the upper right corner of the console to browse to the *setupdebugenv_demo.js* script file located inside the project folder.
11. The watch window populates with the appropriate variables required to debug the system, as shown in Figure 34.
12. Click on the *Continuous Refresh* button, , on the watch window to enable a continuous update of values from the controller.
13. Figure 34 shows how the watch window appears.

Figure 34. Build Level 2: Demo Mode Expressions View



Expression	Type	Value	Address
buildInfo	enum enum_BuildLevel	BuildLevel2_CurrentLoop_AC	0x0000B847@Data
guiInvStart	unsigned int	0	0x0000B843@Data
guiInvStop	unsigned int	0	0x0000B844@Data
invIoRef	float	0.07	0x0000B026@Data
pvInverterState	enum pvInverterStates	pvInverter_ON	0x0000B84B@Data
faultInfo	enum enum_Fault	NoFault	0x0000B841@Data
boardStatus	enum enum_boardStatus	boardStatus_NoFault	0x0000B00B@Data
guiVbus	float	372.0003	0x0000B014@Data
guiPrms	float	84.06957	0x0000B01E@Data
guiVrms	float	109.7899	0x0000B01A@Data
guiIrms	float	0.7668021	0x0000B030@Data
guiVo	float	-125.2302	0x0000B022@Data
guiIo	float	-1.061597	0x0000B05A@Data
guiIi	float	0.511084	0x0000B01C@Data
guiFreqAvg	float	59.96399	0x0000B04A@Data
guiPowerFactor	float	0.9962764	0x0000B04E@Data
guiVema	float	98.84576	0x0000B04C@Data
gridVrmsNominal	float	110.0	0x0000B86A@Data
gridFreqNominal	float	60.0	0x0000B86C@Data
Add new expression			

14. Hover the cursor on the buttons on the horizontal toolbar to enable real-time mode.
15. Click the button.

3.1.2.3.4.3 Running the Code

1. Click to run the project.
2. Set *guiInvStart* to 1 to begin the demo code.
3. As no grid voltage (that is, voltage from the AC source connected to the output of the inverter) is present, the inverter will be in a checkGrid state.
4. Slowly raise the AC voltage to 110 V_{RMS} at 60 Hz.
5. The inverter state machine then sequences to checking for DC voltage.
6. To feed current into the grid the DC voltage (which in case of PV inverters is provided from the panel or panel plus some conditioning circuit), it must be greater than the peak of the AC voltage connected at the output of the inverter.
7. In this case, the output voltage of 110 V_{RMS} is connected, raise the DC bus to greater than 200 V to let the inverter start and feed power into the grid.
8. As soon as the input DC voltage is raised above 200 V, for this setup, hear the relay click when the inverter starts. Increase the DC bus up to the rated voltage of 380 V.
9. Now increase the current reference to modulate the power that is fed from the inverter by changing *invIoRef*. For the setup described in this design guide, change this from 0.01 to 0.07 gradually in steps of 0.01.
10. Observe the current that is shared on the load by the inverter, and the AC source. Spiking around the zero crossing can occur. These spikes can be mitigated by the user by selecting a different inverter configuration, or using a different modulation scheme.
11. The verification of the grid connected mode of operation is complete.
12. The inverter can be stopped by writing 1 to *guiInvStop*, which disengages the relay.
13. Reduce the DC bus to zero and reduce the AC voltage connected at the inverter output to zero.

14. Fully halting the MCU when in real-time mode is a two-step process:
    a. Click the *Halt Button* on the toolbar, , or use *Target* → *Halt* to stop the processor.
    b. Click  to take the MCU out of real-time mode.
15. Click  to reset the MCU.
16. Click the *Terminate Debug Session* button, , or go to *Target* → *Terminate All* to close the CCS debug session.

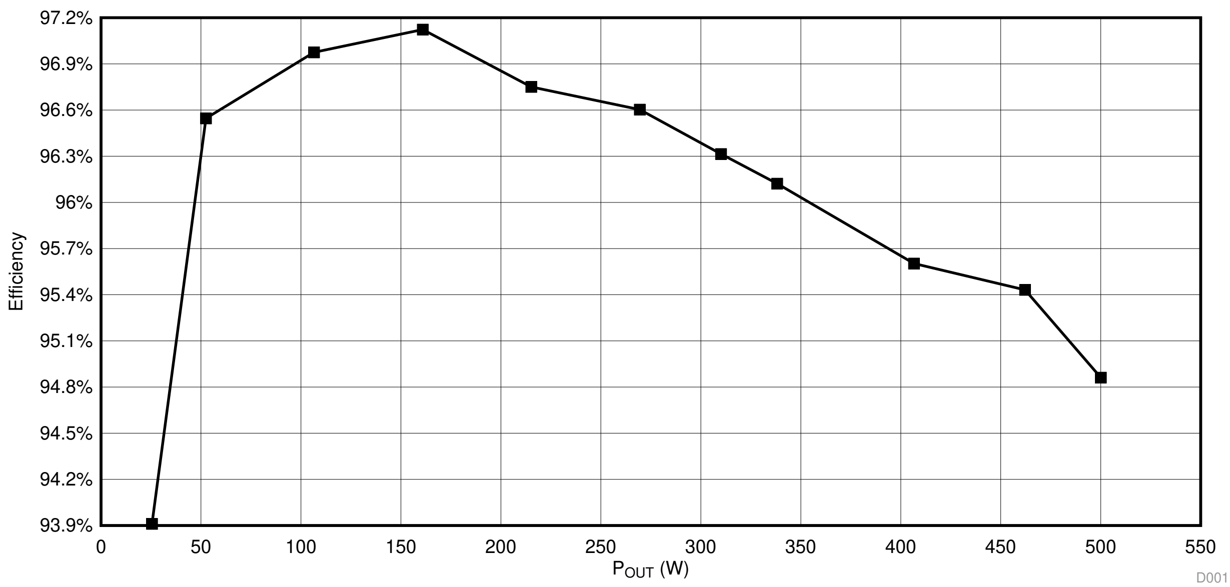
3.2 Testing and Results

3.2.1 Test Results With Grid Connection at 120 V_{RMS} and 60-Hz Loads

3.2.1.1 Power Stage Efficiency

Figure 35 shows the power stage efficiency of the reference design when operating at 120 V_{RMS} of output.

Figure 35. Efficiency of the Design When Operating at 120 V_{RMS} of Output



3.2.1.2 Steady State Waveform

Figure 36 and Figure 37 show the steady-state current and voltage at 500 W of output power with two different zoom levels.

Figure 36. Steady-State Output Voltage and Current at 120 V_{RMS}, P_{OUT} of 500 W, Zoomed In

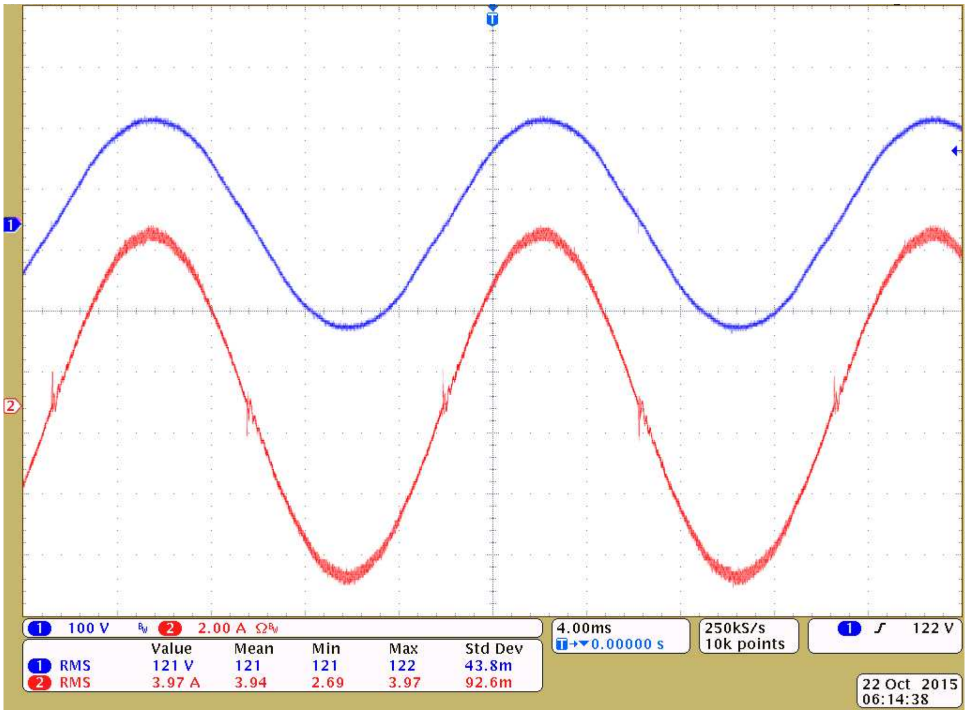


Figure 37. Steady-State Output Voltage and Current Waveform at 120 V_{RMS}, P_{OUT} of 500 W, Multiple Cycles

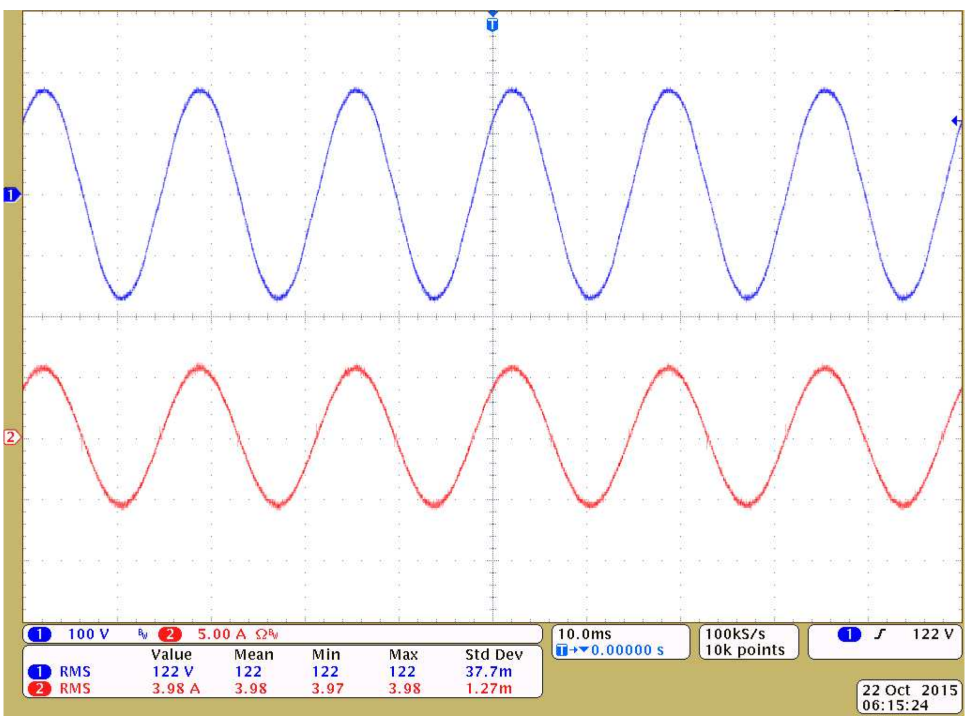
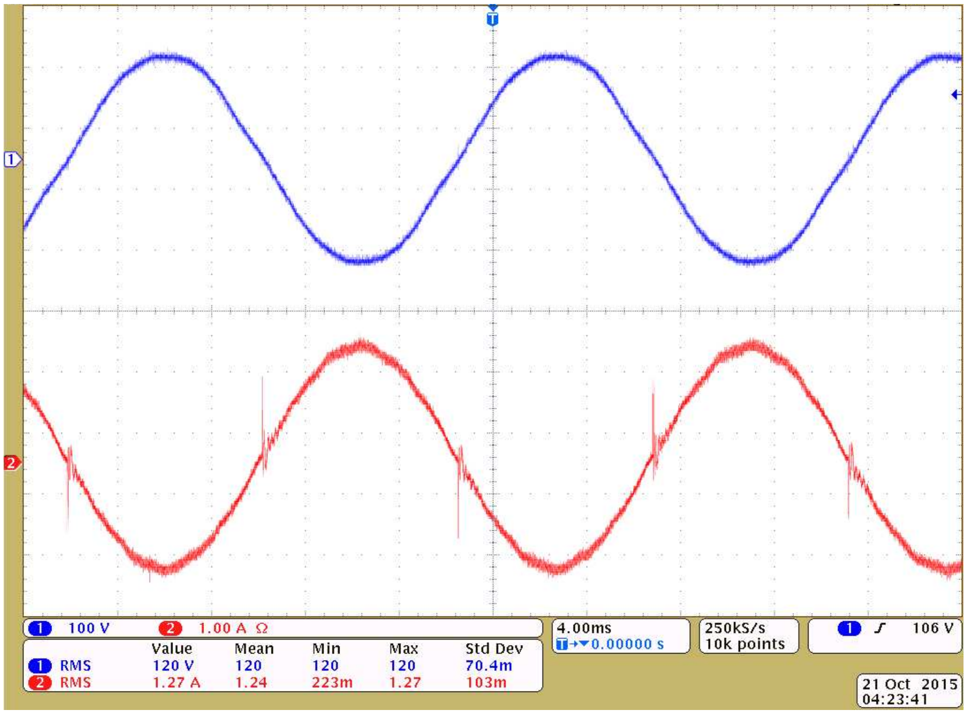


Figure 38 shows the steady state voltage and current operating at 160 W with 120 V_{RMS} of output voltage.

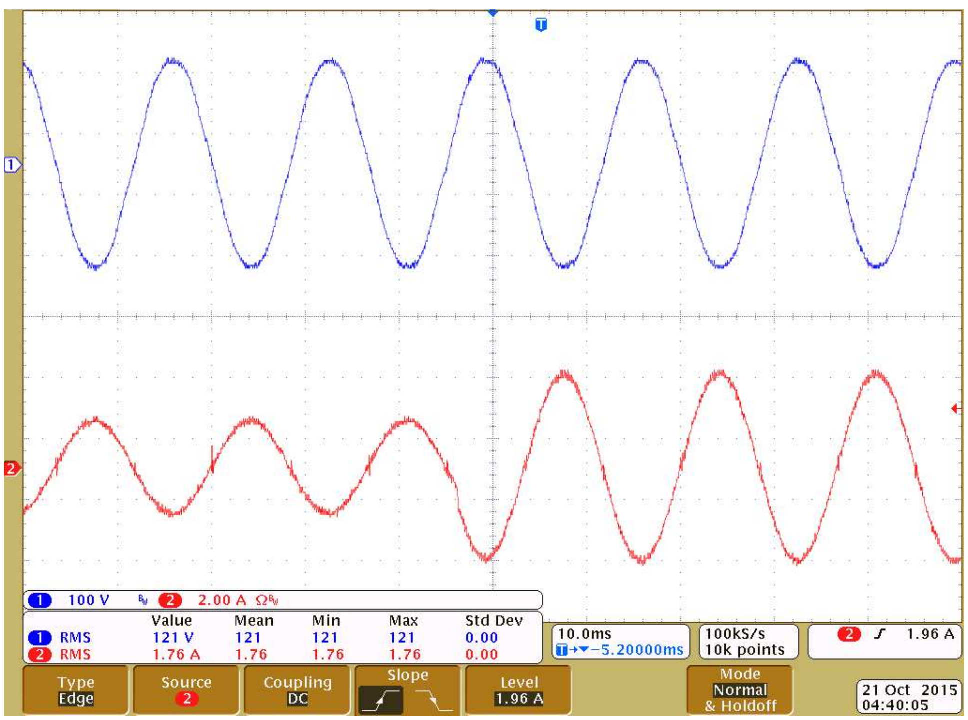
Figure 38. Steady-State Output Voltage and Current at 120 V_{RMS}, P_{OUT} of 160 W, Zoomed In



3.2.1.3 *Transient Waveform With Step Change in Load*

Figure 39 shows the transient output current as the current command is changed.

Figure 39. Transient Current Reference Change at 120 V_{RMS}

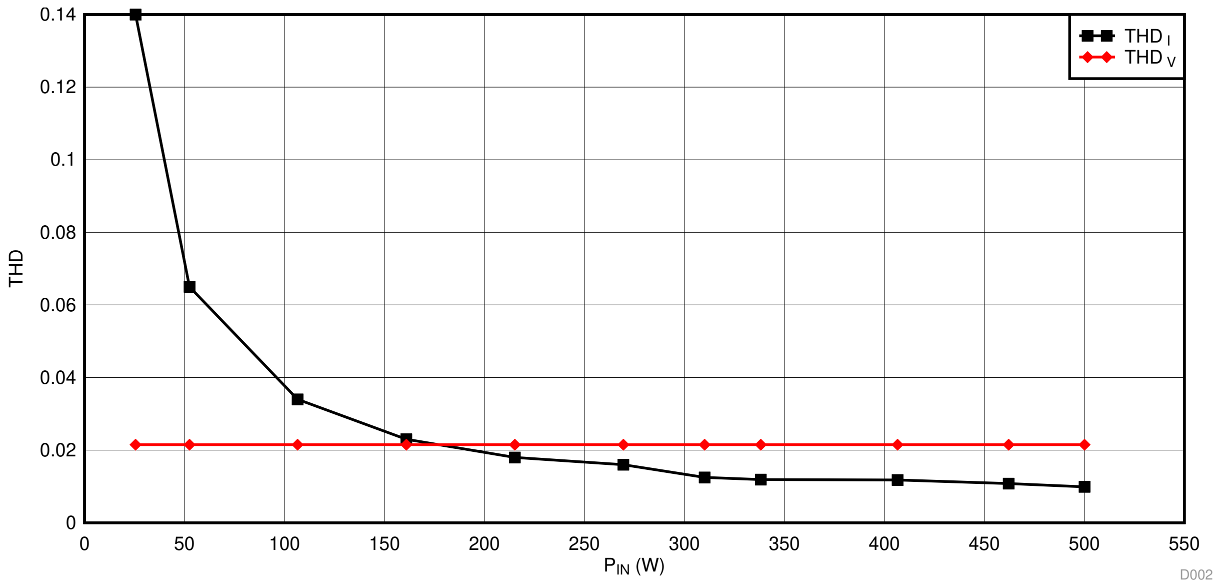


3.2.1.4 Total Harmonic Distortion Using SDFM

Figure 40 shows the output current total harmonic distortion across the power level while operating at 120 V_{RMS}, compared with the voltage distortion of the grid present at test time.

NOTE: The THD of the current is lower than the THD of the voltage at power level greater than 30% rated load.

Figure 40. THD of Output Current



The harmonic controllers are added one by one in the current compensator and the THD of the current plotted in Figure 41 for each combination. Table 5 lists the complete test data. Observe that the first, third, fifth, and seventh harmonic controllers affect the THD significantly; however, with the addition of the ninth harmonic controller the THD improves marginally.

Figure 41. THD at 120 V_{RMS} With Different Resonant Controller Combination

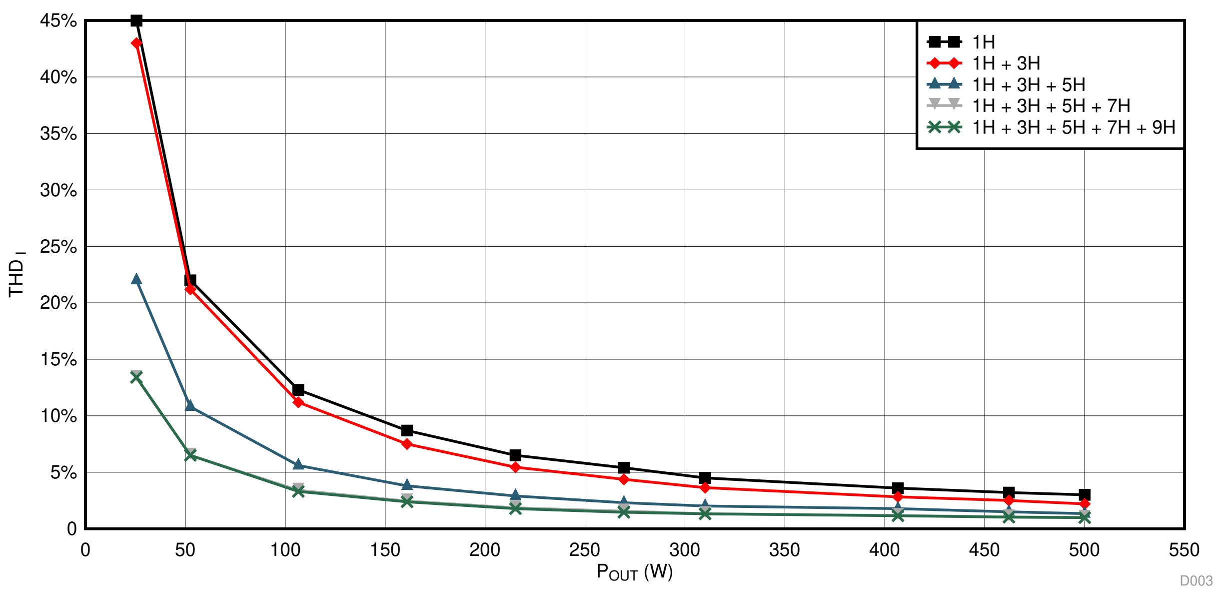


Table 5. THD With Different Resonant Controller Combinations

P_{OUT}	THDi WITH 1H	THDi WITH 1H AND 3H	THDi WITH 1H, 3H AND 5H	THDi WITH 1H, 3H, 5H, AND 7H	THDi WITH 1H, 3H, 5H, 7H, AND 9H
25.524	45	43	22	13.4	13.4
52.48	22	21.2	10.8	6.5	6.5
106.54	12.3	11.2	5.6	3.4	3.3
160.9	8.7	7.5	3.8	2.45	2.38
215.18	6.5	5.45	2.9	1.85	1.78
269.44	5.4	4.37	2.3	1.56	1.46
310.07	4.5	3.64	2.01	1.33	1.32
406.59	3.6	2.82	1.78	1.17	1.15
462.12	3.2	2.5	1.5	1.05	1.02
500.03	3	2.2	1.34	0.99	0.98

3.2.1.5 Test Data Table With SDFM-Based Sensing

Table 6 lists the complete test data of the design when operating with 110 V_{RMS} of output.

Table 6. Test Data Table for SDFM-Based Sensing and Output Current Control at 110 V_{RMS}

V_{IN}	I_{IN}	V_{OUT}	I_{OUT}	P_{IN}	P_{OUT}	THDi	EFFICIENCY	PF	THDv
382.8	0.071	122.27	0.2162	27.1788	25.524	14	0.93911431	0.96	2.01
382.8	0.142	122.41	0.4325	54.3576	52.48	6.5	0.965458372	0.9917	2.01
382.8	0.287	122.53	0.8714	109.8636	106.54	3.4	0.969747942	0.9977	2.01
382.6	0.433	122.72	1.3118	165.6658	160.9	2.3	0.971232445	0.9987	2.01
382.8	0.581	122.93	1.7522	222.4068	215.18	1.78	0.967506389	0.9989	2.01
382.6	0.729	122.99	2.1929	278.9154	269.44	1.46	0.966027692	0.999	2.01
382.8	0.841	122.98	2.5229	321.9348	310.07	1.3	0.963145333	0.999	2.01
382.8	0.919	123.36	2.7427	351.7932	338.15	1.2	0.961218125	0.999	2.01
382.8	1.111	123.55	3.2926	425.2908	406.59	1.18	0.956028205	0.999	2.01
382.8	1.265	123.86	3.7325	484.242	462.12	1.08	0.95431623	0.999	2.01
382.8	1.377	123.55	4.0561	527.1156	500.03	0.99	0.948615446	0.999	2.01

4 Design Files

For schematics, bill of materials (BOM), the altium project, and Gerber files, see http://www.ti.com/tool/TIDM-HV-1PH-DCAC or to the DigitalPower SDK package at:

<sdk_install_path>/solutions/tidm_hv_1ph_dcac

5 Software Files

To download the software files for this reference design, see the link at http://www.ti.com/tool/C2000WARE-DIGITALPOWER-SDK. This reference design can also be found at:

<sdk_install_path>/solutions/tidm_hv_1ph_dcac

\docs → Documentation

\hardware → PCB Altium Project, Gerbers, BOM, sense_calculation.xlsx

\<device> <control_mode> → CCS Project

6 Related Documentation

- Department of Energy Technology, *Design and Control of an Inverter for Photovoltaic Applications*, PHD Theses, Aalborg University, 2009
- IEEE, *Filter Design for grid connected PV inverters in ICSET*, 2008.
- IET Power Electronics, *Frequency Tracking of Digital Resonant Filters for Control of Power Converter Connected to Public Distribution Systems*, vol. 4, no. 4, pp. 454-462, Apr 2011.
- IEEE, *A New Single Phase PLL Structure Based on Second Order Generalized Integrator*, 37th IEEE PESC, 2006.
- Texas Instruments, *C2000™ Software Frequency Response Analyzer (SFRA) Library and Compensation Designer User's Guide*
- Texas Instruments, *TMS320F2837xD Dual-Core Delfino™ Microcontrollers Technical Reference Manual*
- Texas Instruments, *TMS320F2837xD Dual-Core Delfino™ Microcontrollers Data Sheet*
- Texas Instruments, *TMS320F28004x Piccolo™ Microcontrollers Technical Reference Manual*
- Texas Instruments, *TMS320F28004x Piccolo™ Microcontrollers Data Sheet*

6.1 Trademarks

C2000, E2E, powerSUITE, Code Composer Studio, Delfino, Piccolo are trademarks of Texas Instruments.
Excel is a registered trademark of Microsoft Corporation.
All other trademarks are the property of their respective owners.

7 About the Author

MANISH BHARDWAJ is a systems application engineer with the C2000 Microcontrollers System Solutions Group at Texas Instruments, where he is responsible for developing reference design solutions for digital power, motor control and solar power applications. Before joining TI in 2009, Manish received his masters of science in electrical and computer engineering from Georgia Institute of Technology, Atlanta and bachelor of engineering from Netaji Subhash Institute of Technology, University of Delhi, India.

Revision History

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

Changes from C Revision (May 2019) to D Revision — **Page**

- Changed powerSUITE Page for the Grid Connected Inverter Solution image ... 22
- Changed Project Explorer View of Solution Project image ... 23
- Changed Current Loop Design Using Compensation Designer image ... 32
- Changed Compensation Designer With Measured Plant Frequency Response Data image ... 36
- Changed SPRUHZ5 to SPRUIK4 link ... 47

Changes from B Revision (November 2017) to C Revision — **Page**

- Changed TMS310F280049M to TMS310F280049C ... 1
- Added and TMS320F280049C ... 1
- Changed TMS310F280049M from block diagram to TMS320F280049C ... 1
- Deleted TMS310F280049M from block diagram ... 5
- Changed TMS320F280049M to TMS320F280049C ... 19
- Changed <sdk_install_path>\c2000ware\boards\controlcards\TMDSCNCD280049M to <sdk_install_path>\c2000ware\boards\controlcards\TMDSCNCD280049C ... 19
- Changed TMS320F280049M to TMS320F280049C ... 19
- Changed TMS320F280049M to TMS320F280049C ... 21

Changes from A Revision (October 2016) to B Revision — **Page**

- Updated formatting to fit current design guide template ... 1
- Added TMS320F280049M to *Resources* ... 1
- Added "Supports TMS320F28377D and TMS320F280049M" to *Features* ... 1
- Added TMS320F280049M to the block diagram ... 1
- Changed location of Excel sheet of calculations ... 14
- Changed location of info sheet in Section 3.1.1.2: *Control Card Settings* ... 19
- Added required settings for revision A of the TMS320F280049M control card in Section 3.1.1.2: *Control Card Settings* 19
- Added note in Section 3.1.2: *Firmware: powerSUITE and Incremental Build Software* ... 21
- Changed Step 5 in Section 3.1.2.1: *Opening the Project Inside Code Composer Studio™* ... 21
- Updated Figure 21: *Build Level 1 Control Diagram: Open Loop Project* ... 24
- Changed 1.26 V_{RMS} to 1.26 I_{RMS} in Step 13 in Section 3.1.2.3.1.5: *Running the Code* ... 28
- Updated Figure 26: *Build Level 2 Control Diagram: Closed Current Loop* ... 30
- Changed location of the SFRA.exe file from controlSUITE/libs/app_libs/SFRA/<version>/GUI to <sdk_install_path>/libraries/sfra/GUI ... 35
- Changed invloRefInst to invloRef in Step 46 in Section 3.1.2.3.2.4: *Running the Code* ... 36
- Deleted the step "Set the *clearInvTrip* variable to 1 to clear the inverter trip" in Section 3.1.2.3.3.2: *Running the Code* . 39
- Added the *TMS320F28004x Piccolo™ Microcontrollers Technical Reference Manual* to Section 6: *Related Documentation* ... 47
- Added the *TMS320F28004x Piccolo™ Microcontrollers Data Sheet* to Section 6: *Related Documentation* ... 47

Changes from Original (November 2015) to A Revision — **Page**

- Added TIEVM-HV-1PH-DCAC product page link in *Design Resources* ... 1
- Added configuration (CFG) ... 7
- Added Figure 12: *Control Scheme Used for Grid Connected Inverter Control* ... 12
- Added note in Section 3: *Getting Started With Hardware* ... 16
- Deleted "Keep in mind that the output capacitor is large (for example, 20 uF)." from Step 2 in Section 3.1: *Base Board*