# Digitally Controlled Solar Micro Inverter Design using C2000 Piccolo Microcontroller

# User's Guide

Literature Number: TIDU405B
October 2014–Revised June 2017

# Digitally Controlled Solar Micro Inverter using C2000™ Piccolo Microcontroller

This document presents the implementation details of a digitally-controlled solar micro inverter using the C2000 microcontroller. A 250-W isolated micro inverter design presents all the necessary PV inverter functions using the Piccolo-B (F28035) control card. This document describes the power stages on the micro inverter board, as well as an incremental build level system that builds the software by verifying open loop operation and closed loop operation. This guide describes control structures and algorithms for controlling power flow, maximizing power from the PV panel (MPPT), and locking to the grid using phase locked loop (PLL), along with hardware details of Texas Instruments Solar Micro Inverter Kit (TMDSOLARUINVKIT).

**NOTE:** The micro inverter board design follows a control card concept; therefore, a different control card can be used depending on the system requirements.

**CAUTION**

The TI Solar Micro Inverter board produces high voltages and should only be handled by experienced power supply professionals in a lab environment. Power may also produce high temperatures in some components; take appropriate safety measures before working with this board.

# 1 Introduction

Energy from renewable sources, such as solar and wind, is gaining interest as the world's power demand increases and non-renewable resources are depleted. A large component of this demand is from industries and houses connected to the electrical grid. Because of this, attempts are made to raise the percentage of energy sourced from renewable sources into the grid. Photovoltaic (PV) energy sources increase the renewable content because of their ubiquitous nature and extended life time due to an absence of moving parts.

The PV panel is a non-linear DC source; an inverter must feed current into the grid, and a maximum power tracking algorithm must maximize power from the panel. Therefore the key challenge in any PV inverter system design is to feed a clean current into the grid while maintaining the maximum power point of the panel. A typical PV grid-tied inverter consists of a string of PV panels connected to a single inverter stage; these are called string inverters. This PV inverter architecture, however, suffers from partial shading problems. An emerging architecture includes an inverter on each panel, as seen in Figure 1. The localized MPPT at each panel improves the performance of the system under partial shading and unmatched panel conditions. The Texas Instruments C2000 microcontroller family, with its enhanced peripheral set and optimized CPU core for control tasks, is ideal for controlling the power conversion.

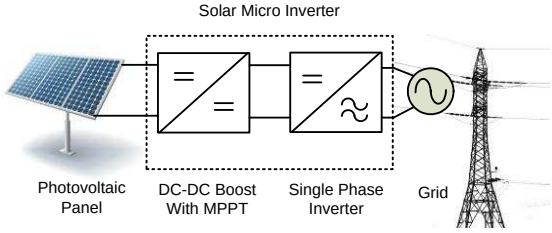



**Figure 1. Grid Tied PV Inverter**

This user guide presents an overview of the hardware and the detailed software implementation of a PV micro inverter system, using the C2000 MCU on Texas Instrument's solar micro inverter kit (TMDSSOLARUINVKIT). All of the key features needed in PV inverter applications such as MPPT, closed loop current control of inverter, and grid synchronization are implemented on the kit using the TMS320F28035 Micro Controller.

The TMDSSOLARUINVKIT hardware consists of two stages: (1) an active clamp flyback converter with a secondary voltage multiplier and (2) a DC-AC inverter. A block diagram of the kits is shown in Figure 2.

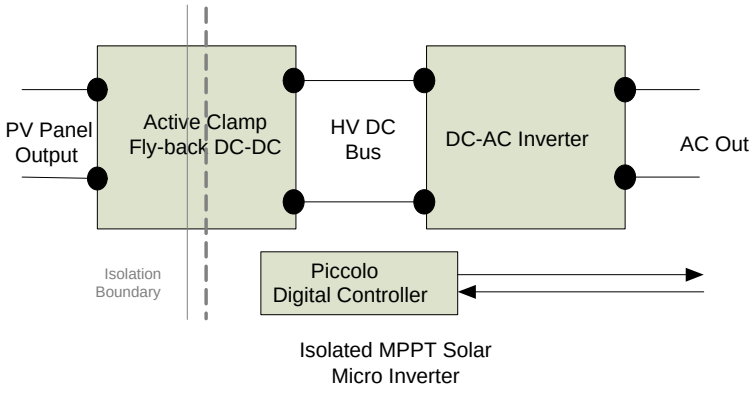



**Figure 2. Micro Inverter Block Diagram**

The DC-DC converter draws DC current from the PV panel such that the panel operates at its maximum power transfer point. This requires maintaining the panel output, such as the DC-DC converter input, at a level determined by the MPPT algorithm. In this implementation, the MPPT algorithm determines the panel output current (reference current) for maximum power transfer. A current control loop for the flyback converter then ensures that the converter input current tracks the MPPT reference current. The flyback converter also provides high frequency isolation for the DC-DC stage. The output of the flyback stage is a high voltage DC bus, which drives the DC-AC inverter. The inverter stage maintains the DC bus at a desired set point and injects a controlled sine wave current into the grid. The inverter also implements grid synchronization to maintain its current waveform locked to the phase and frequency of the grid voltage. Figure 3 illustrates the control scheme for a complete grid connected to a PV micro inverter. All of the key functions are implemented on the F28035 MCU for the Solar Micro Inverter kit.

A C2000 piccolo microcontroller with its on-chip PWM, ADC, and analog comparator modules can implement complete digital control of a micro inverter system. Figure 4 shows a simplified diagram of different stages present on the Solar Micro Inverter kit.

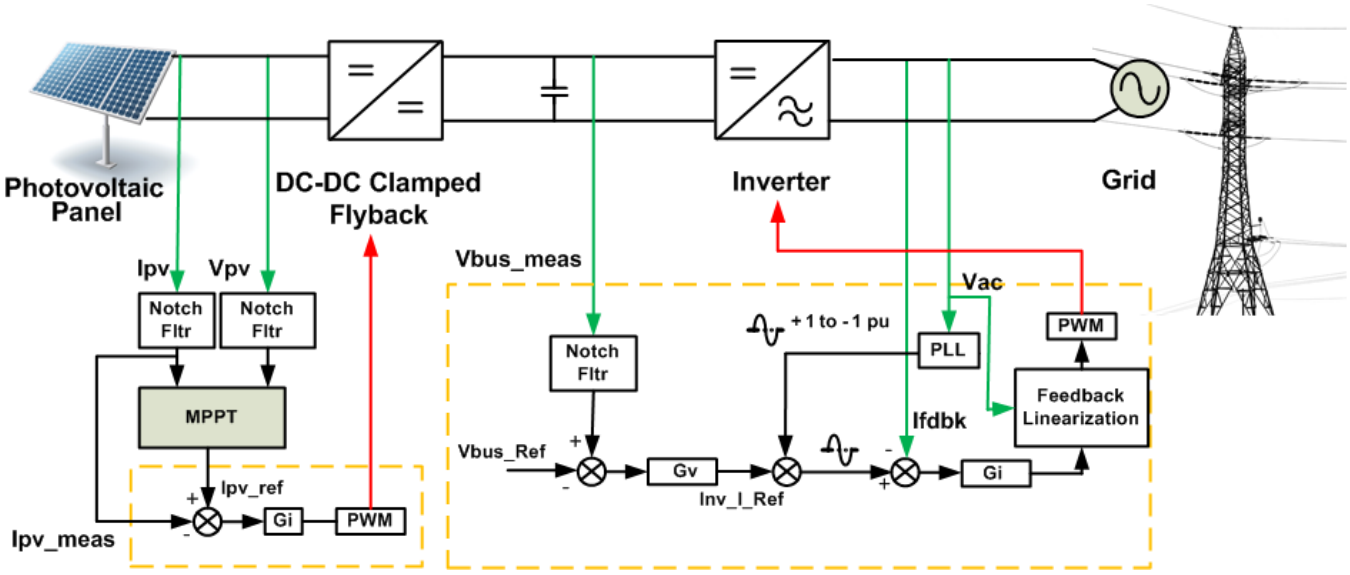



**Figure 3. Control of Grid-Connected Solar Micro Inverter**

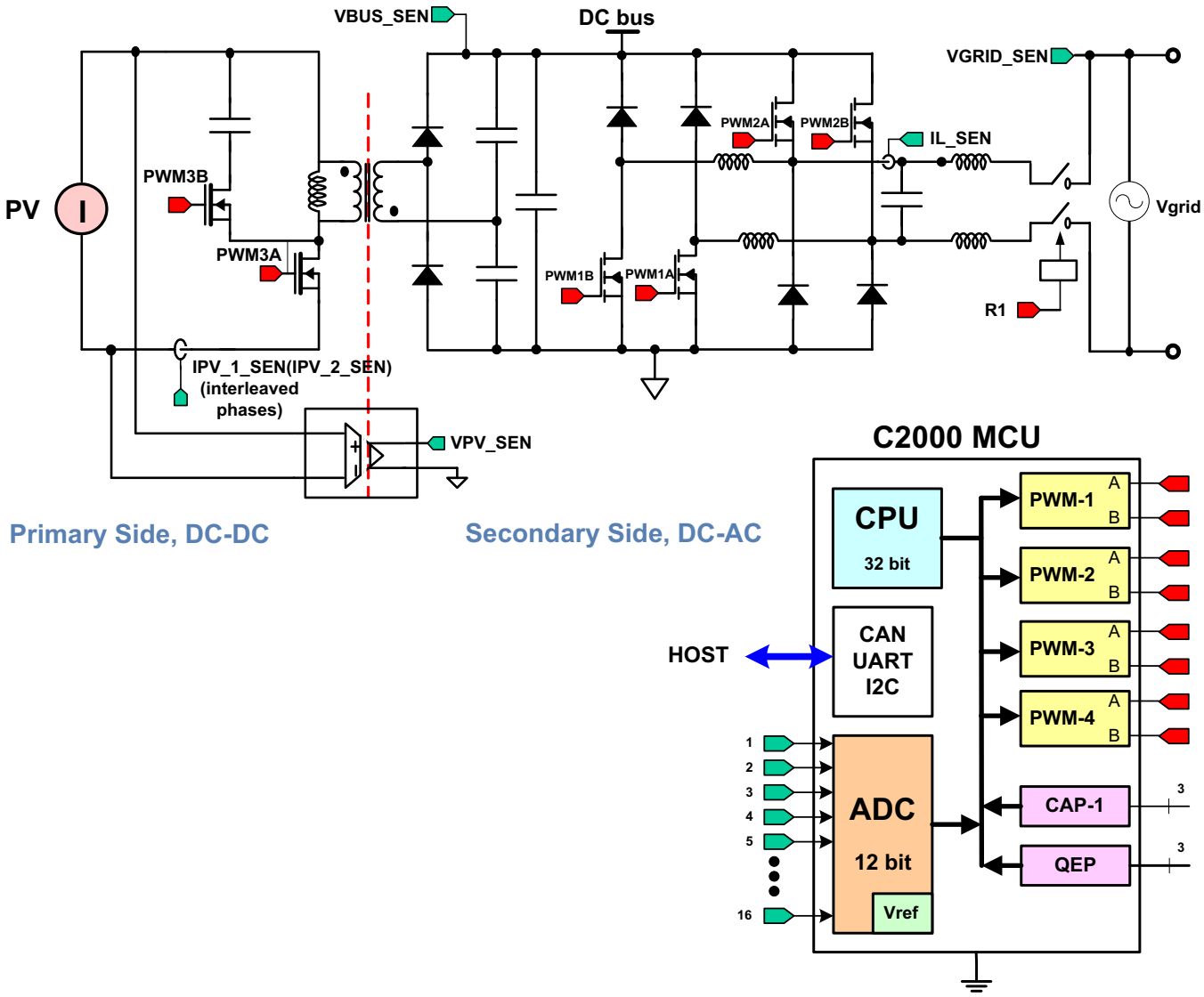



**Figure 4. Solar Micro Inverter Kit Power Stage Diagram**

## 2 Hardware and Control

This section describes the hardware and control scheme implemented in the Solar Micro Inverter kit.

### *2.1 Kit Contents*

The kit contents include:

- TMDSSOLARUINVKIT base board
- TMDSCNCD28035ISO control card
- USB mini to A cable
- 15-V DC external isolated bias supply
- 12-V DC external isolated bias supply
- AC cord

The following sections describe the power stages present on the kit.

## 2.2 Clamped Fly-back DC-DC Stage

Figure 5 illustrates the MPPT active clamp flyback DC-DC power stage implemented on the micro inverter kit.

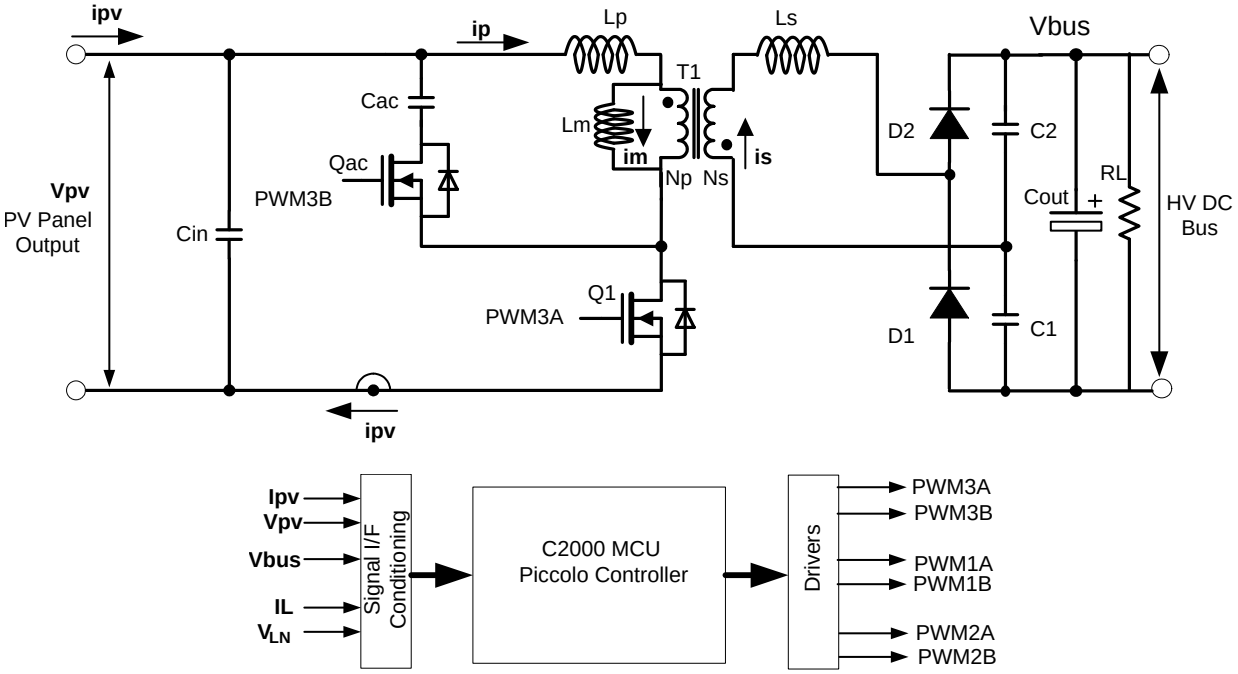



**Figure 5. MPPT Fly-Back DC-DC Converter Control using C2000 MCU**

The PV panel output voltage, Vpv, is applied to the active clamp fly-back stage input. Transformer T1, MOSFET Q1, diode D2, and capacitor C2 together form the conventional flyback stage. MOSFET Qac and capacitor Cac form the primary side clamping circuit. The capacitor C1 with diode D1 provide a voltage multiplier circuit at the flyback converter output. This multiplier circuit operates in a forward converter mode to transfer energy from the input to the output. Figure 5 also indicates every interface signal needed for full control of the DC-DC converter using a C2000 micro-controller (MCU). The MCU controls the hardware using three feedback signals and two PWM outputs, PWM3A and PWM3B. The feedback signals include the panel output voltage (Vpv), the flyback input current (Ipv), and the flyback output voltage (Vbus). Because the MCU is ground referenced to secondary ground (-ve terminal of high voltage DC bus Vbus), panel output voltage Vpv is sensed with a differential isolation amplifier (AMC1200S). Ipv is sensed with a current transformer. As a second option, Ipv is also sensed with an isolated current sensor such as ACS712ELCTR-20A. The sensed signals Ipv and Vpv are used to implement the MPPT algorithm and the current control loop for the DC-DC fly-back stage. The active clamp DC-DC fly-back is chosen to boost the low panel output voltage to a high-voltage DC bus. This high-voltage DC bus is needed for injecting current into the grid with a worldwide voltage range of 90 Vrm to 260 Vrms.

Figure 6 shows the DC-DC converter control loop, using a single current control loop. A Maximum Power Point Tracking (MPPT) algorithm determines the set point current, such as the reference panel current Ipv_ref. The current control loop ensures that the DC/DC input current is regulated at the MPPT reference level by adjusting the duty cycles of the power switches Q1 and Qac.

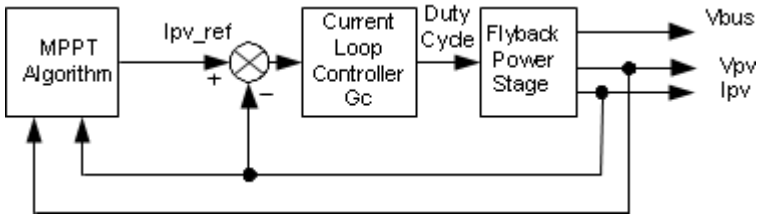



**Figure 6. MPPT DC-DC Converter Control Loop**

The panel current Ipv, sensed through one of the ADC channels, is compared against the reference current Ipv_ref set by the MPPT algorithm. The resulting error signal is then input to the current loop controller Gc, which regulates the panel current at the reference level by generating the flyback converter PWM duty ratio command d for the switches Q1 and Qac. The current controller Gc has the form of a two-pole two zero (2P2Z) compensator.

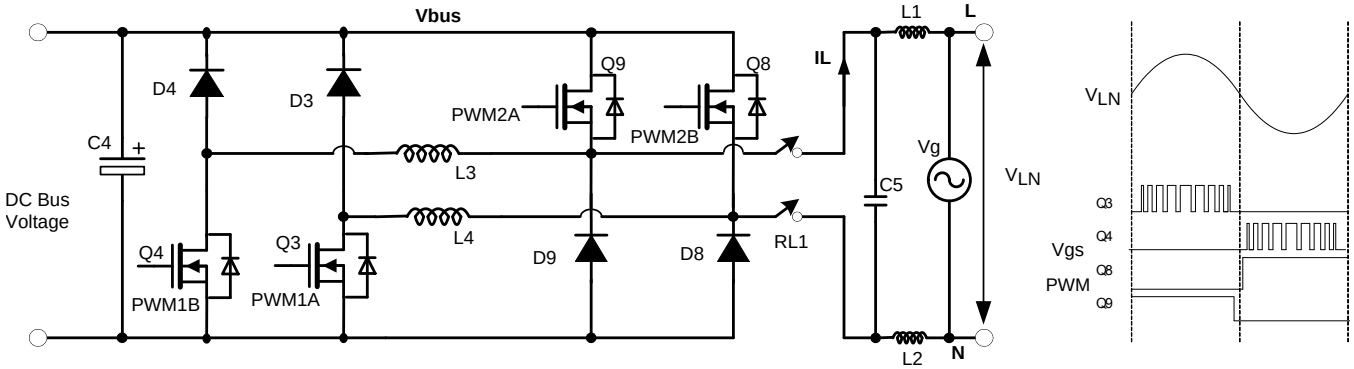
In addition to implementing the current loop controller, the C2000 MCU also monitors the boost output voltage for over-voltage protection.

Every time critical function related to the DC-DC control loops is implemented in a fast sampling loop enabled by the C2000 Micro-controller high speed CPU, interrupts, on chip 12-bit ADC module, and high frequency PWM modules. A detailed description of the software algorithm is provided in the following chapters.

## 2.3 DC-AC Inverter Stage

Figure 7 illustrates the DC-AC inverter control system using the C2000 MCU. The DC-DC output voltage, Vbus, is applied to the inverter stage input.

The inverter output connects to the grid. The inverter is controlled as a current source and consists of two DC-AC buck converters, each operating in one of the half-cycles of the AC line voltage VLN. MOSFETs Q9, Q3, diode D3, and inductor L4 together form the first buck converter when VLN is positive. MOSFETs Q8, Q4, diode D4, and inductor L3 together form the second buck converter when VLN is negative. In each line half-cycle, the corresponding upper switch, Q9 or Q8, stays fully on. The lower switches, Q3 and Q4, operate at a high PWM frequency (50-kHz PWM) as the buck converter switches in their respective line half-cycle. PWM1A and PWM1B control Q3 and Q4 respectively. The upper switches Q9 and Q8 are controlled by PWM2A and PWM2B respectively. This inverter PWM scheme implementation is detailed in Section 3.4. The main inductor L3 and L4, the capacitor C5, and a second pair of small inductors, L1 and L2, together form an LCL filter at the inverter output. The double pole single through (DPST) relay RL1 is controlled by the MCU, and connects or disconnects the inverter to and from the AC mains.



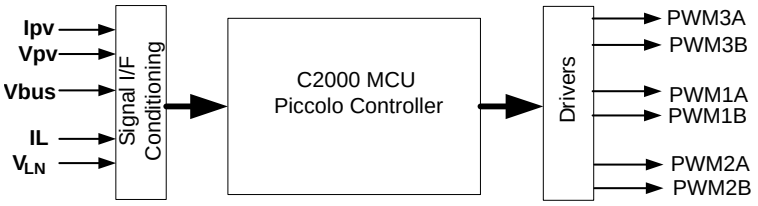



**Figure 7. DC-AC Inverter Control Using C2000 MCU**

Figure 7 indicates the interface signals needed to control the inverter using the C2000 MCU. The MCU controls the hardware, using three feedback signals and four PWM outputs. The signals sensed and fed back to the MCU include the AC line voltage (VLN), the DC bus voltage Vbus, and the main inductor current IL. The sensed signals implement the Vbus control loop, the current control loop, and the PLL control loop for the Inverter. These control loops are shown in Figure 8.

The voltage control loop employs a two pole two zero (2P2Z) compensator and regulates the input DC bus voltage at the reference level Vbus_ref by generating the reference current command for the current loop. The current loop forces the inductor current IL to track the reference signal IL_ref generated by the voltage loop. The current controller has the form of a 3P3Z compensator. The PLL controller allows for locking the phase and frequency of the inductor current to the grid voltage VLN under all conditions. The grid voltage acts as a disturbance for the current controller, and a 2P2Z compensator cannot track to zero steady state error for this disturbance. Therefore, a feedback linearization technique is used in current control loop to calculate the PWM duty ratios for the inverter switches Q3 and Q4 (see Section 2.3.2 for details). This linearization technique assumes that the current loop gain is high at the vicinity of a grid frequency of 47 Hz to 65 Hz. This assumption is later validated by measuring current control loop gain as shown in Figure 9, which shows both the designed (calculated) and measured current loop gain plot. As shown in the plots, the current loop bandwidth is around 1.2 kHz and the gain at 200 Hz or below is more than 20 dB. A detailed description of the software algorithm is provided in the following chapters.

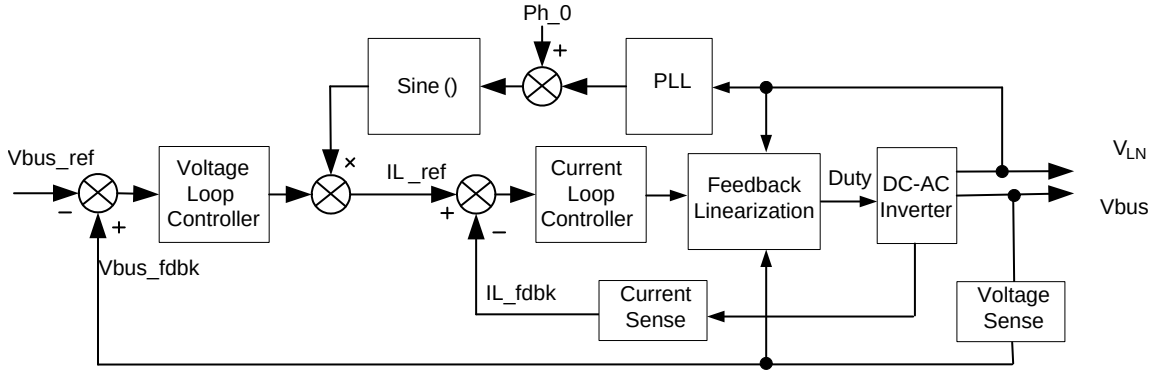



**Figure 8. DC-AC Inverter Control Loop**

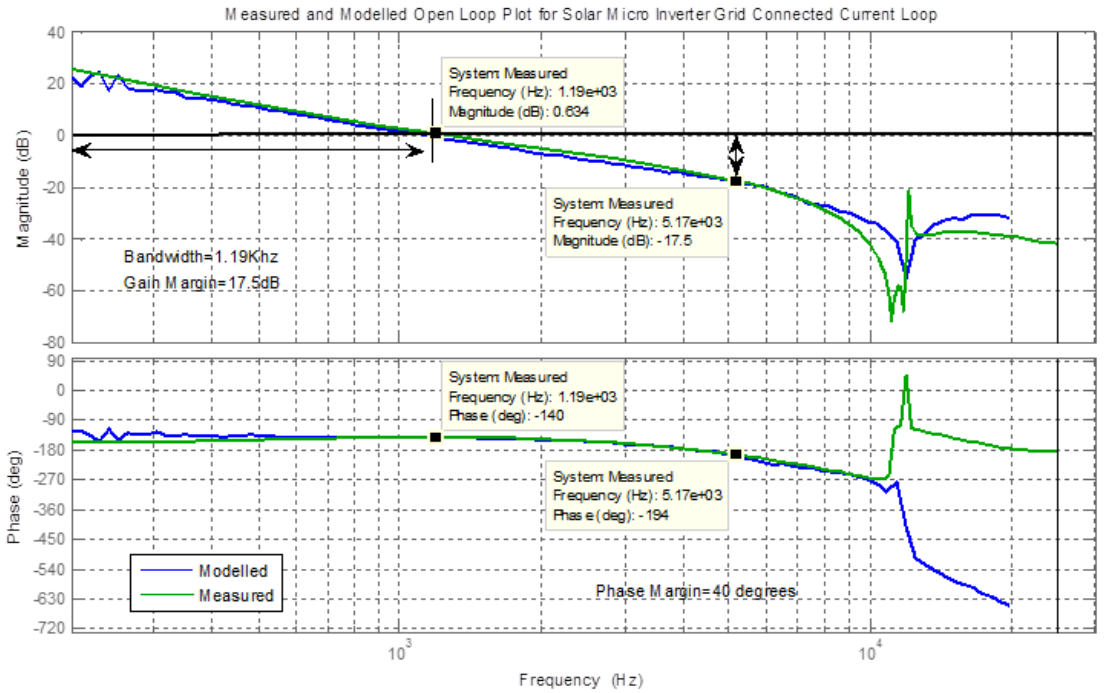



**Figure 9. DC-AC Inverter Current Loop Bode Plot**

### 2.3.1 Inverter Phase Locked Loop

The inverter uses a phase locked loop (PLL) to synchronize its output current to the grid voltage. A functional diagram of a PLL is shown in Figure 10, which consists of a phase detect (PD) consisting of park transform, a loop filter (LPF), and a voltage controlled oscillator(VCO).

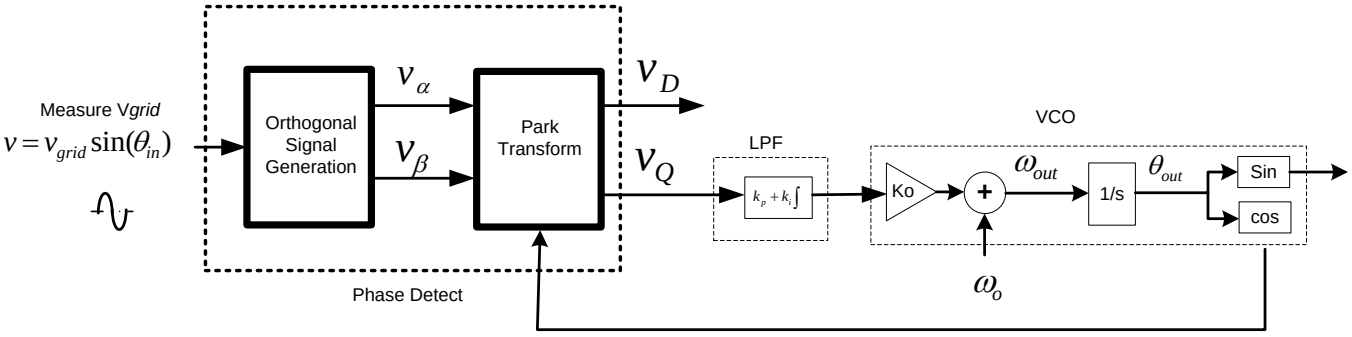



**Figure 10. Inverter PLL Control**

The phase detect (PD) block detects the phase error by generating an orthogonal signal and taking park transform. The orthogonal signal generation is done using a second order generalized integrator (as proposed in *A New Single Phase PLL Structure Based on Second Order Generalized Integrator*, Mihai Ciobotaru, PESC'06). This method can selectively tune the orthogonal signal generator to reject all frequencies other than the grid frequency. Assuming an arbitrary input signal and PLL theta, the phase detect (PD) output is given by the following equation:

$$PD_{output} = v_{in}\begin{bmatrix}\cos(\theta_{in})\\ \sin(\theta_{in})\end{bmatrix}\times\begin{bmatrix}\cos(\theta_{out}) & \sin(\theta_{out})\\ -\sin(\theta_{out}) & \cos(\theta_{out})\end{bmatrix} = v_{in}\begin{bmatrix}\cos(\theta_{in}-\theta_{out})\\ \sin(\theta_{in}-\theta_{out})\end{bmatrix} = \begin{bmatrix}v_d\\ v_q\end{bmatrix} \quad (1)$$

Now assuming PLL is closed to being locked, such that $\theta_{in}-\theta_{out} \approx 0 \rightarrow \sin(\theta_{in}-\theta_{out}) \approx \theta_{in}-\theta_{out}$, therefore is the error in the PLL angle lock.

The loop filter is implemented using a PI controller to keep this error as zero:

$$\frac{ylf(s)}{PhaseDetect(s)} = k_p + \frac{k_p}{T_i} \times \frac{1}{s} = K_p + \frac{K_i}{s} \tag{2}$$

Using control theory, the closed loop transfer function of the PLL structure can be given as

$$H_o(s) = \frac{\theta_{out}(s)}{\theta_{in}(s)} = \frac{LPF(s) \times 1/s}{1 + LPF(s) \times 1/s} = \frac{\left(k_p s + \frac{k_p}{T_i}\right)}{s^2 + k_p s + \frac{k_p}{T_i}} \tag{3}$$

Comparing the above closed loop transfer function equation to a second order system transfer function:

$$H(s) = \frac{2\zeta\omega_n s + \omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \tag{4}$$

The natural frequency and the damping ration of the PLL is given by:

$$\omega_n = \sqrt{\frac{k_p}{T_i}} \qquad \zeta = \sqrt{\frac{T_i k_p}{4}} \tag{5}$$

Ignoring the LHP zero from the closed loop transfer equation, step response to a general second order equation, that is

$$H(s) = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \tag{6}$$

is given as:

$$y(t) = 1 - ce^{-\sigma t}\sin(\omega_d t + \phi) \tag{7}$$

Where the settling time is given as the time it takes for the response to settle between an error band. If, say, this error is $\theta$, then

$$1 - \partial = 1 - ce^{-\sigma t_s} \Rightarrow \partial = ce^{-\sigma t_s} \Rightarrow t_s = \frac{1}{\sigma} \times \ln\left(\frac{c}{\partial}\right)$$

Where $\sigma = \varsigma\omega_n$ and $c = \frac{\omega_n}{\omega_d}$ and $\omega_d = \sqrt{1-\varsigma^2}\,\omega_n$

Where

- $\sigma = \zeta\omega_n$
- $c = \omega_n/\omega_d$
- and $\omega_d = \sqrt{1-\zeta^2}\omega_n$ (8)

Now using a settling time of 30 ms and a 5% error band and a damping ratio of 0.7, the natural frequency would be 119.014. Back substituting, the user gets $K_p$ = 166.6 and $K_i$ = 27755.55.

In the digital domain the loop filter is implemented, according to the following discrete equation:

$$ylf[n] = ylf[n-1] \times A1 + PhaseDetect_q[n] \times B0 + PhaseDetect_q[n-1] \times B1 \tag{9}$$

Using Z transform this equation can be re-written as

$$\frac{ylf(z)}{PhaseDetect(z)} = \frac{B0 + B1 \times z^{-1}}{1 - z^{-1}} \tag{10}$$

Now using Bi-linear transformation on the LPF transfer function, replace $s = \frac{2}{T}\left(\frac{z-1}{z+1}\right)$ where T= Sampling Time:

$$\frac{ylf(z)}{PhaseDetect(z)} = \frac{\left(\frac{2\times K_p + K_i \times T}{2}\right) - \left(\frac{2\times K_p - K_i \times T}{2}\right) z^{-1}}{1 - z^{-1}} \quad (11)$$

Equation 10 and Equation 11 can be compared to map the proportional and integral gain of the PI controller from the analog domain into the digital domain.

$$B0 = \left(\frac{2\times K_p + K_i \times T}{2}\right) \text{ and } \left(B1 = \frac{2\times K_p - K_i \times T}{2}\right) \quad (12)$$

For example, on the micro inverter kit the ISR rate is 50 Khz, and therefore B0 = 166.877556 and B1 = -166.322444 are used for the loop filter.

### 2.3.2 Inverter Current Control Loop

The inverter has an LCL filter at its output as shown in Figure 11. Here $V_i$ is the inverter output voltage and $V_g$ is the grid voltage. While controlling the inverter current IL, the small signal gain for inverter current to duty ratio is,

$$\frac{\tilde{i}_L}{\tilde{d}} = \frac{V_{bus}\left(Z_c + Z_g\right)}{Z_g \times Z_c + \left(Z_c + Z_g\right) \times Z_i} \quad (13)$$

This assumes that the change in inductor current $i_L$ has no effect on grid voltage $V_g$. that is, this is a stiff grid system.

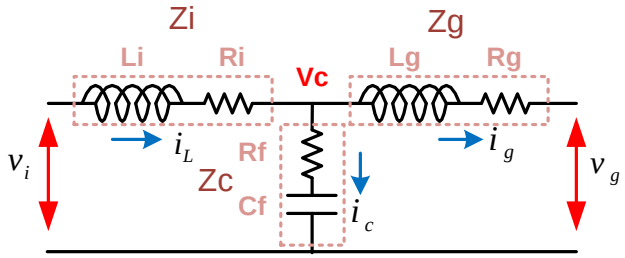



**Figure 11. Inverter LCL Filter**

The inverter current loop power stage control model is represented by the diagram in Figure 12. Here $i_L$ is the current in the main inductor (inverter side inductor). Also, it is assumed that the current $i_C$ that flows through the filter capacitor Cf is negligible compared to the inductor current $i_L$.

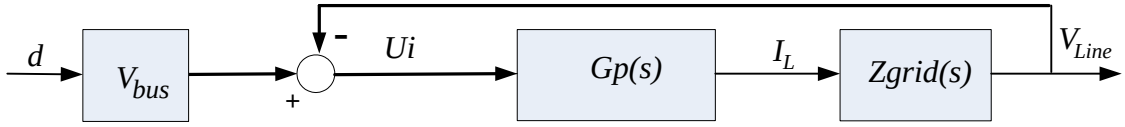



**Figure 12. Inverter Current Loop Power Stage Control**

The current loop stage diagram in Figure 12 shows that calculating $I_L$ from d requires the grid voltage $V_{Line}$ which is again related to $I_L$ by the unknown grid impedance $Z_{grid}$. To avoid this problem, a feedback linearization scheme is used under the assumption that the current loop controller is designed such that the resulting current loop gain at grid frequency is high. As a result, $V_{Line}$ can be treated as a DC parameter compared to the dynamics of inverter inductor current $I_L$, that is, the current loop crossover frequency is much higher than the grid frequency. With this assumption, the simplified current loop power stage diagram reduces to the one shown in Figure 13.

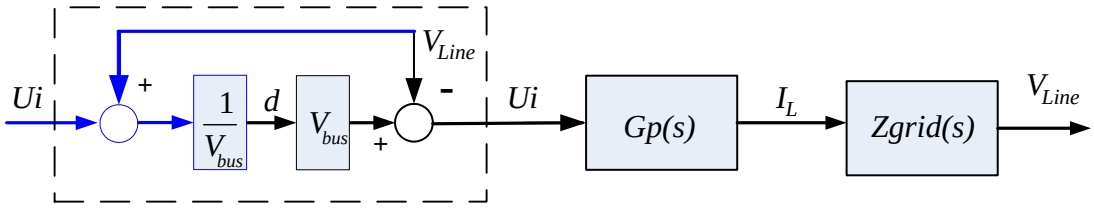



**Figure 13. Simplified Current Loop Power Stage Control**

The simplified diagram in Figure 13 shows that the current loop power stage input is Ui, which controls the inductor current $I_L$. This is valid when the input Ui is controlled by a current loop controller, such that the loop gain at grid frequency is high. The diagram also shows how the PWM duty ratio d is calculated from Ui using the measured values of $V_{bus}$ and $V_{Line}$. This can be done in firmware. Adding the current controller $G_c$, to the power stage diagram in Figure 13, the complete current control loop diagram becomes as the one shown in Figure 14.

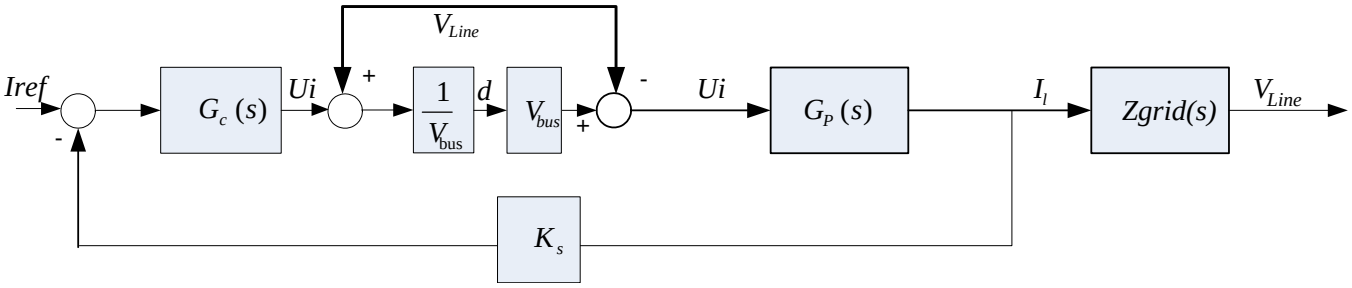



**Figure 14. Inverter Current Control Loop**

Where, Ks is the current feedback gain. Therefore, the loop gain transfer function of the control loop is:

$$G_{loop}(s) = G_c(s)\, K_s G_p \tag{14}$$

Using Matlab the current controller is designed as the following 3P3Z controller:

$$G_c(z) = \frac{0.2866 - 0.3173z^{-1} + 0.338z^{-2} - 0.2616z^{-3}}{1 - 1.584z^{-1} + 0.6978z^{-2} - 0.1137z^{-3}} \tag{15}$$

The loop gain Bode plot for the current loop is shown previously in Figure 9. The plot shows that the cutoff frequency of the current loop is around 1.19 kHz, and the system is stable with good phase margin (PM) and gain margin (GM).

### 2.4 Solar Micro Inverter Electrical Specifications

The following lists the key highlights of the micro inverter:

- Panel voltage: 28 V (min) to 45 V (max)
- DC bus voltage: 400 V (max)
- Output power: 280 W @ 220Vac, 140 W @ 110Vac
- DC-DC and DC-AC combined efficiency around 92% at 50% rated load, THD around 5%

**NOTE:** For 220-V/50-Hz grid operation, the EVM was only tested with an emulated grid, that is, an external AC source (at least 600-VA rating) with a load resistor (200.0-Ohm, 600-W rating) at the output. Therefore, EVM supplied power to the load and not back to the AC supply.

## 3 Software Implementation

This section describes the details of software implementation of control of PV micro inverter.

## 3.1 Project Framework

PV inverter control requires closed loop control of the DC-DC and DC-AC stage. PWM switching rates of the power stages are chosen such that only a single, fast 50-KHz ISR is needed for controlling the DC-DC flyback and the DC-AC inverter stage. A slower ISR (1 KHz) runs the state machine, MPPT, and power measurement functions. The key framework C files used in the project are:

**SolarMicroInv-Main.c —** This file initializes, runs, and manages the application. In addition, this file also contains ISR for inverter stage control, MPPT algorithm execution, data logging, and relay control.

**SolarMicroInv-DevInit_F2803x.c—** This file contains all the initialization routines and configuration of IOs and peripherals for this application. This file also includes functions such as setting up the clocks, PLL, and Watchdog. Most functions in this file are called once during the system initialization, from SolarMicroInv-Main.c.

**SolarMicroInv-Settings.h —** This file contains the settings for the incremental build option, and various defines for PWM frequency and ISR triggers used in the project framework.

**SolarMicroInv-Includes.h—** This file contains of all the header files used by the project.

Figure 15 gives the structure of the PV micro inverter software, with the main background loop, the fast ISR for the DC-DC and DC-AC control loop, and slower ISR for state machine control and MPPT.

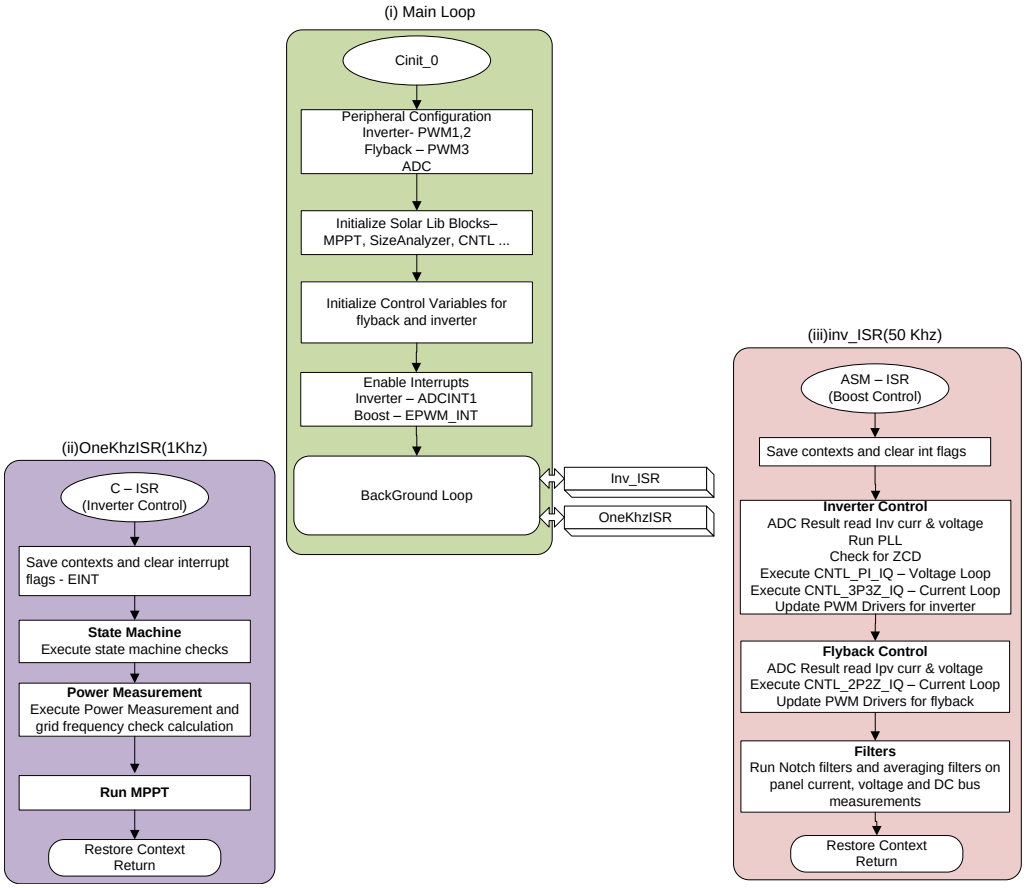



**Figure 15. PV Inverter Software Structure: (i) Main Loop (ii) Inverter ISR (iii) 1-KHz ISR**

## 3.2 Project Dependencies and Resources

**Table 1. Project Resources**

| | |
|---|---|
| Hardware Kit | TMDSSOLARMICROINVKIT |
| Control Card | F28035 ISO |
| Software IDE | CCSv5.5 or later |
| Code Generation Tool | 6.2.5 |
| GUI | Gui Composer Runtime v5.5 or later |

**Table 2. Control Suite Dependencies**

| | |
|---|---|
| Device Support (F28035 Header Files) | controlSUITE\device_support\f2803x\v126 |
| IQMath Library | controlSUITE\libs\math\IQmath\v160 |
| Solar Library | controlSUITE\app_libs\solar\v1.2\IQ |

## 3.3 Control Description

Figure 3 shows the control of a grid-tied PV inverter, which comprises of closed loop control of the boost and closed loop control of the inverter. The following sections describe the software flow for these two modules.

### 3.3.1 DC-DC Flyback with MPPT Control Software

To get the most energy out of the solar panel, the panel must be operated at its maximum power point. The maximum power point is not fixed and changes with temperature and light intensity. Different techniques are used to track maximum power point of the panel, like Perturb and Observe (PnO) or the incremental conductance (INCC) algorithm. To track the MPP, input voltage ($V_{pv}$) and input current ($I_{pv}$) are sensed on a periodic basis, and the power stage is controlled to regulate the input current value. The reference value of the input current is provided by the MPPT algorithm.

The panel current reference changes at a slow rate of around 10 Hz, and the power stage current regulation runs at 50 KHz. The control loop for the DC-DC stage does not control the boost stage output voltage. Boost output voltage is regulated by the DC-AC inverter, which modulates the current fed into the grid to keep this voltage regulated.

Figure 16 gives the software diagram for the DC-DC stage using the blocks from the solar library. Notch filter is used to reduce any errors in the MPPT algorithm due to the AC power ripple seen by the panel.

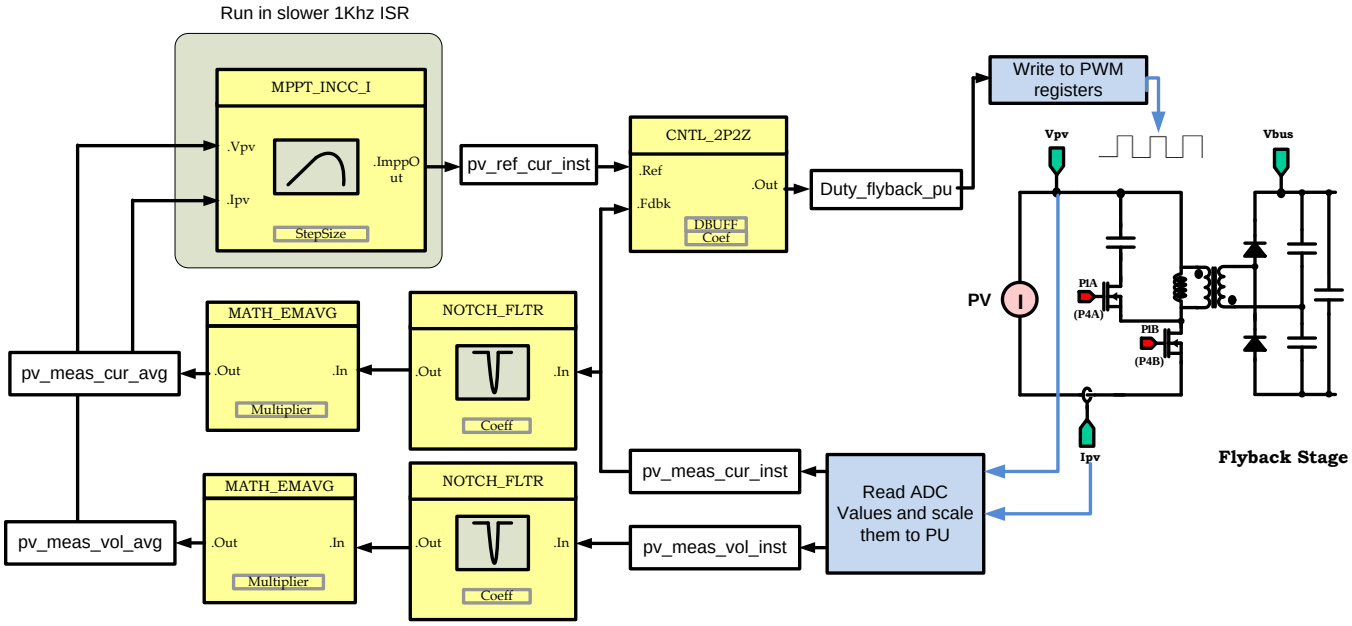



**Figure 16. DC-DC 1-ph Boost with MPPT Software Diagram**

### 3.3.2 DC-AC Single Phase Inverter Control Software

The inverter stage gets input from the DC-DC boost stage output. For the inverter, the current fed into the grid is given by the equation:

$$i_{grid} = \frac{V_{dc} \times D - v_{grid}}{Z_{LCL}}$$

where

D is the duty ratio of the high frequency PWM switches in the inverter power stage. (16)

For the inverter to feed current into the grid, $V_{dc}$ must always be greater than the max grid voltage. Also, the DC bus is not regulated by the DC-DC boost stage. Therefore the inverter stage software uses nested control loops: an outer voltage loop and an inner current loop. The voltage loop generates the reference current command for the current loop. In this case, increasing the current command increases the load on the inverter DC bus and causes a drop in the DC bus voltage. Therefore, the sign for reference and the feedback are reversed: the voltage error signal is calculated by subtracting the reference voltage signal from the DC bus feedback signal. The current command from the voltage loop output is then multiplied by the AC angle to get the instantaneous current reference. With grid-connected inverter software, PLL provides this grid angle. The instantaneous current reference is then used by the current compensator along with the feedback current to provide the duty ratio for the inverter switches. A notch filter removes the second harmonic (of the grid frequency) ripple from the DC bus measurement. This enables running the DC bus controller at much higher bandwidth, without affecting the current THD, as shown in Figure 17.

Additionally, to accommodate for the disturbance in the current injection due to grid voltage a feedforward term is introduced, for calculation of the duty cycle for the inverter. This is called feedback linearization and was described in Section 2.3.2.

$$D = \frac{\left(i_g^* - i_g\right) \times G_c(s) + v_{grid}}{V_{DC}} \quad (17)$$

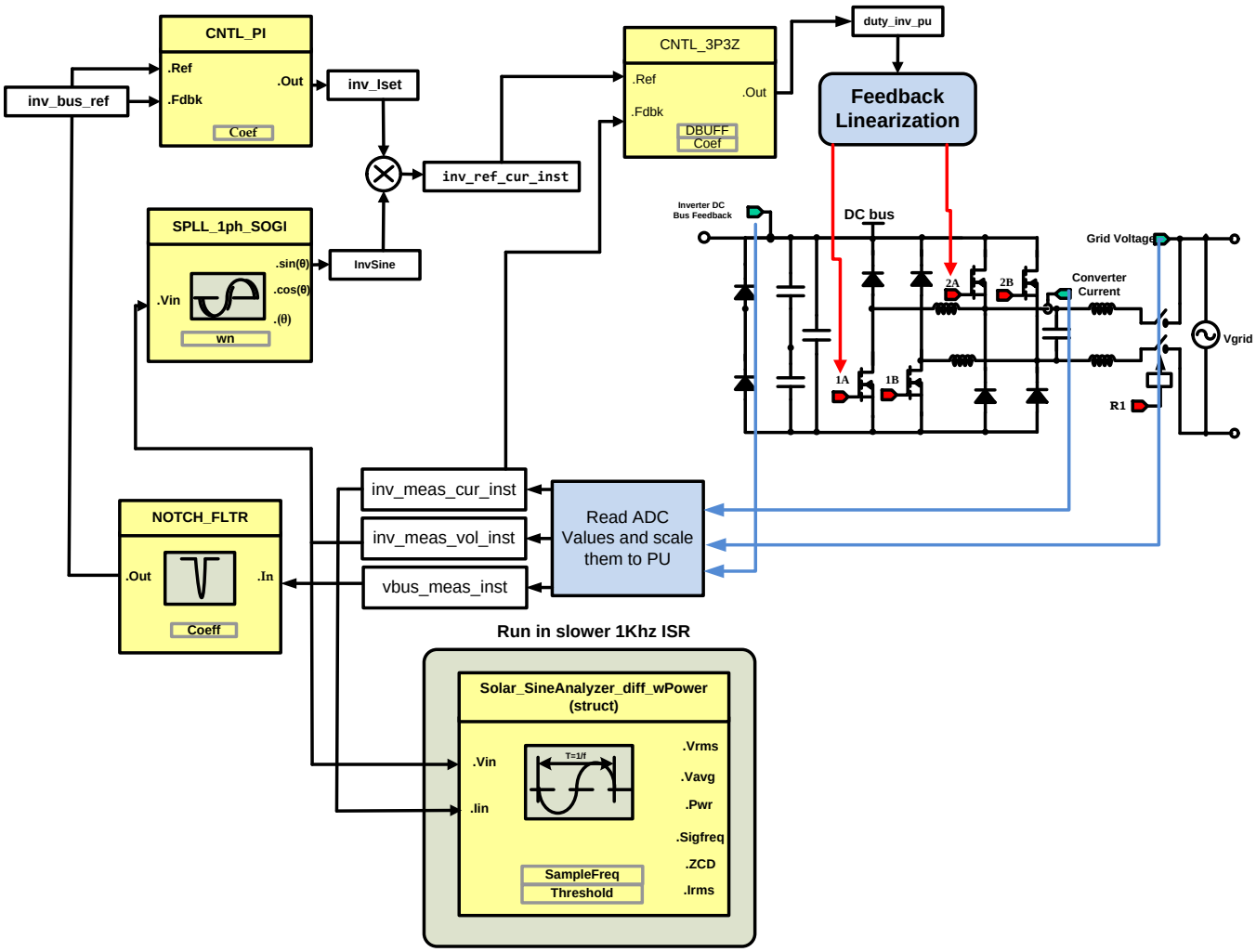



**Figure 17. Closed Loop Current Control of Inverter with Grid Connection**

## 3.4 DC-DC and DC-AC Integration

The DC-DC stage is switched at 100 Khz and DC-AC stage at 50 Khz. The peripheral ADC and PWMs on the C2000 device family have been designed to integrate multifrequency control loops and guarantee sampling at correct instances of the PWM waveform. However, as only one ADC is present (two sample and holds), the multi rate ISRs must not conflict with the ADC resource at any instance. For this the phase, the shift mechanism of the PWMs on the ePWM peripheral is employed. Figure 18 illustrates the timing diagram for configuring the EPWM for the inverter, and shows how to generate PWM waveforms for both the positive and negative half of the sine wave.

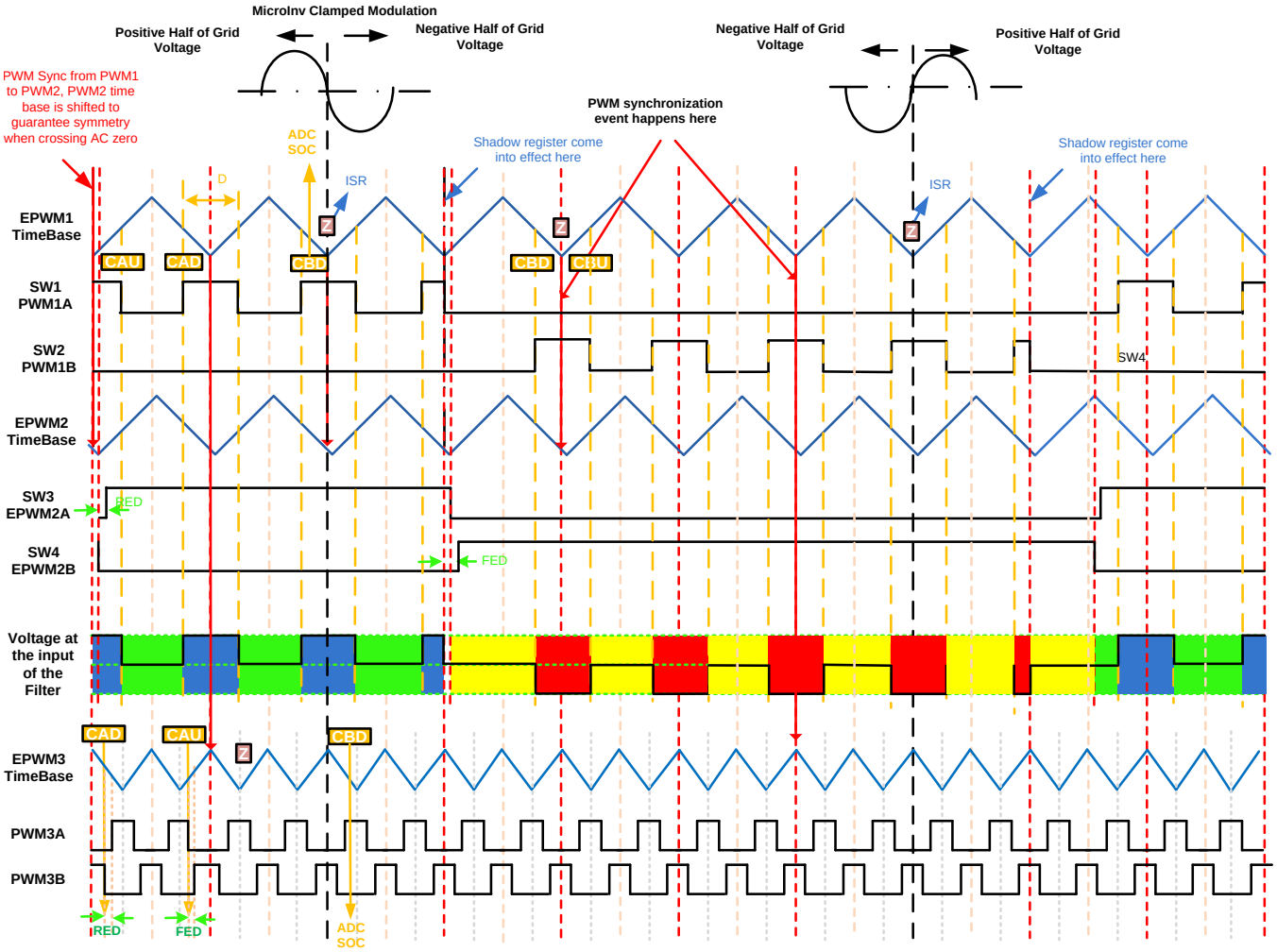



**Figure 18. PWM Switching Scheme for DC-DC and DC-AC Stage Integration**

## 4 Software Setup

Installing Code Composer and controlSUITE

1. If not already installed, install Code Composer v5.5 or later.
2. Go to http://www.ti.com/controlsuite and run the controlSUITE installer. Select to install the SolarMicroInverter software, and allow the installer to download all automatically-checked software components. If controlSUITE has been installed previously, select to update the software if the micro inverter kit folder is not available.

Setup Code Composer Studio

1. Open Code Composer Studio.
2. Once Code Composer Studio opens, the workspace launcher asks to select a workspace location as shown in Figure 19. Note that the workspace is a location on the hard drive where all user settings for the IDE, such as which projects are open, what configuration is selected, and which are saved. This can be anywhere on the disk, the location mentioned below is just for reference. Also note that if this is not the first time running Code Composer, this dialog may not appear.
   (a) Click the 'Browse…' button.
   (b) Create the path *C:\Documents and Settings\My Documents\ CCS_workspaces\ProjectWorkspace* by making new folders as necessary.
   (c) Uncheck the box that says 'Use this as the default and do not ask again.'

(d) Click OK.

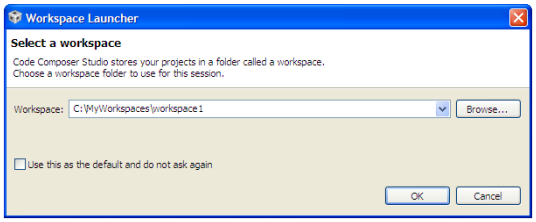



**Figure 19. Opening a New Workspace**

3. Configure Code Composer to know which MCU to connect to. If the user has previously connected to a F28035 device using the XDS100v1 emulator, they may skip this step. Click Target -> New Target Configuration. In the popup window as shown in Figure 20, name the new configuration xds100-f28035.ccxml. Ensure that the Use shared location checkbox is checked, and click Finish.

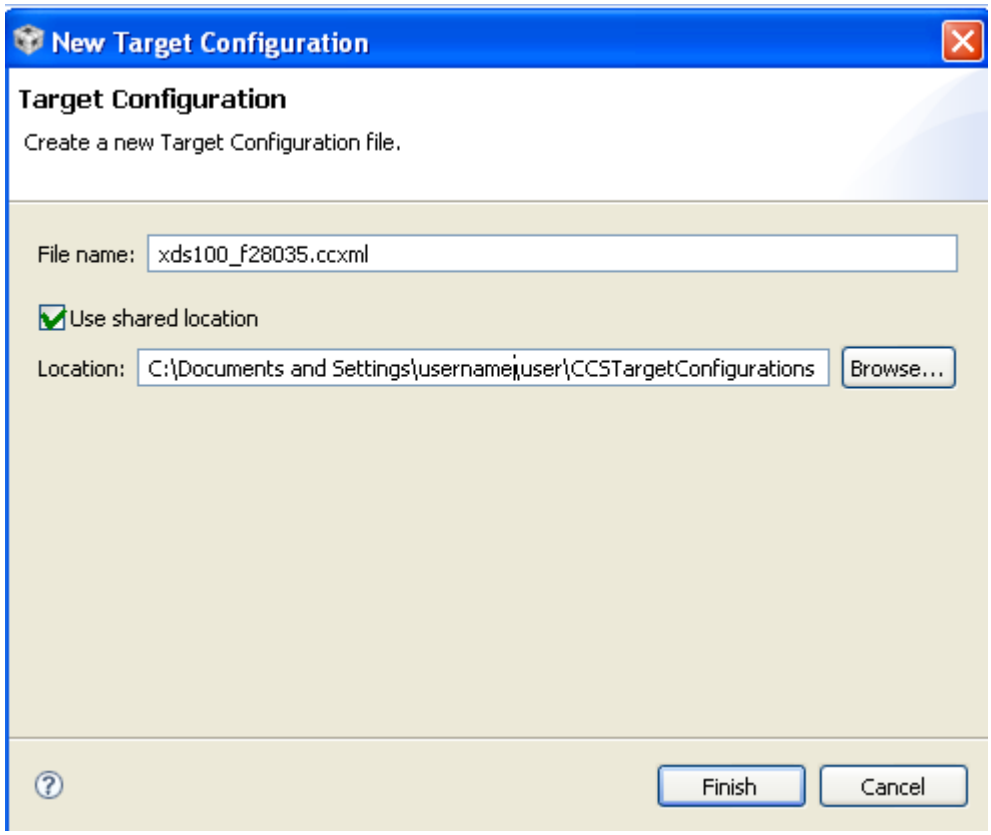



**Figure 20. New Target Configuration for XDS100 v1 and F28035 MCU**

4. A new tab now appears as shown in Figure 21. Select and enter the options as shown:
   (a) Connection – Texas Instruments XDS100v1 USB Emulator
   (b) Device – TMS320F28035
   (c) Click Save
   (d) Close the xds100-f28035.ccxml tab

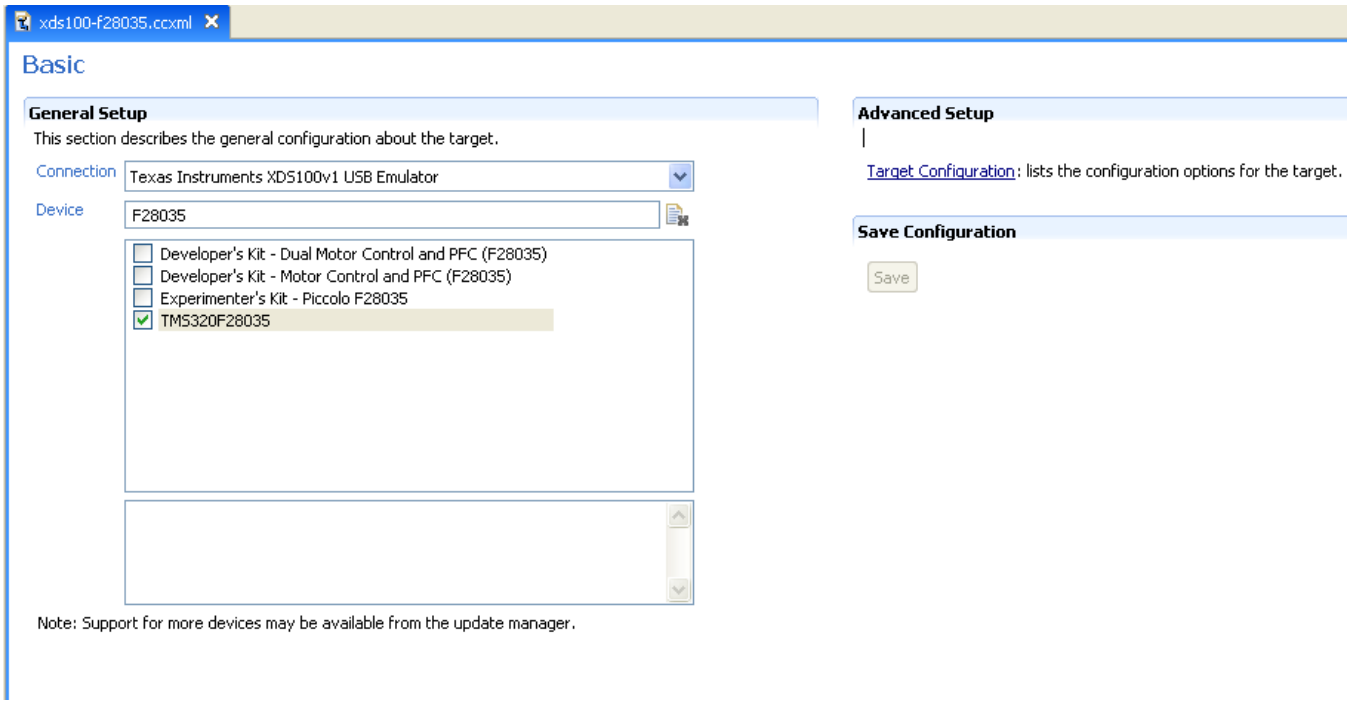



**Figure 21. Target Configuration**

5. If this is the user's first time using Code Composer, the xds100-F28035 configuration is now set as the default target configuration for Code Composer. Check this by going to View->Target Configurations. In the User Defined section, right-click on the xds100-F28035.ccxml file and select Set as Default. This tab also allows the user to reuse existing target configurations and link them to specific projects.
6. Add the solar micro inverter project into the current workspace by clicking Project→Import Existing CCS/CCE Eclipse Project, as shown in Figure 22.
   (a) Select the root directory of the solar micro inverter at *\controlSUITE\development_kits\TMDSSOLARUINVKIT_vX\MicroInv_F2803x*.

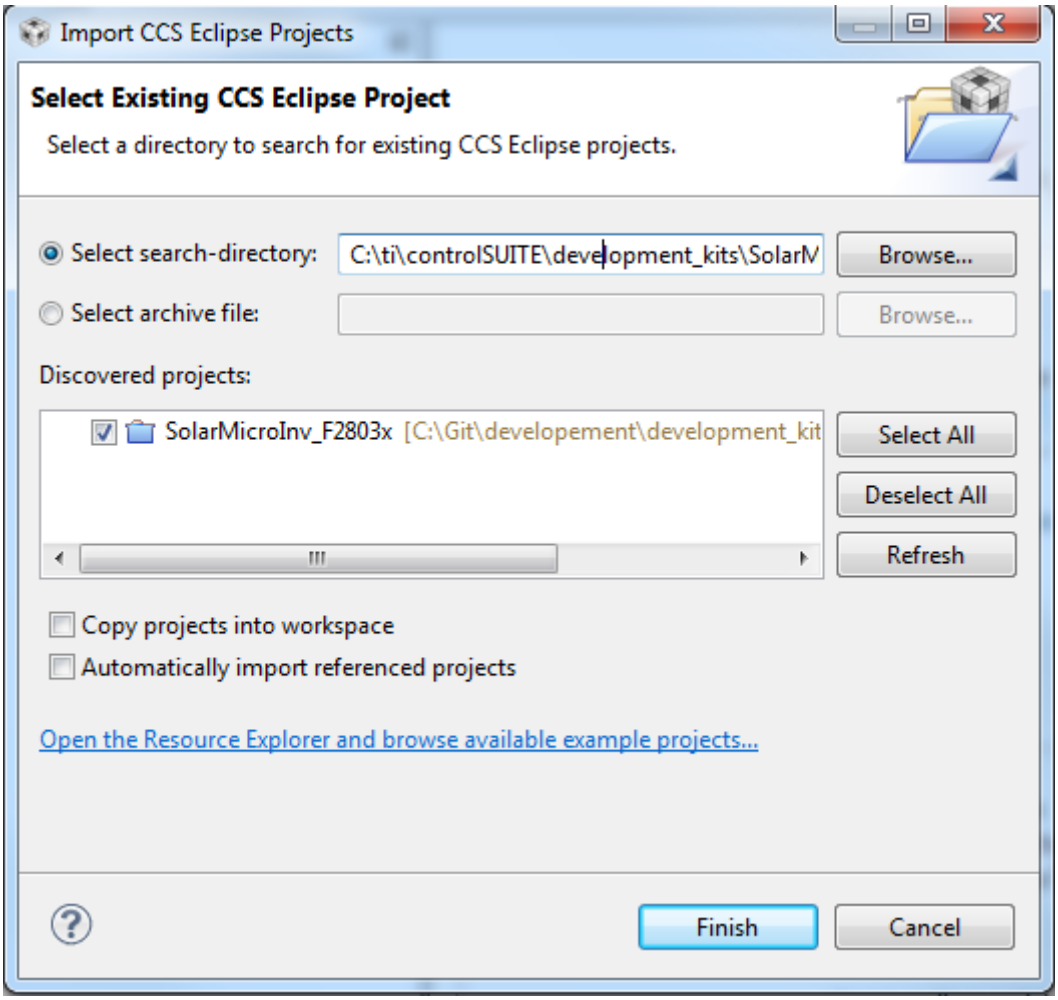



**Figure 22. Adding F28035 PV Inverter Project to the Workspace**

(b) Click Finish to copy all of the projects relevant for the kit into the workspace. For only a particular project to be copied, uncheck the box next to the other project names.

Configuring a Project

1. Expand the file structure of the project from the C/C++ Projects tab. Right-click on the project name and select Set as Active Project, if this is not already the case.
2. The project includes the target configuration for the XDS100v1 with F28035. Ensure this is set as active and default. If not, go to View→Target Configurations to see the XDS100v1 F28035 target configuration configured earlier. Right click on the target configuration name to link it to the imported project.
3. Figure 23 shows the project in the CCS C/C++ Project tab, and all the key files used in the project.

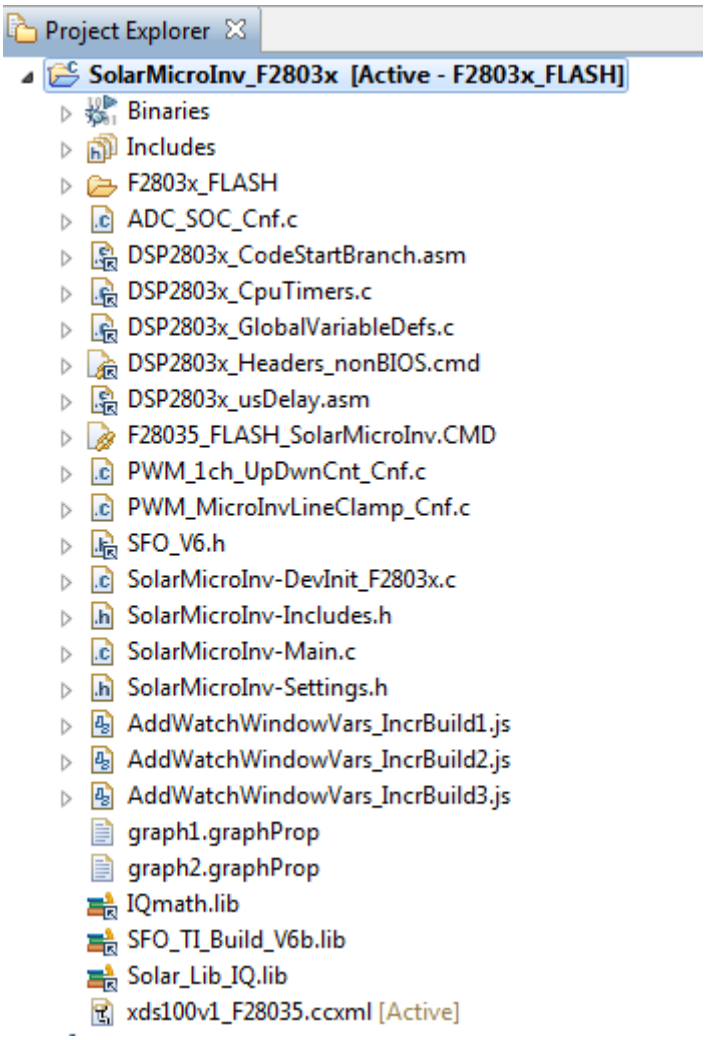



**Figure 23. PV Inverter Project in C/C++ Tab**

# 5 Hardware Setup

## 5.1 Equipment Required

The following equipment is required to run the solar micro inverter kit:

- Input power source, which can be supplied using one of the following options; however option (a) is highly recommended for initial debug and testing.
  - (a) Isolated DC power supply rated 25-44 VDC, 20 A, and 400 W min (Agilent Programmable DC source, model N5769A or similar).

    **NOTE:** MPPT must be disabled in the GUI before starting the inverter when using DC power supply.

  - (b) Solar panel emulator Agilent E4360 Solar Array Simulator or an equivalent
  - (c) Solar panel of 25-V to 44-V output, rated for around 140 W (max) if connecting to grid at 110 Vrms or aound 240 W (max) if connecting to 220 Vrms.
- 200-Ohms, 500-W power resistor
- 100-Ohms, 1-KW power resistor

- 1000-Ohms 500-W power resistor

The following equipment should be connected to the micro inverter board to observe different voltages and currents:

- Volt-meters of up to 500-V input range
- Oscilloscope of around 200-MHz bandwidth.
- High volt differential probe, Tektronix P5205 or similar.
- Current probe, Tektronix TCP202 or similar
- Current meter for output AC current, up to 5-A (RMS) range
- Current meter for input DC current, up to 12-A (DC) range

## 5.2 Jumper and Basic Board Setup

The micro inverter board has a primary (DC-DC) and secondary (DC-AC) side. Each of these sides can be powered using separate external bias power supplies. (Options for using internal bias are available but not used in this document. For details see Section 5.3)

1. Inspect and install jumpers. Before applying any power, ensure that these jumpers are configured properly. Table 3 lists the jumper configuration needed to run the GUI for the micro inverter EVM. Figure 24 shows some of the jumper locations on the PCB.

**Table 3. List of Jumper Settings for C2000 Solar Micro Inverter EVM**

| JUMPER NO. | JUMPER SETTINGS | COMMENTS |
|---|---|---|
| J15 | Install jumper | Jumper for 12-V external bias. |
| J38, J39 | Do not install jumpers | These jumpers can be used to provide the bias to the primary side. When J38 is populated, the primary side gets bias supply from the PV input voltage bias. When J39 is populated, the primary side is biased from the grid bias supply. (Note only one of the jumpers J15, J38, or J39 should be populated) |
| J40 | Install jumper | Jumper for 15-V external bias. |
| J41, J42 | Do not install jumpers | These jumpers can be used to provide bias supply to the secondary side. J41 sources the bias from the grid bias supply, and J42 sources the bias from the PV side bias for the secondary side. (Note only one of the jumpers J40, J41, or J42 should be populated) |
| TP8, TP9 | Jumper wire must be installed | This connects VBUS (flyback stage output) to inverter input |
| J17, J19, J32, J33 | Install all four jumpers | Jumpers for inverter stage PWM outputs from C2000 |
| J30, J31 | Install both jumpers | Jumpers for flyback stage PWM outputs from C2000 |

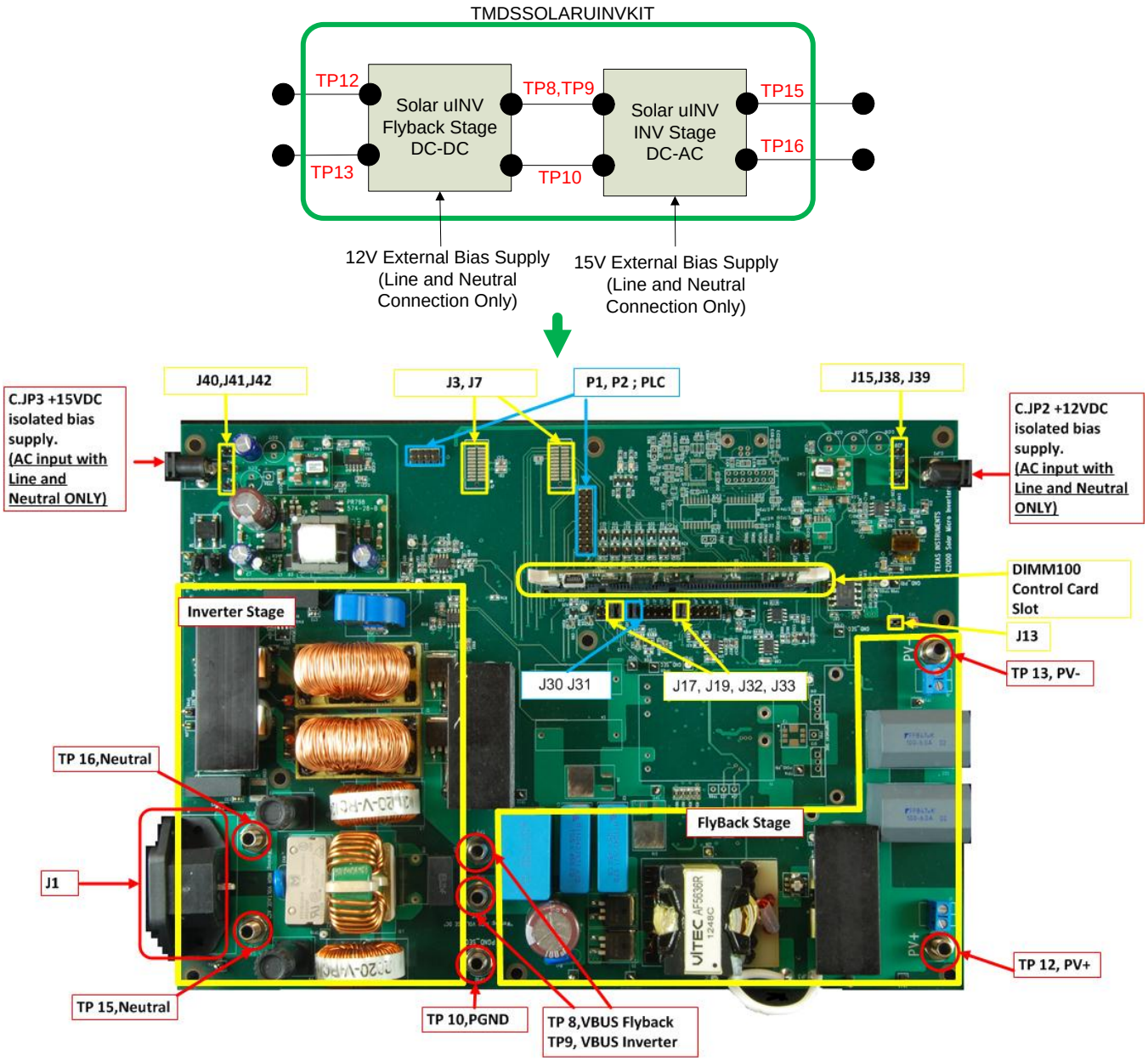



**Figure 24. Jumper and Connector Locations on C2000 Solar Micro Inverter**

2. Install and verify that the F28035 ISO control card is connected to the EVM header U6.
3. Check that the switch SW3 is set to the ON position on the control card, to enable a JTAG connection.
4. Connect a USB cable from the ISO control card to the PC on which GUI will run. Verify that the LED LD4 on the control card is ON indicating a USB connection.
5. Verify a jumper cable is connected between TP8 and TP9 to connect the DC-DC stage output to the DC-AC stage input. If not, install this jumper cable using a #16 cable.
6. Connect the external +12-VDC isolated-bias supply to the C:JP2 connector and power it on using 110-Vac or 220-Vac input. Ensure the supply uses Neutral and Line connection only (no earth connection).
7. Connect the external +15-VDC isolated-bias supply to the C:JP3 connector and power it on using 110-Vac or 220-Vac input. Ensure the supply uses Neutral and Line connection only (no earth connection).
8. Verify the control card is now powered by checking if the LD1 on the control card is ON.

This completes the basic hardware setup, further setup is described in the individual incremental builds.

### 5.3 Using Internal Bias Power Supply

The board has the option to generate bias supply from the AC source on the secondary side and from the panel on the primary side. Table 4 lists the jumper connections needed to use the internal bias power supply.

**Table 4. Jumper Configuration for using Internal Bias Power Supply on the Board**

| JUMPER NO. | JUMPER SETTINGS | COMMENTS |
|---|---|---|
| J39 | Install jumper | Jumper for on-board aux/bias supply output. |
| J38, J15 | Do not install jumpers | |
| J42 | Install jumper | Jumper for on-board aux/bias supply output. |
| J41, J40 | Do not install jumpers | |
| J45, J46 | Install jumper | Jumper for power input to on-board aux/bias supply. |
| TP8 – TP9 | Jumper wire must be installed | This connects Vbus (Fly-back stage output) to inverter input |
| J17, J19, J32, J33 | Install all 4 jumpers | Jumpers for inverter stage PWM |
| J30, J31 | Install both jumpers | Jumpers for Fly-back stage PWM |

## 6 Software Builds

The project is divided into simplified incremental builds to run smaller subsystems of increasing complexity until the target system is built up completely, to make it easier to learn the board and the software. This approach is also good for debugging and testing the boards. The various build options are shown below. To select a particular build option, set the appropriate value in the BUILD setting, found in the {ProjectName}-*Settings.h* file. Once the build option is selected, compile the complete project by selecting the rebuild-all compiler option. The following chapters provide more details on how to run each of the build options.

The following are the build options supported on the solar micro inverter kit.

Build 1: Open loop check for inverter and DC-DC flyback stage, software test for SPLL

Build 2: Individually test the closed current loop inverter and closed current loop for DC-DC flyback (the two stages are not connected).

For the inverter stage GRID_CONNECT, #define is used to incrementally test the closed loop performance of the loop without the grid connection at first, and then with the grid connection:

- **GRID_CONNECT 0**: Forced ramp angle is used for the inverter control, tests the inverter current loop control with resistive load at the output.
- **GRID_CONNECT 1**: AC source/grid is connected at the output of the micro inverter, and SPLL is used to calculate the grid angle. Inverter current loop control with grid connection is tested in this build. (Note that a resistive load can be connected at the output to see grid current inversion). The {ProjectName}-*Settings.h* file must be modified with the AC source/grid AC frequency for the GRID_FREQ define.

For the DC-DC flyback stage, the MPPT define specifies if the MPPT is implemented or not.

- **MPPT 0**: Fixed input current is commanded from the flyback stage. The user must ensure a proper resistive load is connected at the output.
- **MPPT 1**: Flyback stage current command is used to maintain the input panel at MPPT. The user must ensure a proper resistive load is connected at the output and a panel or a panel emulator is connected at the input.

Build 3: Tests the action of the inverter and DC-DC stages when connected together. Output of the flyback stage is connected to the inverter input directly, with no resistive load at the flyback stage output. The operation of the stages is commanded by a state machine.

- **GRID_CONNECT 0, MPPT 0**: This option is used to test both stages together with a resistive load at

the output of the inverter. This tests the board with a DC source at the input instead of a PV panel emulator.

- **GRID_CONNECT 0, MPPT 1**: This option tests the combined action of both stages and the state machine when a resistive load is at the output and input is from a PV / PV emulator source.
- **GRID_CONNECT 1, MPPT 0**: This option tests both stages together with a grid connection at the output and a DC source at the input,instead of a PV panel emulator; therefore MPPT is disabled.
- **GRID_CONNECT 1, MPPT 1**: This option tests the combined action of both stages and the state machine when the grid is connected at the inverter output. DC input is from a PV / PV emulator source, therefore MPPT is enabled. This is the full PV inverter system build.

All software files related to this C28x controlled Solar Explorer system, such as the main source files, ISR assembly files, and the project file for C framework, are located in the directory: …*controlSUITE\development_kits\TMDSSOLARUINVKIT_v100 \MicroInv_F2803x*.

## *6.1 BUILD = 1*

### 6.1.1 Objective

The objectives of this build are:

- Evaluate the PWM and ADC software driver modules
- Verify the MOSFET gate driver, voltage, and current sensing circuit
- Become familiar with the operation of Code Composer Studio (CCS)
- Test the SPLL module

Under this build, the system runs in open loop for the inverter power stage and DCDC boost stage. The steps required for building and running a project are explained in the following sections.

### 6.1.2 Overview

The software in Build1 is configured to quickly evaluate the PWM driver module and ADC drivers for the inverter and the flyback stage, by viewing the related waveforms on a scope, a multi meter, or on an expressions window in CCS. The user can also observe the effect of changing the inverter modulation index and flyback duty on the power stage through observing variables in the watch window.

### 6.1.3 Flyback Open Loop Check

#### *6.1.3.1 Procedure*

Assuming the hardware and software setups are followed as described in Section 4 and Section 5.2, arrange the setup such that a DC supply is connected to the flyback input stage, and a 1K Ohm resistor is connected at the output of the flyback stage. The connection points where the power supply and the load must be connected are highlighted in Figure 25.

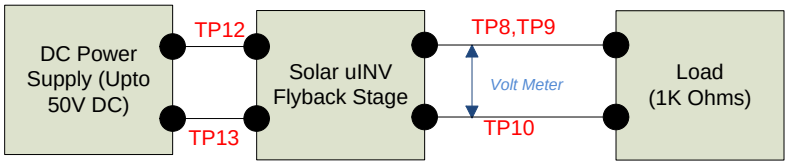



**Figure 25. Power Stage Connections for Flyback Open Loop Check**

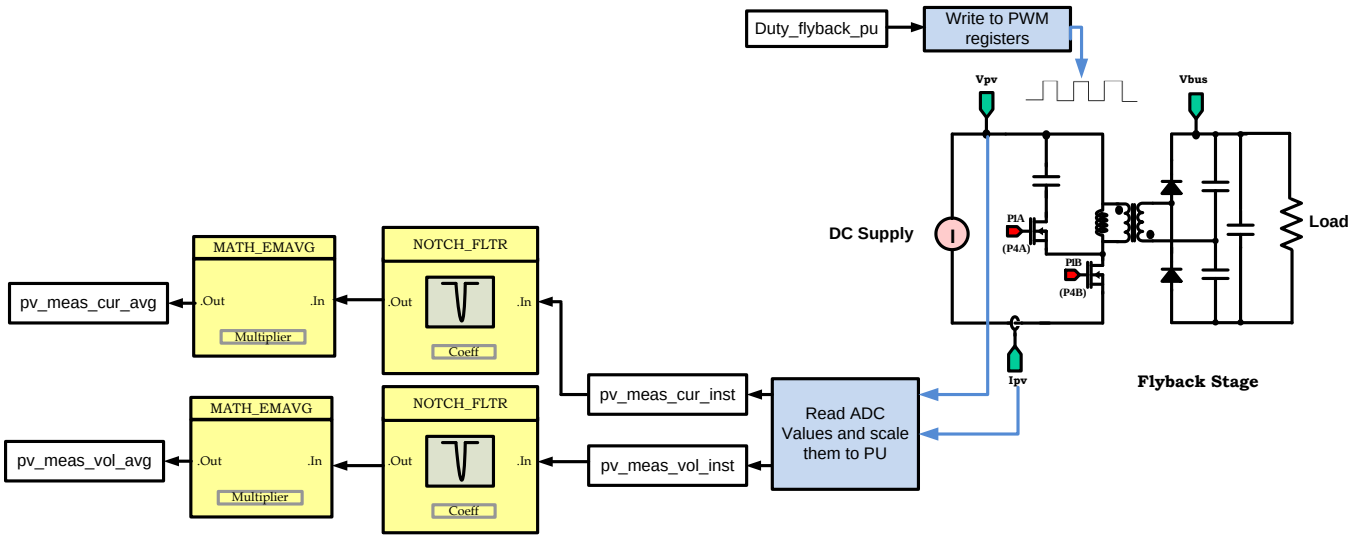



**Figure 26. Build 1 Flyback Open Loop Check Software Diagram**

1. In CCS click the plus sign (+) sign on the project name in the project window to display all the files used in the micro inverter project. Open and inspect the *SolarMicroInv-DevInit_F2803x.c* by double clicking on the filename. Inspect the system clock and peripheral clocks setup in this file. Also notice how the shared GPIO pins are configured for each peripheral.
2. Open and inspect *SolarMicroInv-Main.c*. Locate the main() function in this file.
3. In the main() is the device and peripheral initialization, notice the call made to the *DeviceInit()* function and other peripheral initialization functions. Inspect the PWM initialization and ADC initialization for the flyback and the inverter.
4. Check the initialization of the solar lib modules and the control variables.
5. Run an offset calibration routine to calculate the analog offset on the ADC input channels for PV current and the grid.
6. Configure the interrupt for a 50-KHz ISR, which is used for the control loop and 1-KHz ISR for background tasks. The 1-KHz ISR is chosen such that the 50-KHz ISR can interrupt the 1-KHz ISR.

### 6.1.3.2 Build and Load the Project

1. Select the incremental build option as 1 in the *SolarMicroInv-Settings.h* file.

**NOTE:** Whenever changing the incremental build option, always select Rebuild All.

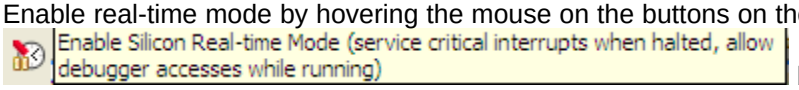
2. Click the Project →Rebuild All button and watch the tools run in the build window. The project will compile successfully.
3. Click Target→Debug Active Project. The program loads into the flash of the F28035 device. The user should now be at the start of main().

### 6.1.3.3 Debug Environment Windows

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 *AddWatchWindowVars_IncrBuild1.js* script file located inside the project folder. This populates the watch window with the appropriate variables needed to debug the system and populate the variables for an appropriate Q formats. Click on Continuous Refresh button on the watch window to enable a continuous update of values from the controller, as shown in Figure 27.

### 6.1.3.4 Using Real-Time Emulation

Real-time emulation is a special emulation feature that allows windows within Code Composer Studio to be updated at a rate up to 10 Hz 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 the mouse on the buttons on the horizontal toolbar and clicking the 

button.
2. A message box may appear. If so, select YES to enable debug events. This sets bit 1 (DGBM bit) of status register 1 (ST1) to 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.
3. Click on the Continuous Refresh button  for the watch view.

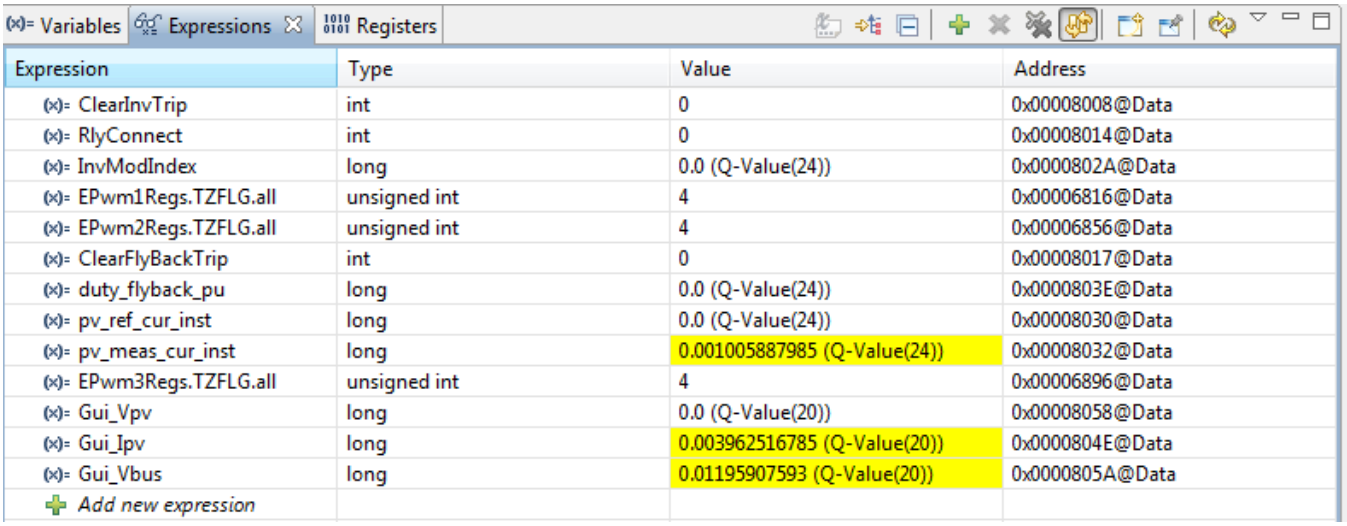



**Figure 27. Watch Window Populated using Script**

### 6.1.3.5 Run The Code

1. Run the project, 
2. In the watch view, check the value of Gui_Vpv, which will be the voltage at the input of the board. Slowly raise this voltage to 30 V of the input power supply and see Gui_Vpv change as the input voltage is increased, as shown in Figure 28.
3. Clear the flyback stage trip by writing a 1 to the ClearFlyBacktrip variable. The EPwm3Regs.TZFLG.all goes to zero as soon as the trip is cleared.
4. Enter a duty in the duty_flyback_pu field. Starting from 0.1, go up to 0.3. The watch window appears as shown in Figure 28:

Variables | Expressions | Registers

| Expression | Type | Value | Address |
|---|---|---|---|
| ClearInvTrip | int | 0 | 0x00008008@Data |
| RlyConnect | int | 0 | 0x00008014@Data |
| InvModIndex | long | 0.0 (Q-Value(24)) | 0x0000802A@Data |
| EPwm1Regs.TZFLG.all | unsigned int | 4 | 0x00006816@Data |
| EPwm2Regs.TZFLG.all | unsigned int | 4 | 0x00006856@Data |
| ClearFlyBackTrip | int | 0 | 0x00008017@Data |
| duty_flyback_pu | long | 0.3000000119 (Q-Value(24)) | 0x0000803E@Data |
| pv_ref_cur_inst | long | 0.0 (Q-Value(24)) | 0x00008030@Data |
| pv_meas_cur_inst | long | 0.1225879192 (Q-Value(24)) | 0x00008032@Data |
| EPwm3Regs.TZFLG.all | unsigned int | 0 | 0x00006896@Data |
| Gui_Vpv | long | 30.03022957 (Q-Value(20)) | 0x00008058@Data |
| Gui_Ipv | long | 1.797566414 (Q-Value(20)) | 0x0000804E@Data |
| Gui_Vbus | long | 222.7752609 (Q-Value(20)) | 0x0000805A@Data |
| *Add new expression* | | | |

**Figure 28. Watch Window During Running of Build 1 Flyback Stage**

5. Check the multimeter reading for the flyback stage output matches that of the watch window, and the current from the DC power supply equals the measurement in the watch window.
6. This completes the DC flyback open loop check. If this does not work, check the PWM and gate drive signals using a differential probe referring to the kit schematics.
7. To end this test, enter the duty_flyback_pu to zero and reduce the DC power supply voltage to 0 V. Disconnect the flyback stage DC power supply input and the resistive load at the output of the flyback stage. The code can be kept running for the next check.

### 6.1.4 Inverter Open Loop Check

Next check the inverter stage in open loop. Assuming the hardware and software setups are followed from the previous builds, rearrange the setup such that a DC supply is connected to the inverter input stage and a 100-200 Ohm resistor is connected at the output of the inverter stage. The connection points where the power supply and the load must be connected are highlighted in Figure 29. Keep the HV supply off while making the connections and do not turn it on unless instructed, which occurs later in the document. The software structure used for the open loop test of the inverter is shown in Figure 30.

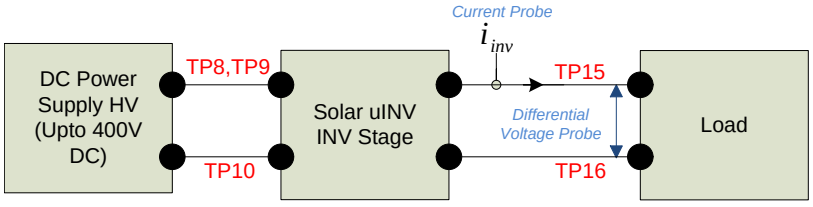



**Figure 29. Power Stage Connections for Inverter Open Loop Check**

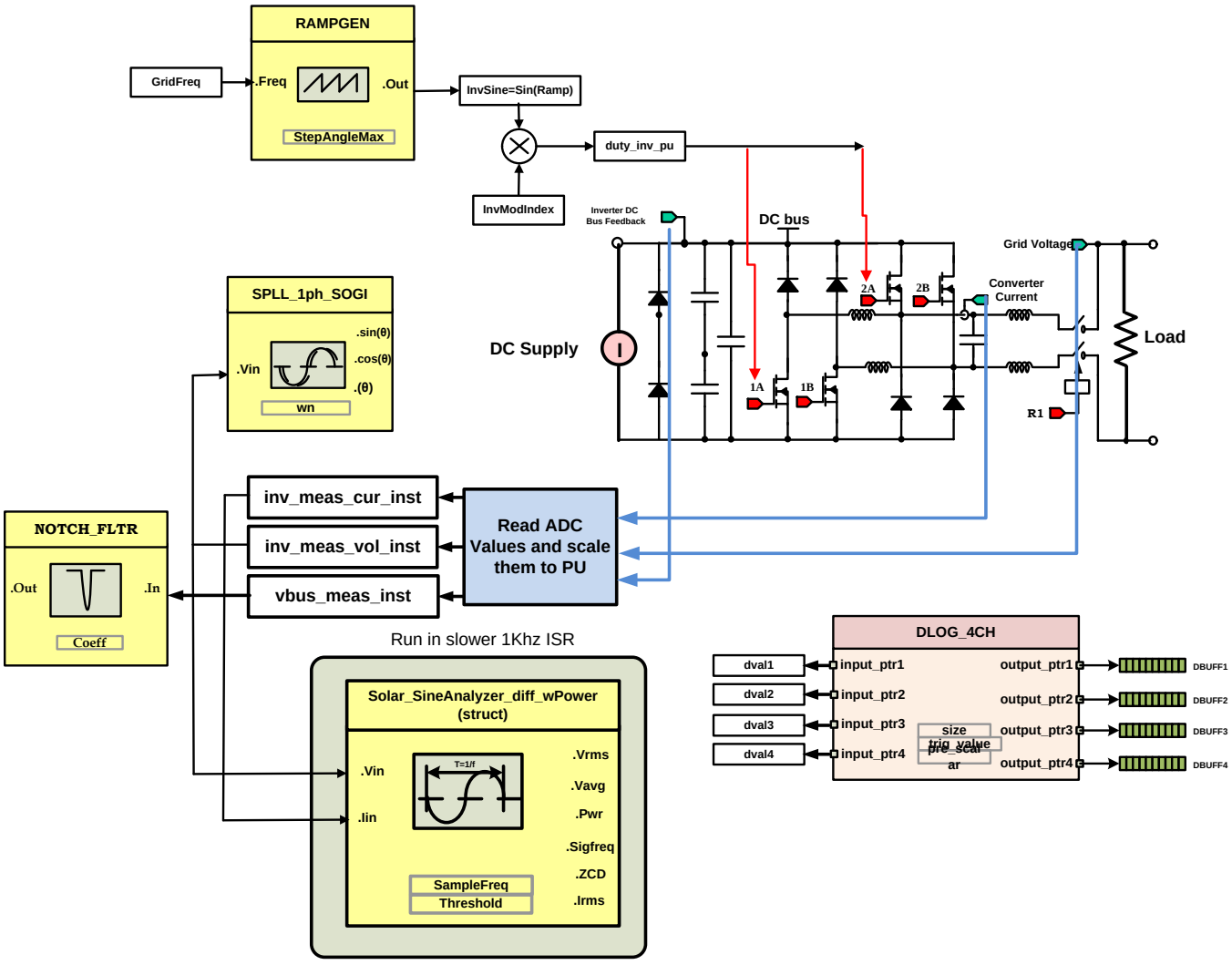



**Figure 30. Build 1 Inverter Open Loop Check Software Diagram**

Ensure no high voltages are present on the board, the input of the flyback stage is zero, and no voltages are present at the output. Disconnect the load at the flyback stage output, if not already done, and connect a high-voltage DC supply at the input. Connect a 100-Ohms load at the output of the inverter with a connection for a differential probe and a current probe as shown in Figure 29.

### 6.1.4.1 Run the Code

1. The code may already be running from the previous section 0; if not, follow the steps from the previous sections to import the micro inverter project into CCS, set the incremental build level to 1, compile and build the project, launch a debug session, enable real time mode, populate the watch window using the script for build level1, and run the program.
2. With the program running, clear the inverter trip by writing a 1 to the ClearInvTrip variable in the watch window. Notice the clearing of the trip flag of the EPWM1 and 2 TZFLG in the watch window.
3. Connect the inverter to the load by firing the Relay ON, by writing 1 to the RelayConnect variable.
4. Write a 0.5 to InvModIndex.
5. Turn on the HV DC supply and slowly increase the DC bus voltage. Observe the DC Bus voltage reading on the GUI to verify the connections are correct.

6. Increase the voltage up to 300 V at the DC input side, and observe the Gui_Vrms and Gui_Irms values in the watch expressions. Observe the voltage and the currents on the scope, which should match Figure 31 and Figure 32 respectively.

Variables | Expressions | Registers

| Expression | Type | Value | Address |
|---|---|---|---|
| ClearInvTrip | int | 0 | 0x00008008@Data |
| RlyConnect | int | 1 | 0x00008014@Data |
| InvModIndex | long | 0.5 (Q-Value(24)) | 0x0000802A@Data |
| EPwm1Regs.TZFLG.all | unsigned int | 0 | 0x00006816@Data |
| EPwm2Regs.TZFLG.all | unsigned int | 0 | 0x00006856@Data |
| ClearFlyBackTrip | int | 0 | 0x00008017@Data |
| duty_flyback_pu | long | 0.0 (Q-Value(24)) | 0x0000803E@Data |
| pv_ref_cur_inst | long | 0.0 (Q-Value(24)) | 0x00008030@Data |
| pv_meas_cur_inst | long | 0.0 (Q-Value(24)) | 0x00008032@Data |
| EPwm3Regs.TZFLG.all | unsigned int | 4 | 0x00006896@Data |
| Gui_Vpv | long | 0.6422424316 (Q-Value(20)) | 0x00008058@Data |
| Gui_Ipv | long | 0.003275871277 (Q-Value(20)) | 0x0000804E@Data |
| Gui_Vbus | long | 299.2802601 (Q-Value(20)) | 0x0000805A@Data |
| Gui_Vrms | long | 106.890625 (Q-Value(6)) | 0x0000805C@Data |
| Gui_Irms | long | 1.086669922 (Q-Value(12)) | 0x00008050@Data |
| Gui_Prms | long | 111.375 (Q-Value(6)) | 0x00008052@Data |

**Figure 31. Watch Expression Build Level 1 Inverter Open Loop Check**

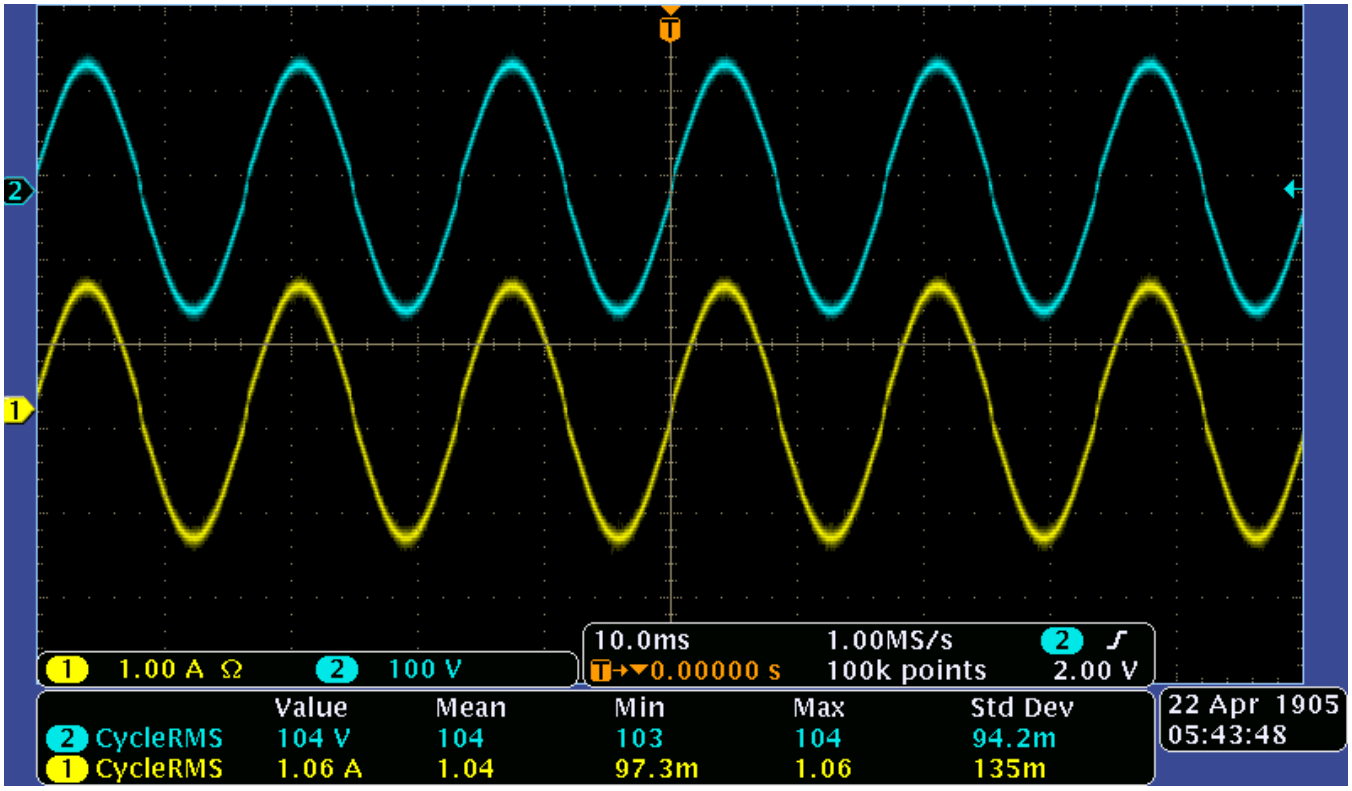



**Figure 32. Build Level 1 Inverter Voltage and Current Check**

7. The inverter current and voltage measurements can also be verified by viewing the data in the graph window. Inverter sine, PLL output, inverter measured current, and voltage are mapped to DLOG variables in the ISR as follows:

```
// First two DBUFF check SPLL operations
D_val1 = (int16)_IQtoIQ15(InvSine);
D_val2 = (int16)_IQtoIQ15(spll2.sin<<1);
// Next two check Vac and Iac measurement
D_val3 = (int16)_IQtoIQ15(inv_meas_cur_inst);
D_val4 = (int16)_IQtoIQ15(inv_meas_vol_inst);
DLOG_4CH_IQ_MACRO(dlog1);
```

DBUFF3 and DBUFF4 are used for inverter current and voltage measurements respectively. Go to Tools→Graph→DualTime and click on Import and point to the graph2.GraphProp file inside the project folder. This populates the graph properties window as shown in Figure 33. Alternatively, the user can enter the values as shown in Figure 33 manually. Once the entries are verified, click OK. Two graphs now appear in CCS. Click on Continuous Refresh on these graphs.

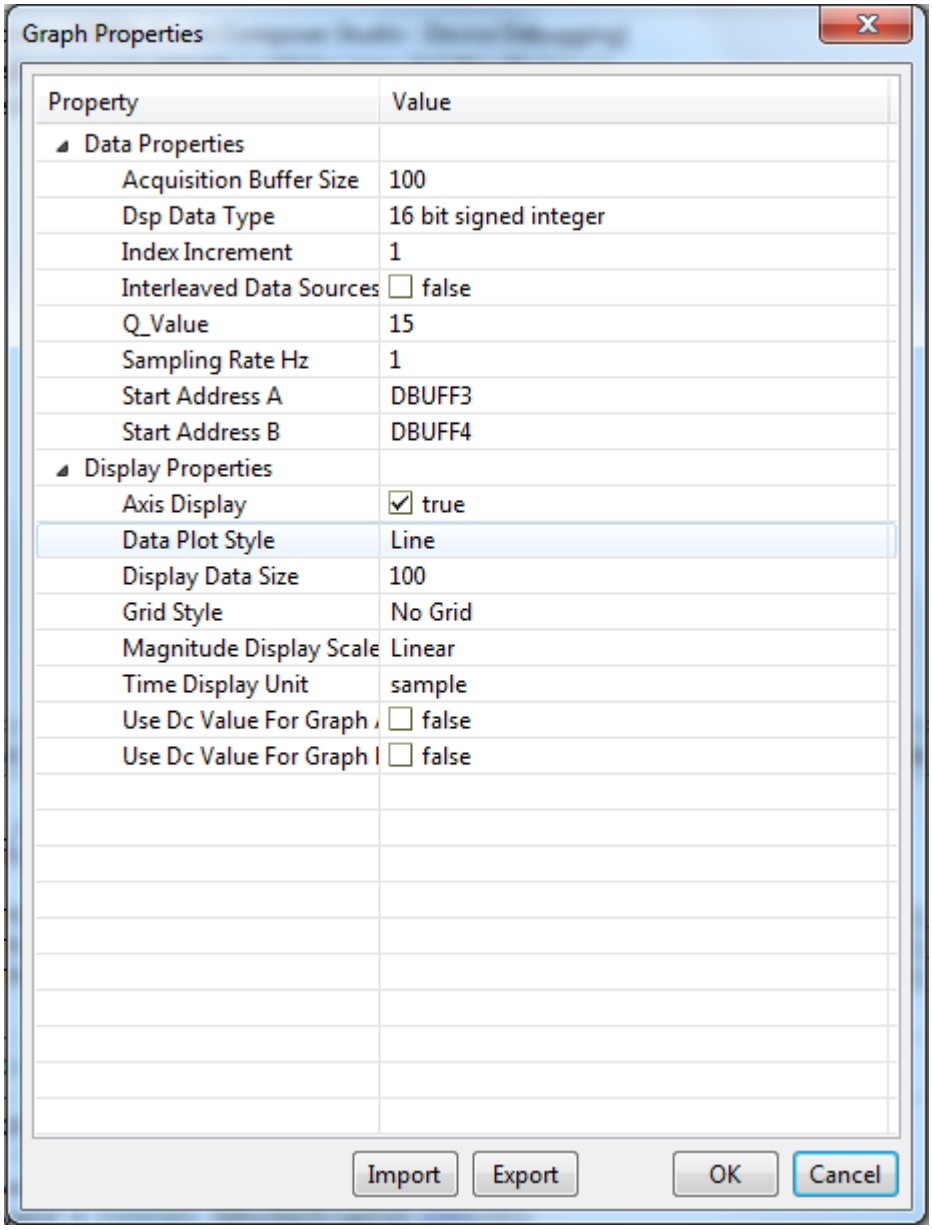



**Figure 33. Graph Window Properties**

8. The graphs will appear as shown below.

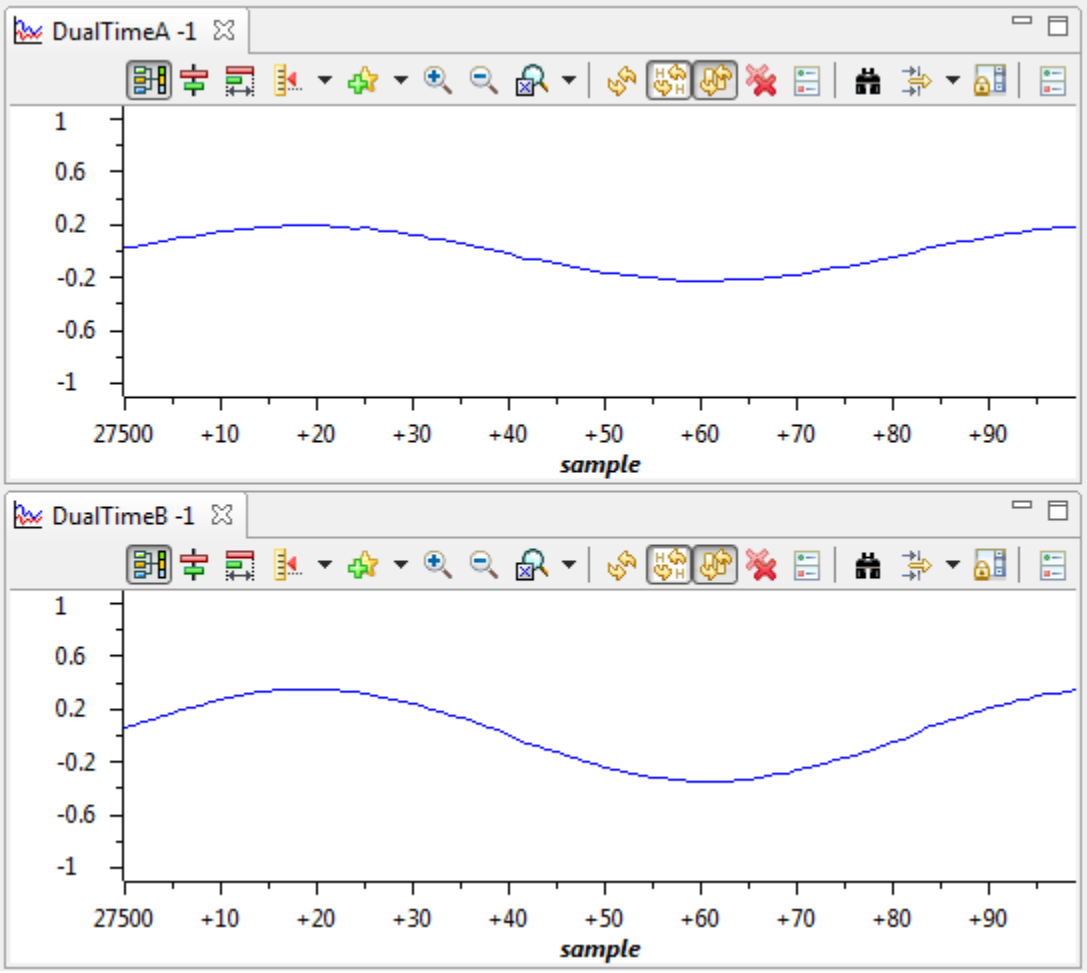



**Figure 34. Build 1 Inverter Open Loop CCS Graphs**

9. Change the InvModIndex variable and watch the voltage and current measurements change in the watch window and the graphs.
10. PLL performance can also be checked by importing the graph1.GraphProp file, which plots the invsine, the forced angle and the PLL output sine value.
11. This completes the inverter open loop check. To end the debug session, enter 0 in the InvModIndex and slowly reduce the DC supply voltage input to the inverter to zero. Put RelayConnect to 0.
12. Fully halting the MCU when in real-time mode is a two-step process. First, halt the processor by using the Halt button on the toolbar , or by using Target→Halt. Then take the MCU out of real-time mode by clicking on . Finally reset the MCU .
13. Close CCS debug session by clicking on Terminate Debug Session (Target→Terminate all).

## 6.2 Build = 2

### 6.2.1 Objective

The objective of this build is to run a closed current loop individually for the flyback and inverter stage. MPPT is tested for the flyback stage, and grid-connected current control is tested for the inverter stage individually.

### 6.2.2 Overview

The software in Build2 is configured to quickly evaluate the closed current loop performance of the flyback and the inverter stage. The steps required for building and running a project are explained in the following sections.

### 6.2.3 Flyback Closed Current Loop without MPPT

#### 6.2.3.1 Procedure

In this build, the hardware setup from BUILD1 is used for the flyback stage. Optionally, a current probe can be connected at the input to check the closed input current loop performance of the flyback stage. Assuming the hardware and software setups are followed from the previous builds, rearrange the setup such that a DC supply is connected to the flyback input and a 500-Ohm resistor is connected at the output of the flyback stage. The connection points where the power supply and the load need to be connected are highlighted in Figure 35. The software structure used for the closed loop flyback test is shown in Figure 36.

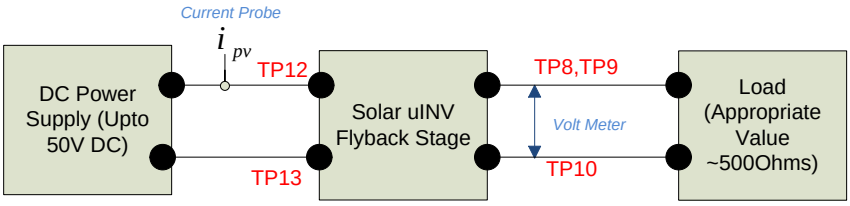



**Figure 35. Build Level 2 Flyback Close Current Loop Operation**

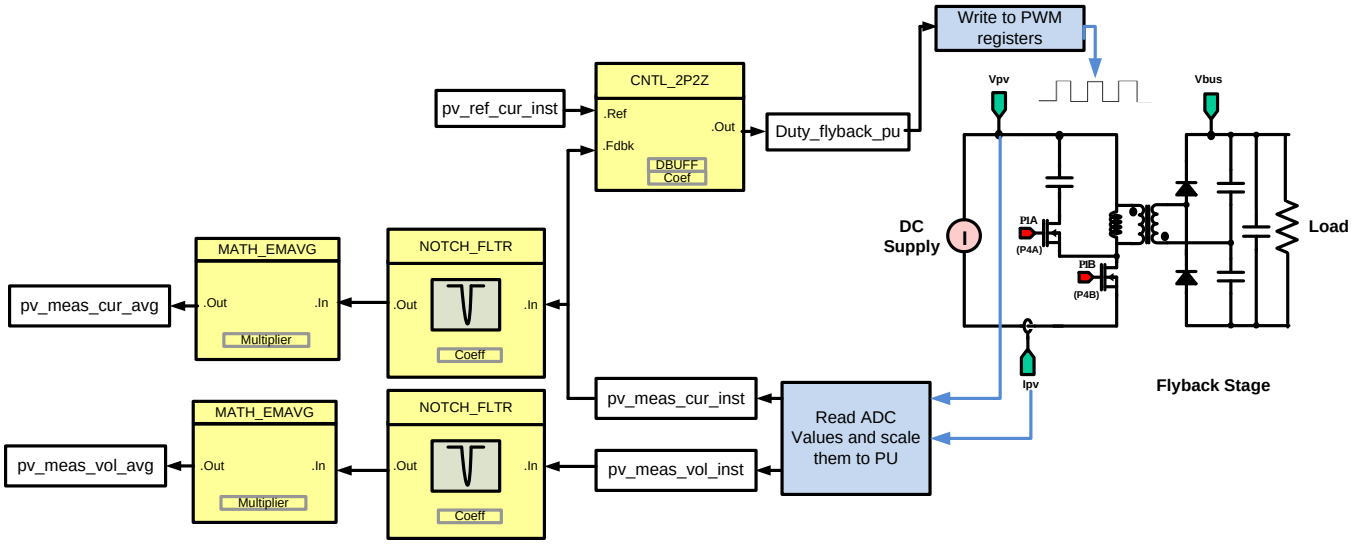



**Figure 36. Build 2 Closed Loop Flyback Test without MPPT**

1. Follow the steps from the previous sections to import the micro inverter project into CCS.
2. Change the build level open the *SolarMicroInv-Settings.h* file. Ensure the defines are as below:

```
#define INCR_BUILD 2
#define GRID_CONNECT 0
#define MPPT 0
```

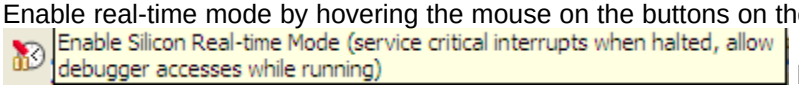
**NOTE:** When changing the incremental build option, always select Rebuild All.

3. Click the Project→Rebuild All button and watch the tools run in the build window.
4. Click on Target→Debug Active Project. The program loads into the flash. The user should now be at the start of main().

#### 6.2.3.2 Debug Environment Windows

1. Click on View→Scripting Console to open the scripting console, and click Open on the top left corner of this console window to open the AddWatchWindowVars_IncrBuild2.js script located inside the project folder. This populates the watch window with the appropriate variables needed to debug the system and the appropriate Q formats. Click on the Continuous Refresh button  on the watch window to enable continuous update of values from the controller.

#### 6.2.3.3 Using Real-Time Emulation

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

button.
2. A message box may appear. If so, select YES to enable debug events. This sets bit 1 (DGBM bit) of the status register 1 (ST1) to 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.
3. Click on the Continuous Refresh  button for the watch view.

#### 6.2.3.4 Run the Code

1. Run the project by clicking on the  button.
2. In the watch view, check the value of Gui_Vpv, which is the voltage at the input of the board. Slowly raise this voltage to 30 V and watch the Gui_Vpv change as the input voltage increases.
3. Clear the flyback stage trip by writing a 1 to the ClearFlyBacktrip variable. EPwm3Regs.TZFLG.all will go to zero as soon as the trip is cleared.
4. Enter an input current value in pu format in the pv_ref_cur_inst. For example, for 1.5 Amps, enter _IQ24(0.1), the max current sense is 15 Amps at the input of the flyback. Note the meas_cur_inst matches the value for the set reference. Change the pv_ref_cur_inst and see the input track to the new reference, as shown in Figure 37.

**CAUTION**

The user must ensure that the DC bus voltage at the output of the flyback must never exceed >400 V, and must connect the appropriate load to match this condition.

Variables | Expressions | Registers

| Expression | Type | Value | Address |
|---|---|---|---|
| ClearInvTrip | int | 0 | 0x00008015@Data |
| RlyConnect | int | 0 | 0x00008010@Data |
| inv_Iset | long | 0.0 (Q-Value(24)) | 0x00008040@Data |
| CloseIloopInv | int | 0 | 0x00008013@Data |
| EPwm1Regs.TZFLG.all | unsigned int | 4 | 0x00006816@Data |
| EPwm2Regs.TZFLG.all | unsigned int | 4 | 0x00006856@Data |
| Gui_Prms | long | 0.0 (Q-Value(6)) | 0x0000804C@Data |
| Gui_Vrms | long | 0.0 (Q-Value(6)) | 0x00008050@Data |
| Gui_Irms | long | 0.0 (Q-Value(12)) | 0x00008052@Data |
| ClearFlyBackTrip | int | 0 | 0x00008016@Data |
| duty_flyback_pu | long | 0.1521959305 (Q-Value(24)) | 0x00008044@Data |
| pv_ref_cur_inst | long | 0.09999996424 (Q-Value(24)) | 0x00008034@Data |
| pv_meas_cur_inst | long | 0.09426760674 (Q-Value(24)) | 0x00008026@Data |
| EPwm3Regs.TZFLG.all | unsigned int | 0 | 0x00006896@Data |
| Gui_Vpv | long | 29.43839264 (Q-Value(20)) | 0x0000805A@Data |
| Gui_Ipv | long | 1.491422653 (Q-Value(20)) | 0x00008058@Data |
| Gui_Vbus | long | 163.3015804 (Q-Value(20)) | 0x00008054@Data |
| *Add new expression* | | | |

**Figure 37. Build 2 Watch Expressions Closed Loop Flyback Check Without MPPT**

5. This completes the flyback closed current loop check without MPPT. Enter 0 for the pv_ref_cur_inst in the watch window, and reduce the DC input voltage 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 the Terminate Debug Session  button (Target→Terminate all).

### 6.2.4 Flyback Closed Current Loop with MPPT

#### *6.2.4.1 Procedure*

In this build, the MPPT algorithm is tested. For this, a PV panel or a PV panel emulator must be used. Ensure that no part of the board is energized. If a panel emulator is used for the input source, make the connections as shown in Figure 38. An appropriate load at the output of the flyback stage must be connected to ensure the output of the flyback stage does not go beyond 400 V for the maximum power programmed in the PV panel emulator. Typical characteristics programmed in a panel emulator can be:

1. Open circuit voltage 40 V
2. Maximum power point voltage 30 V
3. Short circuit current 5.2 Amps
4. Maximum power point current 4 Amps

The software used for this build is shown in Figure 39.

**CAUTION**

The panel is an energized source and extreme caution must be taken to connect this to the board. Refer to your organization's safety procedure before connecting the panel.

If only a panel is available, do not connect the panel at this stage and instead connect the load at the output of the flyback stage. Use #16 wires for all connections.

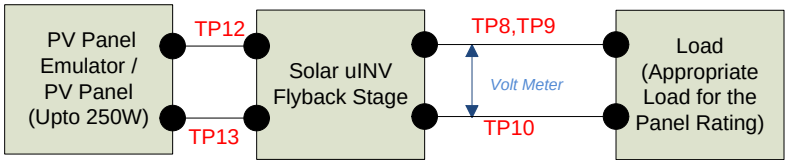



**Figure 38. Build 2 Closed Loop Flyback Check with MPPT**

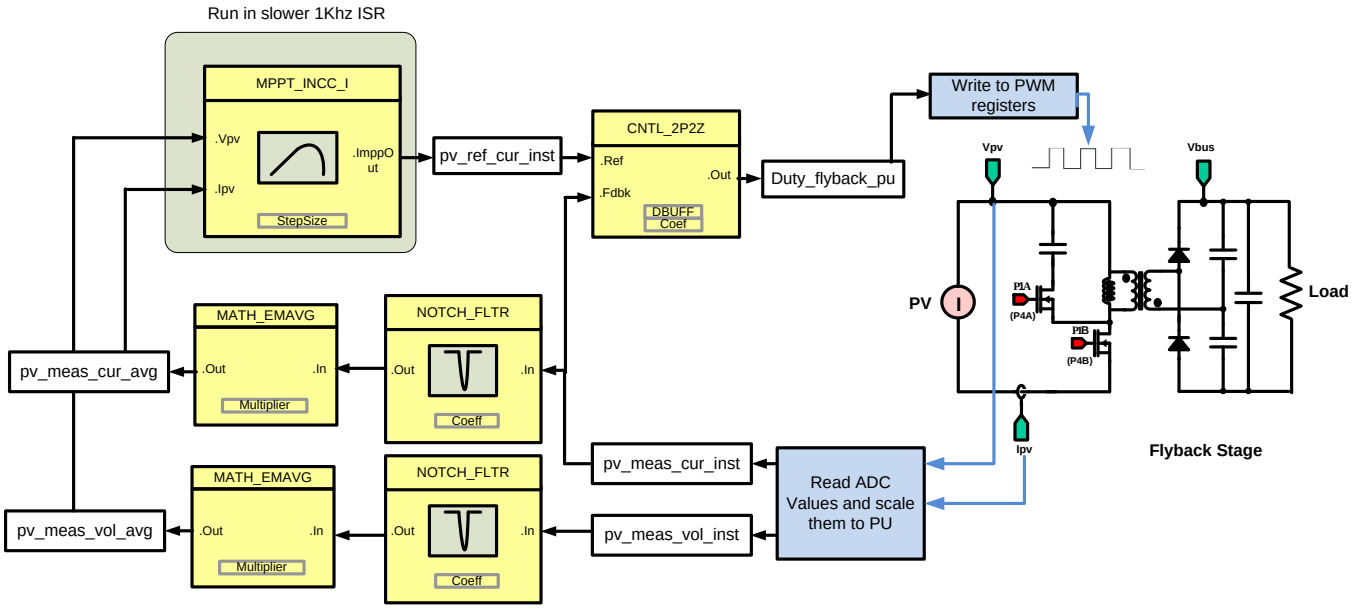



**Figure 39. Build 2 Software Diagram for Closed Current Loop Flyback Check with MPPT**

1. Follow the steps from the previous sections to import the micro inverter project into CCS.
2. Change the build level; open the *SolarMicroInv-Settings.h* file. Ensure the defines are as below:

```
#define INCR_BUILD 2
#define GRID_CONNECT 0
#define MPPT 1
```

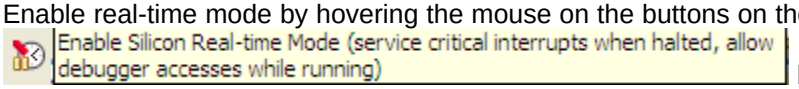
**NOTE:** When changing the incremental build option, always select Rebuild All.

3. Click the Project→Rebuild All button and watch the tools run in the build window.
4. Click on Target→Debug Active Project. The program loads into the flash. The user should now be at the start of main().

### 6.2.4.2 Debug Environment Windows

Click on View→Scripting Console to open the scripting console, and open the AddWatchWindowVars_IncrBuild2.js script located inside the project folder. This populates the watch window with the appropriate variables needed to debug the system and the appropriate Q formats. Click on the Continuous Refresh button  on the watch window to enable the continuous update of values from the controller.

#### 6.2.4.3 Using Real-Time Emulation

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

button.
2. A message box may appear. If so, select YES to enable debug events. This sets bit 1 (DGBM bit) of the status register 1 (ST1) to 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.
3. Click on the Continuous Refresh button  for the watch view.

#### 6.2.4.4 Run the Code

1. Run the project by clicking on  .
2. At this stage, switch on the PV panel emulator or connect the panel to the board.
3. In the watch view, check the value of Gui_Vpv, which displays the panel open circuit voltage.
4. Clear the flyback stage trip by writing a 1 to the ClearFlyBacktrip variable. EPwm3Regs.TZFLG.all goes to zero as soon as the trip is cleared.
5. Set MPPT_ENABLE to 1, and observe the input PV current slowly rise to the MPPT point.
6. This verifies the MPPT algorithm operation.

**CAUTION**

The user must ensure that the DC bus voltage at the output of the flyback must never exceed >400 V, and must connect the appropriate load to match this condition. Caution must be taken when dealing with PV panels, as they are energized source.

(x)= Variables | Expressions | Registers

| Expression | Type | Value | Address |
|---|---|---|---|
| (x)= ClearInvTrip | int | 0 | 0x00008015@Data |
| (x)= RlyConnect | int | 0 | 0x00008010@Data |
| (x)= inv_Iset | long | 0.0 (Q-Value(24)) | 0x00008040@Data |
| (x)= CloseIloopInv | int | 0 | 0x00008013@Data |
| (x)= EPwm1Regs.TZFLG.all | unsigned int | 4 | 0x00006816@Data |
| (x)= EPwm2Regs.TZFLG.all | unsigned int | 4 | 0x00006856@Data |
| (x)= Gui_Prms | long | 0.0 (Q-Value(6)) | 0x0000804C@Data |
| (x)= Gui_Vrms | long | 0.0 (Q-Value(6)) | 0x00008050@Data |
| (x)= Gui_Irms | long | 0.0 (Q-Value(12)) | 0x00008052@Data |
| (x)= ClearFlyBackTrip | int | 0 | 0x00008016@Data |
| (x)= duty_flyback_pu | long | 0.241617918 (Q-Value(24)) | 0x00008044@Data |
| (x)= pv_ref_cur_inst | long | 0.138372004 (Q-Value(24)) | 0x00008034@Data |
| (x)= pv_meas_cur_inst | long | 0.1369922161 (Q-Value(24)) | 0x00008026@Data |
| (x)= EPwm3Regs.TZFLG.all | unsigned int | 0 | 0x00006896@Data |
| (x)= Gui_Vpv | long | 29.28743935 (Q-Value(20)) | 0x0000805A@Data |
| (x)= Gui_Ipv | long | 2.05637455 (Q-Value(20)) | 0x00008058@Data |
| (x)= Gui_Vbus | long | 192.4721584 (Q-Value(20)) | 0x00008054@Data |
| (x)= MPPT_ENABLE | int | 1 | 0x00008018@Data |

**Figure 40. Build Level 2 Flyback Closed Current Loop with MPPT Watch Expressions**

7. This completes the flyback closed current loop check with MPPT. Enter 0 for MPPT_ENABLE, then 0 for pv_ref_cur_inst in the watch window. Disconnect the panel emulator or panel from the board.

**CAUTION**

Take appropriate precaution when removing the Panel as it is an energized source.

8. 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  .
9. Close the CCS debug session by clicking on Terminate Debug Session  (Target→Terminate All).

### 6.2.5 Inverter Closed Current Loop without GRID

#### *6.2.5.1 Procedure*

In this build, the closed current loop operation of the inverter stage is tested. For this a HV DC power supply is connected at the input of the inverter stage, and a resistive load is connected at the output, as shown in Figure 41. Keep the HV supply off, but set the connections as show below. The software structure used is illustrated in Figure 42.

**CAUTION**

High voltage; proceed with extreme caution. Refer to your organization's safety procedure for handling high voltages.

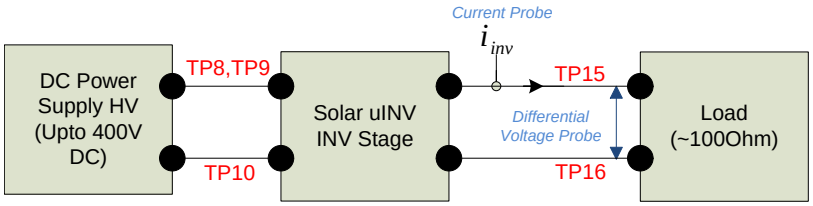



**Figure 41. Build Level 2 Power Stage Connections for Inverter Closed Current Loop Check without Grid**

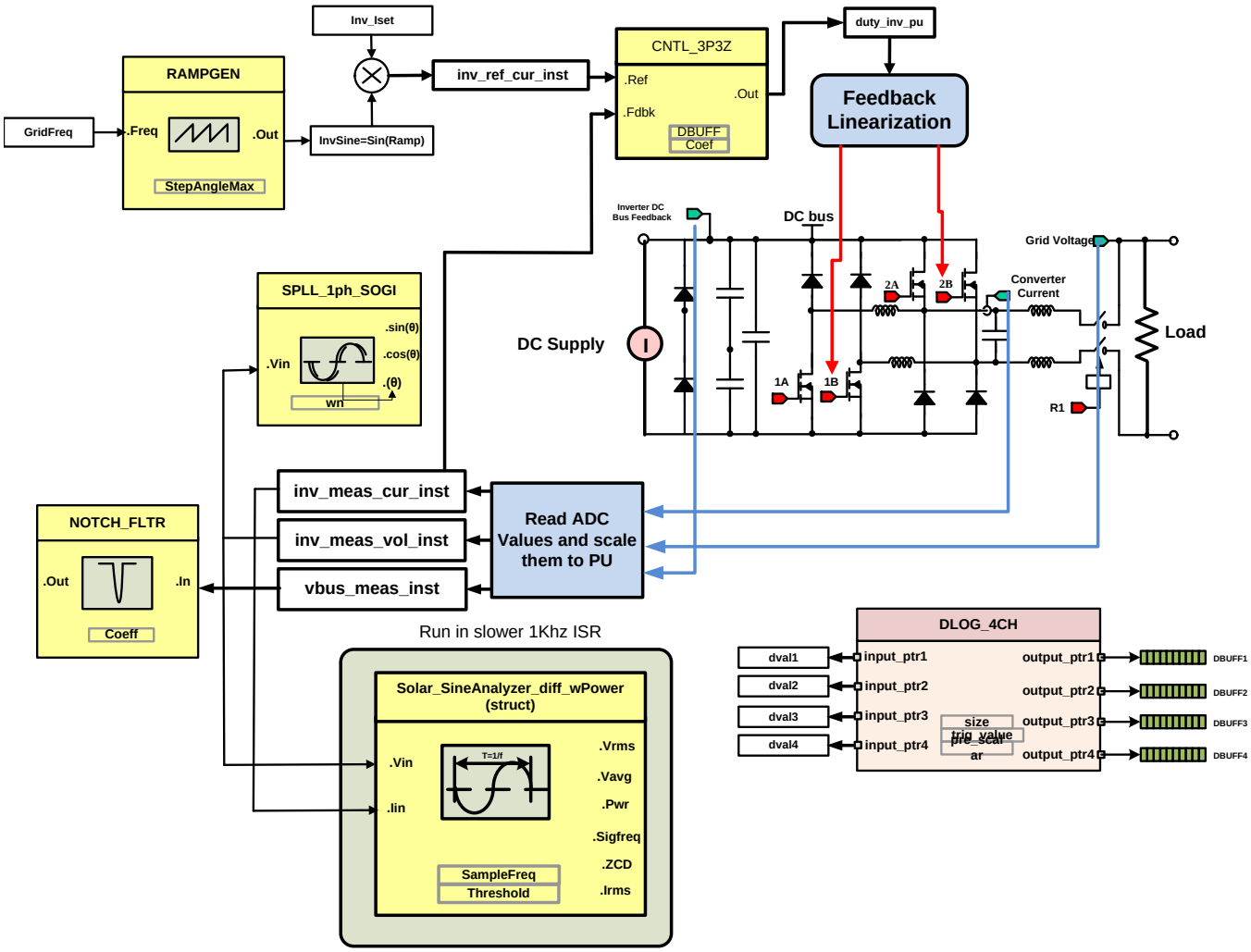



**Figure 42. Build Level 2 Software Diagram for Inverter Closed Current Loop Check without Grid**

1. Follow the steps from the previous sections to import the micro inverter project into CCS.
2. First change the build level; open the *SolarMicroInv-Settings.h* file. Ensure the #defines are as below:

```
#define INCR_BUILD 2
#define GRID_CONNECT 0
#define MPPT 0
```

**NOTE:** When changing the incremental build option, always select Rebuild All.

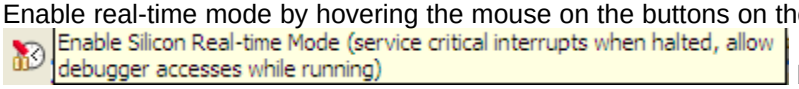
3. Click the Project > Rebuild All button and watch the tools run in the build window.
4. Click on Target > Debug Active Project. The program loads into the flash. The user should now be at the start of main().

### 6.2.5.2 Debug Environment Windows

1. Click on View→Scripting Console to open the scripting console, and open the AddWatchWindowVars_IncrBuild2.js script located inside the project folder. This populates the watch window with the appropriate variables needed to debug the system and the appropriate Q formats.

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

#### 6.2.5.3 Using Real-Time Emulation

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

button.
2. A message box may appear. If so, select YES to enable debug events. This sets bit 1 (DGBM bit) of status register 1 (ST1) to 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.
3. Click on the Continuous Refresh button  for the watch view.

#### 6.2.5.4 Run the Code

1. Run the project by clicking on  .
2. In the watch view, check the value of Gui_Vbus to display the input voltage at the inverter bus, which is zero.
3. Turn on the HV DC supply and increase the input voltage to around 300 V, and verify the reading from the watch expressions of the DC bus to match the input.
4. Connect the relay by writing a 1 to the RlyConnect variable in the watch window
5. Set the current command for the inverter by writing a pu value in the variable inv_Iset. For example, write _IQ24(0.1).
6. Clear the inverter trip by writing a 1 to ClearInvTrip.
7. This starts the inverter closed current loop operation. Gradually increase the current command to 0.25.
8. The closed loop operation can be further verified by viewing the DBUFF3 and DBUFF4 values in the graph window as they display the reference current instantaneous and measured current instantaneous. Real time graphs can be added using the procedure outlined in the open loop test for the inverter. Figure 43 shows the measured and reference current for a 0.25 pu current command. Observe the graphs change by changing the inv_Iset command. Ensure the load power rating matches the power going into the resistor.

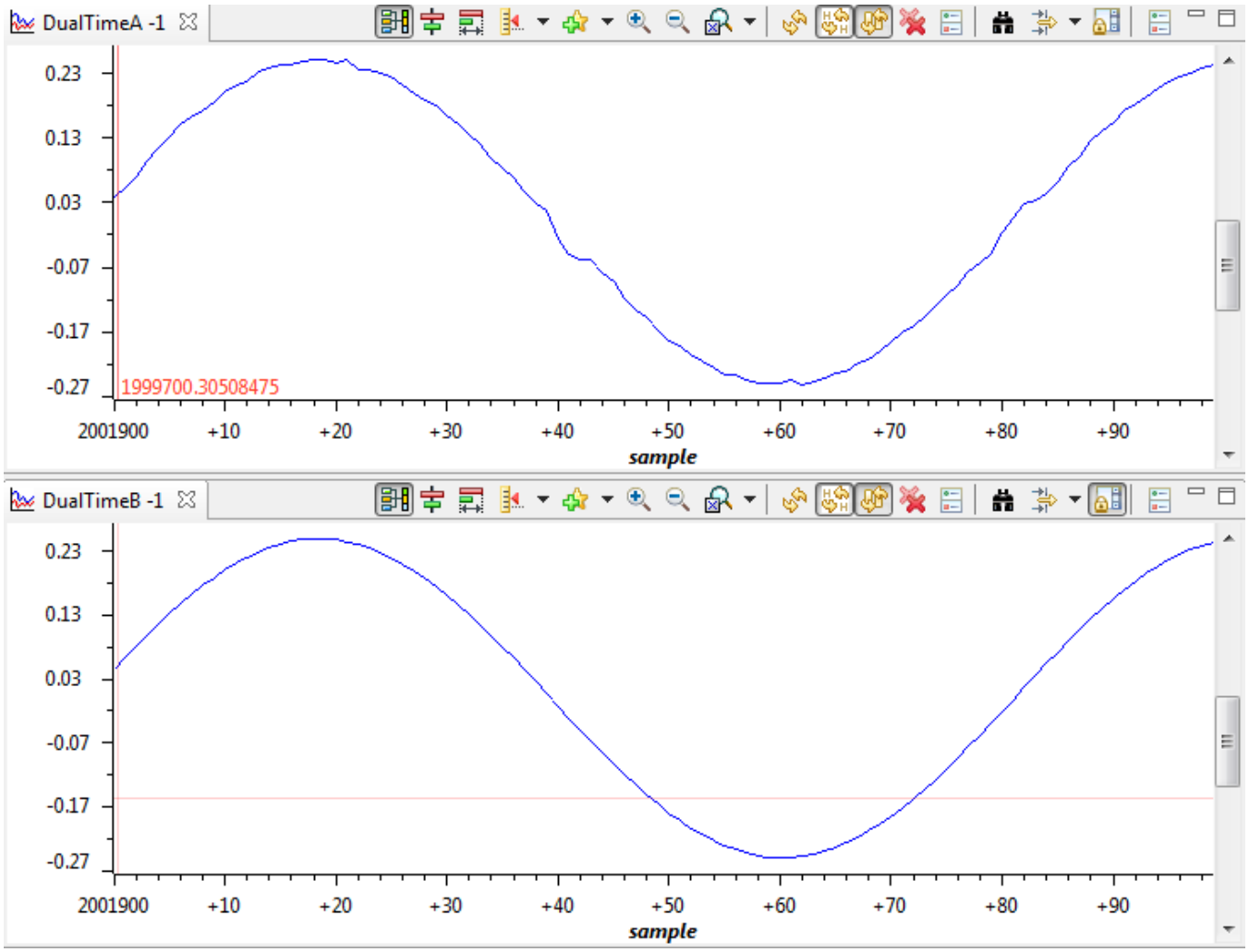



**Figure 43. CCS Graph Window of Closed Current Loop Operation**

9. This completes the closed current loop test for the inverter. Set the current command i.e. inv_Iset to zero.
10. Disconnect the inverter from the load by writing a 0 to RlyConnect.
11. Reduce the DC bus to zero.
12. Fully halting the MCU when in real-time mode is a two-step process. First, halt the processor by using the Halt button on the toolbar , or by using Target > Halt. Then, take the MCU out of real-time mode by clicking on . Finally, reset the MCU .
13. Close the CCS debug session by clicking on Terminate Debug Session (Target→Terminate All).

### 6.2.6 Inverter Closed Current Loop with GRID

#### *6.2.6.1 Procedure*

In this build, the closed current loop operation for the PV inverter is tested when a AC source or grid is connected at the output. Connect the DC power supply , AC supply, and a resistor as shown in Figure 44. Keep the DC and AC power supply off at this point. As in this test, the panel is not used as it is is a truly floating power supply; the ground loop from the DC power supply and the AC source must be disconnected.

**CAUTION**

Refer to the equipment user's guide to ensure proper grounding and avoid any ground loops.

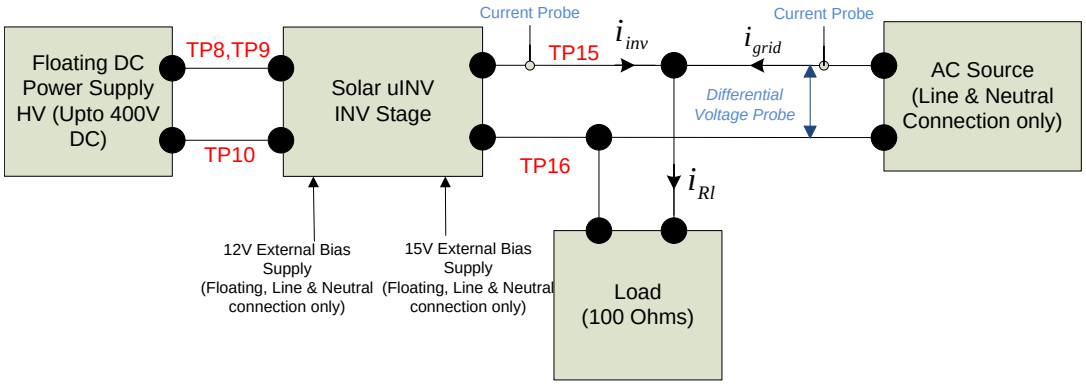



**Figure 44. Inverter Closed Loop Current Check with Grid Connection Build Level 2**

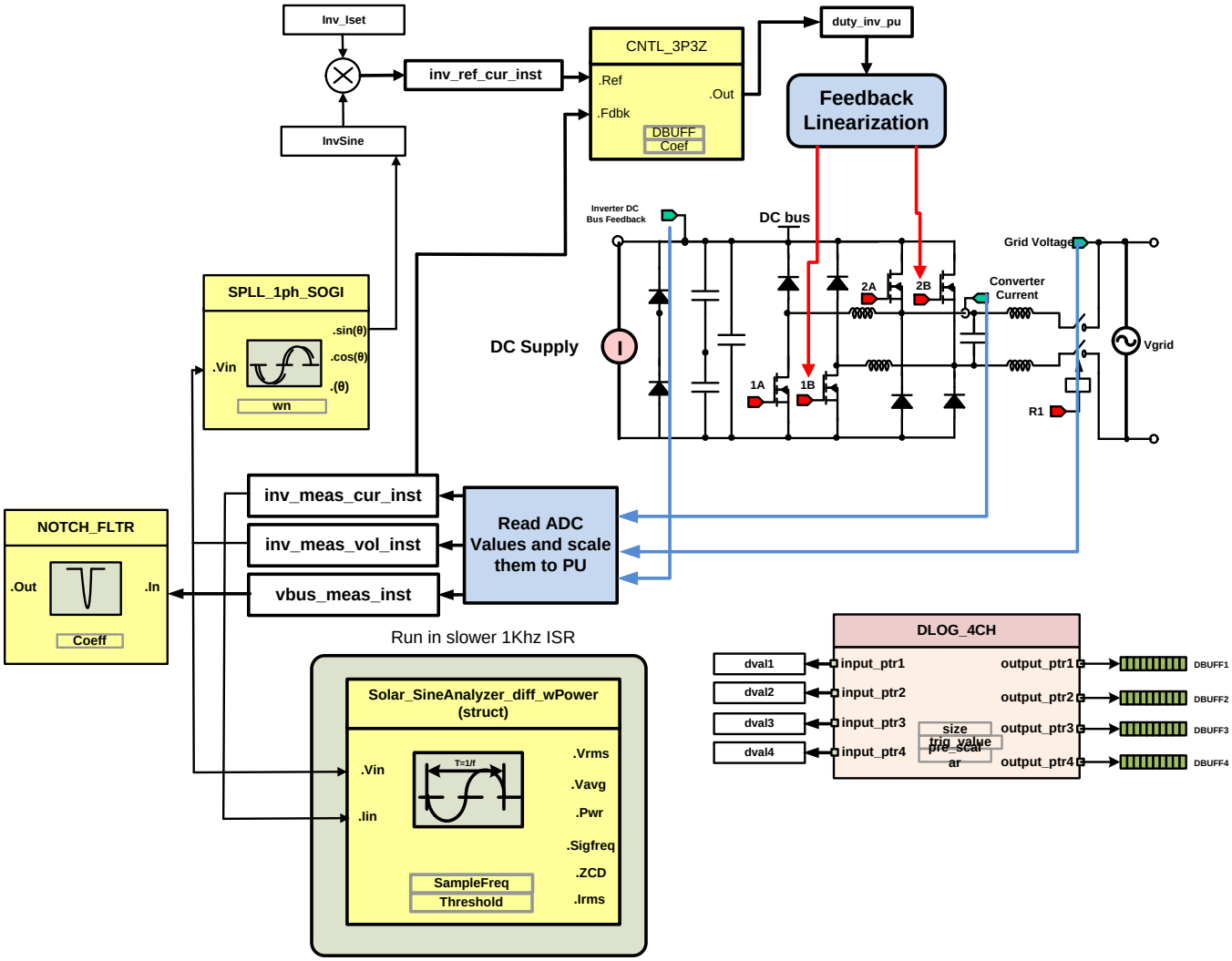



**Figure 45. Software Diagram for Closed Current Loop Inverter with Grid Connection**

1. Follow the steps from the previous sections to import the micro inverter project into CCS.
2. Change the build level by opening the *SolarMicroInv-Settings.h* file. Ensure the #defines are as below. Also modify the grid frequency to 60 Hz.

```
#define INCR_BUILD 2
#define GRID_CONNECT 1
#define MPPT 0
#define GRID_FREQ 60
```

**NOTE:** When changing the incremental build option, always select Rebuild All.

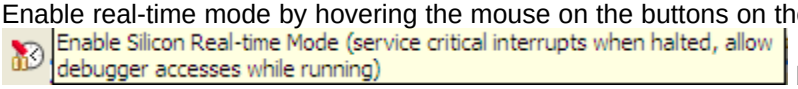
3. Click the Project→Rebuild All button and watch the tools run in the build window.
4. Click on Target→Debug Active Project. The program loads into the flash. The user should now be at the start of main().

#### 6.2.6.2 Debug Environment Windows

1. Click on View→Scripting Console to open the scripting console, and open the AddWatchWindowVars_IncrBuild2.js script located inside the project folder. This populates the watch window with the appropriate variables needed to debug the system and the appropriate Q formats.

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

#### 6.2.6.3 Using Real-time Emulation

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

button.
2. A message box may appear. If so, select YES to enable debug events. This sets bit 1 (DGBM bit) of status register 1 (ST1) to 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.
3. Click on the Continuous Refresh button  for the watch view.

#### 6.2.6.4 Run the Code

1. Run the project by clicking on  .
2. In the watch view, check the value of Gui_Vrms and Gui_Vbus to display the input voltage at the inverter bus and the AC voltage source rms value at the micro inverter output. These should be close to zero at this stage.
3. Slowly increase the input voltage to around 300 V, and verify that Gui_Vbus reads 300 V.
4. Turn the AC source on, and ensure the frequency is 60 Hz and rms voltage is 110.
5. Gui_Vrms will show ~110 V, and the power drawn from the DC power supply will be close to 120 W.
6. Set the current command for the inverter by writing a pu value in the variable inv_Iset. For example, write _IQ24(0.1).
7. Connect the relay by writing a 1 to the RlyConnect variable in the watch window.
8. Clear the inverter trip by writing a 1 to ClearInvTrip.
9. This starts the inverter, feeds current into the load, and drops the power drawn from the AC load. With 0.1 pu for the inv_Iset, the power drawn from the load will drop to around 66 W.

**CAUTION**

The AC source cannot take reverse power flow, therefore do not increase the inv_Iset such that the micro inverter board feeds power into the AC source, as this can damage the AC power supply. The maximum value for the setup described above is around 0.2 pu.

10. The closed loop operation can further be verified by viewing the DBUFF3 and DBUFF4 in the graph window. DBUFF3 and DBUFF4 display the reference current instantaneous and measured current instantaneous. Real time graphs can be added using the procedure outlined in Section 6.1.2.
11. The inv_Iset, maximum 0.2 pu for the setup described above, can be changed and the performance of the grid-connected closed current loop inverter verified.
12. This completes the closed current loop with grid test for the inverter. To stop the operation, put RlyConnect to 0.
13. Set the current command, such as inv_Iset, to zero.
14. Reduce the AC voltage to zero.
15. Reduce the DC bus to zero.
16. Fully halting the MCU when in real-time mode is a two-step process. First, halt the processor by using the Halt button  on the toolbar, or by using Target > Halt. Then, take the MCU out of real-time mode by clicking on  . Finally reset the MCU by clicking on  .
17. Close the CCS debug session by clicking on Terminate Debug Session  (Target→Terminate All).

## 6.3 Build = 3

### 6.3.1 Objective

The objective of this build is to run the full PV inverter system with closed current loop and DC bus voltage control. To connect the PV inverter to grid, a precise state machine must be followed to start the flyback stage, connect the relay, and start the inverter. The software must detect the grid frequency and adjust the DC bus voltage regulation parameters. Figure 46 illustrates the state machine used for the PV inverter system. Two build options are provided; one with MPPT and one without. This enables the user to tune the DC bus voltage controller without MPPT, and enables the use of a DC power supply at the input instead of a PV panel.

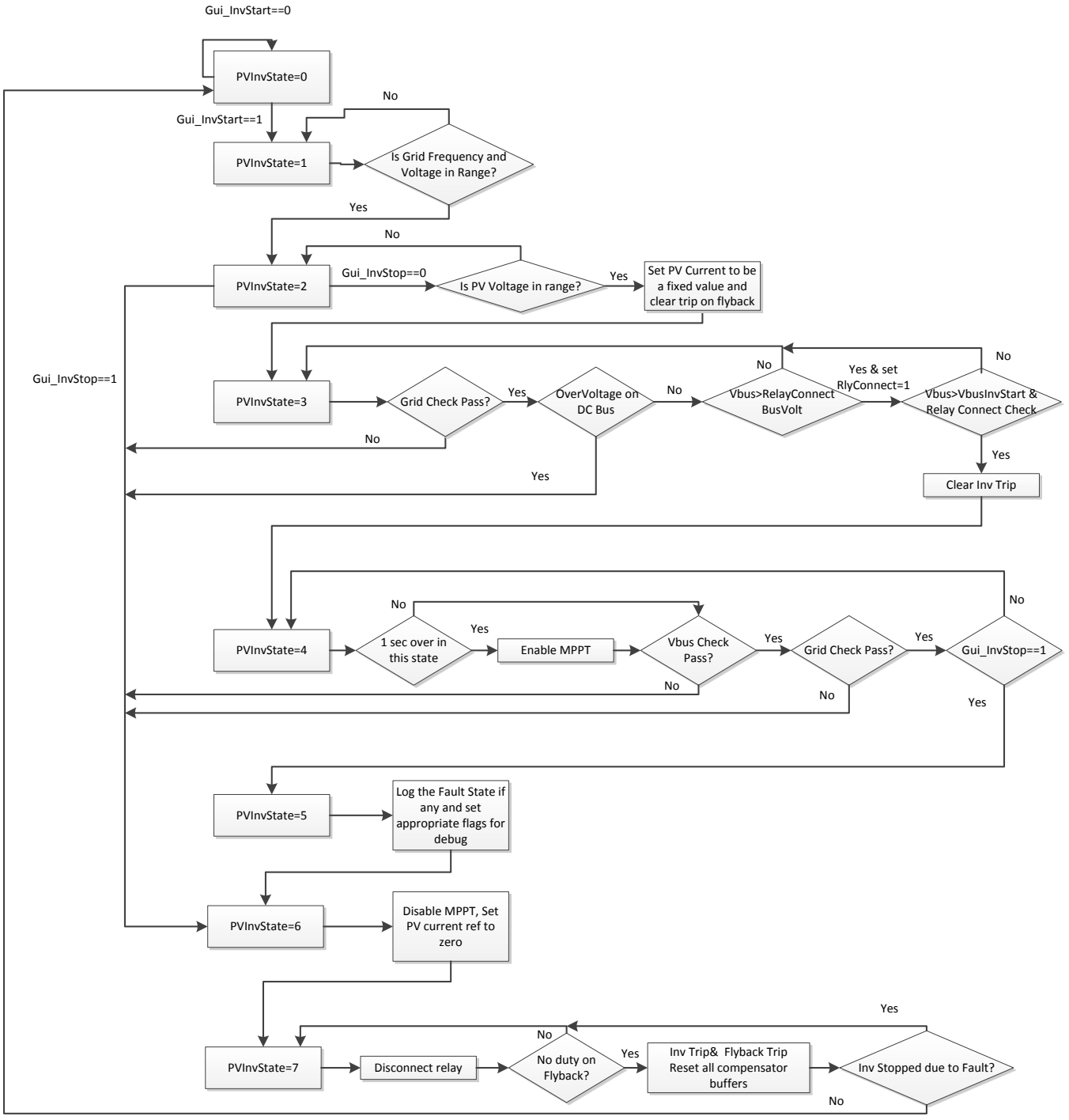



**Figure 46. State Machine for Grid Connection**

### 6.3.2 Full PV Inverter System Check without MPPT

#### 6.3.2.1 Overview

This test checks the full PV inverter system build. A DC power supply is used at the input instead of a panel source. Figure 47 shows the hardware setup with the two stages connected.

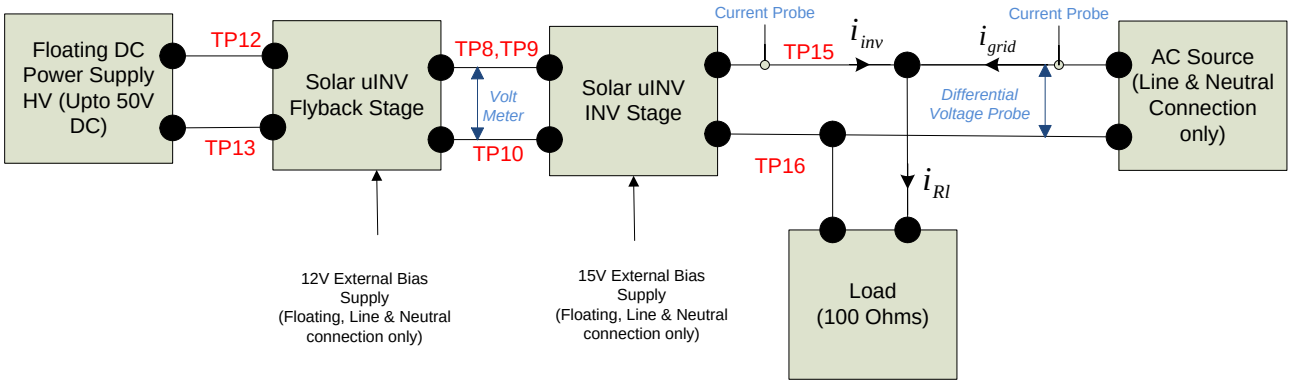



**Figure 47. Full PV Inverter System Hardware Check Build 3 without MPPT**

1. Follow the steps from the previous sections to import the micro inverter project into CCS.
2. Change the build level by opening the *SolarMicroInv-Settings.h* file. Ensure the #defines are as below . Also modify the grid frequency to 60 Hz.

```
#define INCR_BUILD 3
#define GRID_CONNECT 1
#define MPPT 0
#define GRID_FREQ 60
```

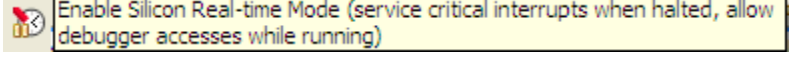
**NOTE:** When changing the incremental build option, always select Rebuild All.

3. Click the Project > Rebuild All button and watch the tools run in the build window.
4. Click on Target > Debug Active Project. The program loads into the flash. The user should now be at the start of main().

#### 6.3.2.2 Debug Environment Windows

Click on View→Scripting Console to open the scripting console, and open the AddWatchWindowVars_IncrBuild3.js script located inside the project folder. This populates the watch window with the appropriate variables needed to debug the system and the appropriate Q formats. Click on the Continuous Refresh button  on the watch window to enable the continuous update of values from the controller.

#### 6.3.2.3 Using Real-Time Emulation

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

button.
2. A message box may appear. If so, select YES to enable debug events. This sets bit 1 (DGBM bit) of status register 1 (ST1) to 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.
3. Click on the Continuous Refresh button  for the watch view.

#### 6.3.2.4 Run the Code

1. Run the project by clicking on  .
2. In the watch view, check the value of Gui_Vrms and Gui_Vpv to display the input voltage at the flyback stage and the AC voltage source rms value at the micro inverter output.
3. Turn the AC source on, and ensure the frequency is 60 Hz and rms voltage is 110.
4. Gui_Vrms will show around 110 V. If the resistive load is used, the power drawn from the DC power supply will be around 120 W.
5. Turn on the DC power supply and increase the voltage to 30 V. Verify this by inspecting the Gui_Vpv variable in the watch window.
6. Turn on the inverter by writing a 1 to Gui_InvStart.
7. This triggers the state machine of the inverter, and the PVInverterState goes to PVInverterActive.

Variables | Expressions | Registers

| Expression | Type | Value | Address |
|---|---|---|---|
| Gui_InvStart | unsigned int | 0 | 0x00008015@Data |
| Gui_InvStop | unsigned int | 0 | 0x00008014@Data |
| PVInverterState | enum PVInverterStates | PVInverterActive | 0x00008017@Data |
| FaultFlag | enum Faults | NoFault | 0x0000801E@Data |
| BusOverVoltageTrip | int | 0 | 0x0000801D@Data |
| BusUnderVoltageTrip | int | 0 | 0x0000801C@Data |
| Gui_Vpv | long | 29.23216438 (Q-Value(20)) | 0x0000808A@Data |
| Gui_Ipv | long | 2.241826057 (Q-Value(20)) | 0x000080A8@Data |
| Gui_Vbus | long | 274.5640392 (Q-Value(20)) | 0x00008088@Data |
| Gui_Prms | long | 56.4375 (Q-Value(6)) | 0x0000807E@Data |
| Gui_Vrms | long | 112.4375 (Q-Value(6)) | 0x00008082@Data |
| Gui_Irms | long | 0.5168457031 (Q-Value(12)) | 0x00008070@Data |
| EPwm3Regs.TZFLG.all | unsigned int | 0 | 0x00006896@Data |
| EPwm1Regs.TZFLG.all | unsigned int | 0 | 0x00006816@Data |
| EPwm2Regs.TZFLG.all | unsigned int | 0 | 0x00006856@Data |
| pv_ref_cur_inst | long | 0.1499999762 (Q-Value(24)) | 0x00008056@Data |
| pv_meas_cur_inst | long | 0.1472461224 (Q-Value(24)) | 0x00008052@Data |
| *Add new expression* | | | |

**Figure 48. Watch Expression for Build 3, PV Inverter Check without MPPT**

8. In this build, a fixed current is commanded from the DC power supply. This can be increased by writing a new value to pv_ref_cur_inst. The DC bus is regulated at around 275 V. This is specified in the SolarMicroInv-Settings.h file, irrespective of the current command from the DC power supply.
9. To stop the inverter operation, write a 1 to Gui_InvStop.
10. Reduce the DC power supply to zero.
11. Remove the AC power source connected at the output of the micro inverter board.

**CAUTION**

There is residual voltage on the DC bus of the inverter. Monitor this voltage through the watch window and wait for it to decrease to the safe voltage level of around 3-4 V. This may take a few minutes.

12. This completes the PV inverter evaluation without MPPT. To stop the operation, put RlyConnect to 0.
13. Set the current command, such as inv_Iset, to zero.
14. Reduce the AC voltage to zero.
15. Reduce the DC bus to zero.

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

### 6.3.3 Full PV Inverter System Check with MPPT

#### *6.3.3.1 Overview*

This test checks the full PV inverter system build. A panel or panel emulator is used at the input of the flyback stage. Figure 49 shows the hardware setup with the two stages connected. If a panel emulator is used for the input source, set the connections as shown in . Typical characteristics programmed in a panel emulator can be:

1. Open circuit voltage 40 V
2. Maximum power point voltage 30 V
3. Short circuit current 5.2 Amps
4. Maximum power point current 4 Amps

If only a panel is available, do not connect the panel at this stage, but ensure the rating of the panel is comparable to the panel emulator settings described above. Use #16 wires for all connections. Also keep the AC source off. Connect the AC source as shown in Figure 49.

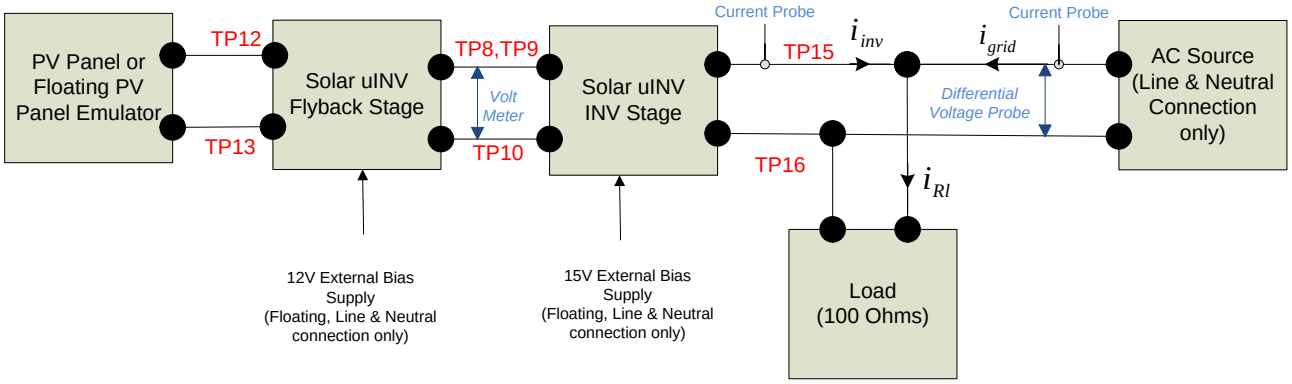



**Figure 49. Hardware Setup for Build level 4, PV Inverter System Check with MPPT**

#### *6.3.3.2 Procedure*

1. Follow the steps from the previous sections to import the micro inverter project into CCS.
2. Change the build level by opening the *SolarMicroInv-Settings.h* file. Ensure the defines are as below. Also modify the grid frequency to 60 Hz.

```
#define INCR_BUILD 3
#define GRID_CONNECT 1
#define MPPT 1
#define GRID_FREQ 60
```

**NOTE:** When changing the incremental build option, always select Rebuild All.

3. Click the Project→Rebuild All button and watch the tools run in the build window.
4. Click on Target→Debug Active Project. The program loads into the flash. The user should now be at the start of main().

#### 6.3.3.3 Debug Environment Windows

Click View→Scripting Console to open the scripting console and open the "AddWatchWindowVars_IncrBuild3.js" script located inside the project folder. This will populate the watch window with appropriate variables needed to debug the system and the appropriate Q formats. Click on the Continuous Refresh button on the watch window to enable continuous update of values from the controller.

#### 6.3.3.4 Using Real-time Emulation

1. Enable real-time mode by hovering the mouse on the buttons on the horizontal toolbar and clicking the Enable Silicon Real-time Mode (service critical interrupts when halted, allow debugger accesses while running) button.
2. A message box may appear. If so, select YES to enable debug events. This sets bit 1 (DGBM bit) of status register 1 (ST1) to 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.
3. Click on the Continuous Refresh button for the watch view.

#### 6.3.3.5 Run the Code

1. Run the project by clicking on .
2. In the watch view, check the value of Gui_Vrms and Gui_Vpv to display the input voltage at the flyback stage and the AC voltage source rms value at the micro inverter output.
3. Turn the AC source on, and ensure the frequency is 60 Hz and rms voltage is 110.
4. Gui_Vrms will show around 110 V. If the resistive load is used, the power drawn from the DC power supply will be around 120 W.
5. Turn on the panel emulator or connect the panel to the input of the board.
6. Turn on the inverter by writing a 1 to Gui_InvStart.
7. This triggers the state machine of the inverter, and the PVInverterState goes to PVInverterActive.

Variables | Expressions | Registers

| Expression | Type | Value | Address |
|---|---|---|---|
| Gui_InvStart | unsigned int | 0 | 0x00008015@Data |
| Gui_InvStop | unsigned int | 0 | 0x00008014@Data |
| PVInverterState | enum PVInverterStates | PVInverterActive | 0x00008017@Data |
| FaultFlag | enum Faults | NoFault | 0x0000801E@Data |
| BusOverVoltageTrip | int | 0 | 0x0000801D@Data |
| BusUnderVoltageTrip | int | 0 | 0x0000801C@Data |
| Gui_Vpv | long | 29.23216438 (Q-Value(20)) | 0x0000808A@Data |
| Gui_Ipv | long | 2.241826057 (Q-Value(20)) | 0x000080A8@Data |
| Gui_Vbus | long | 274.5640392 (Q-Value(20)) | 0x00008088@Data |
| Gui_Prms | long | 56.4375 (Q-Value(6)) | 0x0000807E@Data |
| Gui_Vrms | long | 112.4375 (Q-Value(6)) | 0x00008082@Data |
| Gui_Irms | long | 0.5168457031 (Q-Value(12)) | 0x00008070@Data |
| EPwm3Regs.TZFLG.all | unsigned int | 0 | 0x00006896@Data |
| EPwm1Regs.TZFLG.all | unsigned int | 0 | 0x00006816@Data |
| EPwm2Regs.TZFLG.all | unsigned int | 0 | 0x00006856@Data |
| pv_ref_cur_inst | long | 0.1499999762 (Q-Value(24)) | 0x00008056@Data |
| pv_meas_cur_inst | long | 0.1472461224 (Q-Value(24)) | 0x00008052@Data |
| *Add new expression* | | | |

**Figure 50. Watch Expressions for Build Level 3, PV Inverter System Check with MPPT**

8. Note that the current from the panel slowly increases to the MPP. If a PV panel emulator is used, different shading conditions can be tested and the transient behavior also evaluated. Note that the DC bus is regulated at around 275V. This is specified in the SolarMicroInv-Settings.h file, irrespective of the current command from the DC power supply. Figure 51 shows the waveform of grid voltage and inverter current fed into the grid when connected to a ~120 W MPP panel emulator.

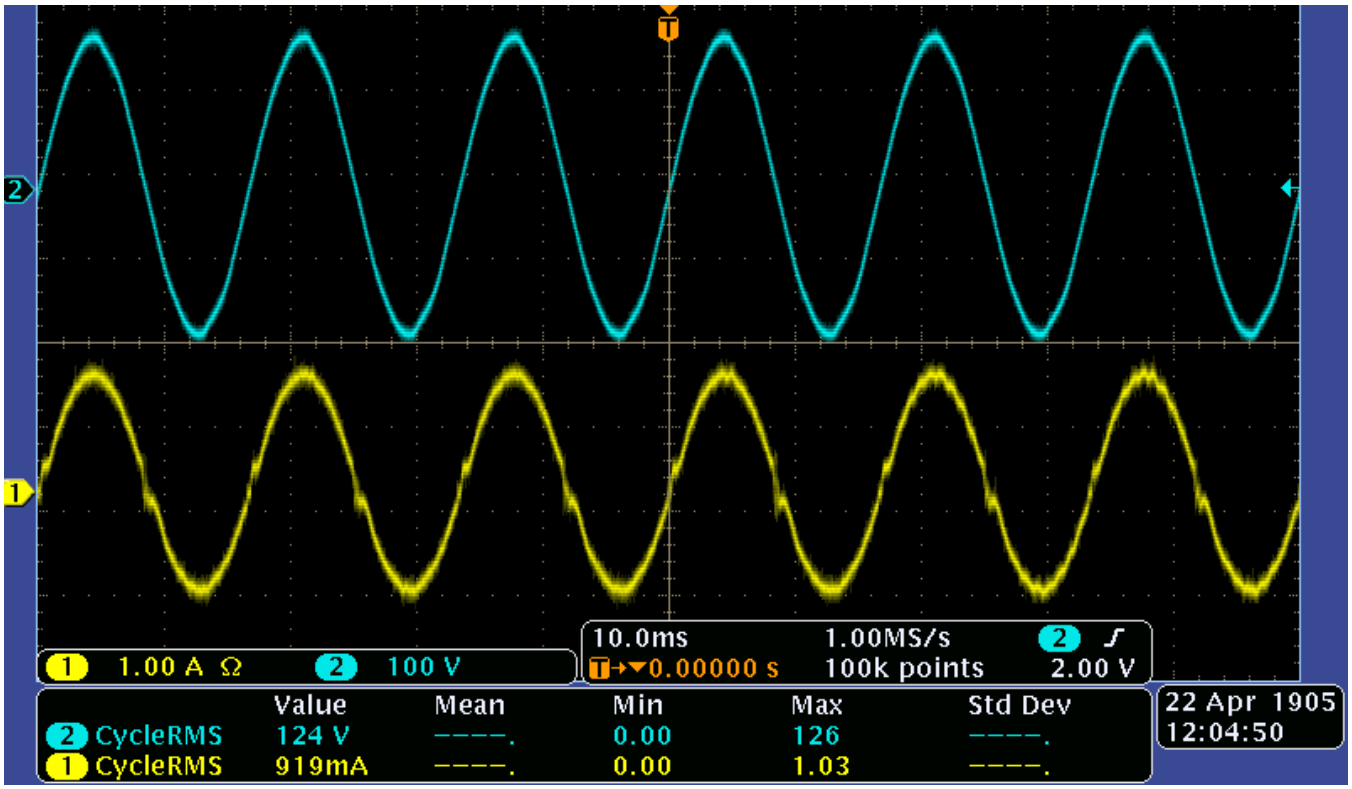



**Figure 51. Build Level 3, Grid Connection Check Voltage and Current Waveform Scope Capture**

Figure 52 is a capture of an inverter start and inverter stop command.

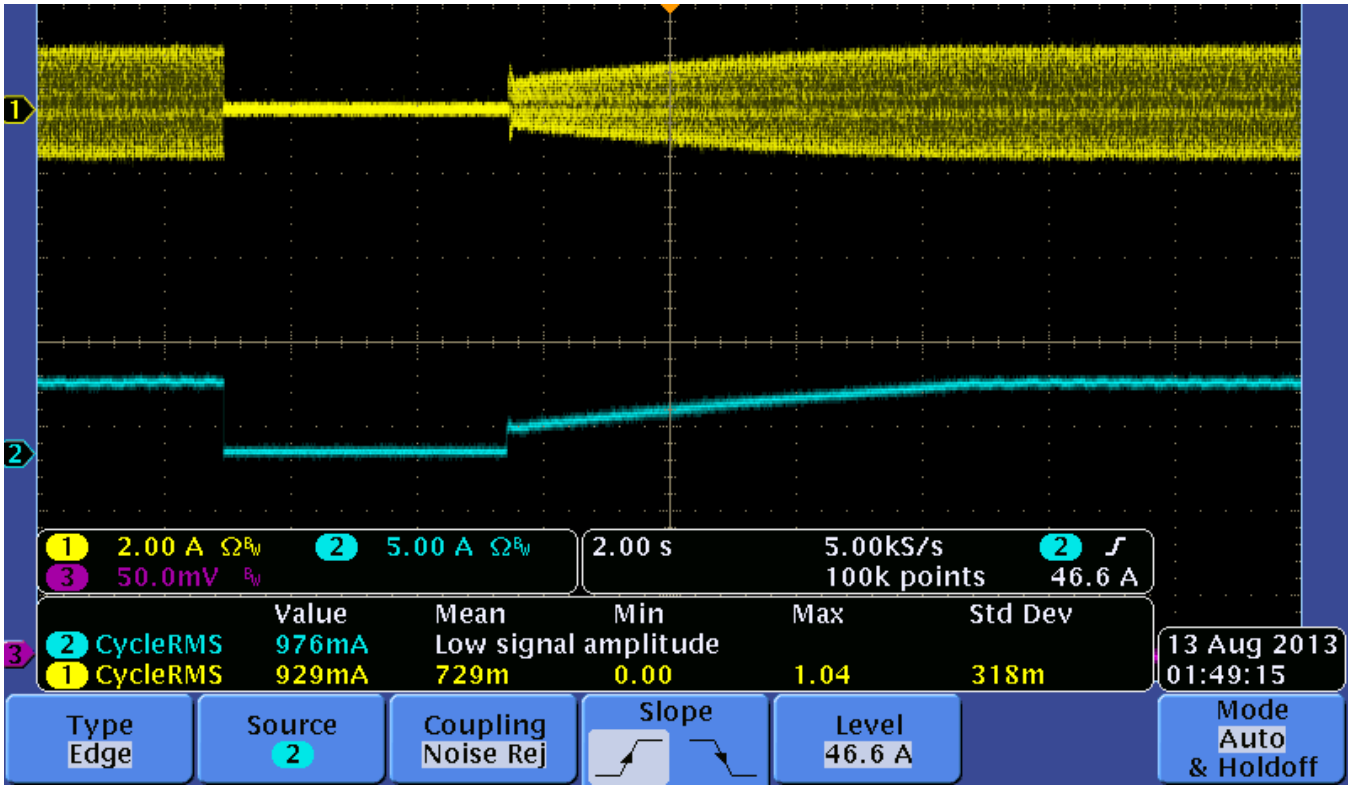



**Figure 52. Stop and Start Sequence of the PV Micro Inverter Scope Capture**

9. To stop the inverter operation, write a 1 to Gui_InvStop.
10. Reduce the DC power supply to zero.
11. Remove the AC power source connected at the output of the uInv board.

> **CAUTION**
>
> There is residual voltage on the DC bus of the inverter. Monitor this voltage through the watch window and wait for it to go down to a safe voltage level of around 3-4 V. This may take a few minutes.

12. This completes the PV inverter evaluation without MPPT. To stop the operation, put RlyConnect to 0.
13. Set the current command, such as inv_Iset, to zero.
14. Reduce the AC voltage to zero.
15. Reduce the DC bus to zero.
16. Fully halting the MCU when in real-time mode is a two-step process. First, halt the processor by using the Halt button on the toolbar, or by using Target > Halt. Then, take the MCU out of real-time mode by clicking on . Finally reset the MCU by clicking on .
17. Close the CCS debug session by clicking on Terminate Debug Session (Target→Terminate All).

# 7 Test Results

Table 5 and Table 6 show the converter performance at different voltage levels.

**NOTE:** The tests were performed with an AC source emulating the grid voltage.

**Table 5. Converter Performance at 120 Vrms / 60-Hz Grid Voltage**

| Vin (V) | Iin (A) | Pin (W) | Vbus (V) | Vgrid RMS (V) | Iout RMS (A) | Pout (W) | Efficiency (%) | THD (%) |
|---|---|---|---|---|---|---|---|---|
| 35.85 | 2.04 | 73.13 | 270 | 125.6 | 0.55 | 67.7 | 92.6 | 8 |
| 35.06 | 3.08 | 107.98 | 270 | 126.37 | 0.8 | 99.7 | 92.32 | 5.5 |
| 34.89 | 4.87 | 169.91 | 270 | 126.6 | 1.22 | 153.73 | 90.48 | 4 |
| 34.73 | 6.54 | 227.13 | 270 | 126.9 | 1.59 | 200.36 | 88.21 | 3 |

**Table 6. Converter Performance at 220 Vrms / 50-Hz Grid Voltage**

| Vin (V) | Iin (A) | Pin (W) | Vbus (V) | Vgrid RMS (V) | Iout RMS (A) | Pout (W) | Efficiency (%) | THD (%) |
|---|---|---|---|---|---|---|---|---|
| 34.91 | 4.06 | 141.73 | 362 | 220 | 0.62 | 132.3 | 93.34 | 10 |
| 34.75 | 5.54 | 192.52 | 360 | 220 | 0.82 | 178.83 | 92.89 | 6.7 |
| 34.59 | 7.02 | 242.82 | 359.2 | 220 | 1.02 | 223.86 | 92.19 | 5.5 |
| 34.33 | 9.23 | 316.87 | 371 | 220.1 | 1.31 | 286.33 | 90.36 | 4.1 |

# 8 References

For more information, refer to the following guides:

1. **Solar Micro Inverter-GUI-QSG** – A quick-start guide for a quick demo of the Solar Micro Inverter EVM using a GUI interface. ..*controlSUITE\development_kits\TMDSSOLARUINVKIT_version\~Docs\QSG_Solar_Micro_Inve rter_GUI_Rev1.0.pdf*
2. **Solar Micro Inverter_Rel-1.0-HWdevPkg** – A folder containing various files related to the Piccolo-B controller card schematics and the Micro Inverter schematic. *..\controlSUITE\development_kits\TMDSSOLARUINVKIT_version\Solar_Micro-Inverter_HWDevPkg*
3. **F28xxx User's Guides** – http://www.ti.com/f28xuserguides
4. **Software Phased-Locked Loop Design Using C2000™ Microcontrollers for Single Phase Grid Connected Inverter** (SPRABT3), July 2013.

# 9 About the Authors

**MANISH BHARDWAJ** is a Systems Application Engineer at Texas Instruments, responsible for developing system solutions for C2000 microcontrollers. 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. He can be reached at mbhardwaj@ti.com.

**SHAMIM CHOUDHURY** has been with Texas Instruments Inc. since December 1997, where he currently leads the C2000 Digital Power Application team. As a member of the C2000 System Applications team for many years, his areas of interest have been on digital control of power converters, switching power supplies, Power Factor Correction, UPS and Solar Inverters. Prior to joining TI, he spent two years at Alcatel, and three years at International Game Technology, as a Design Engineer working on switch mode power supplies. He received his M.S. degree in Electrical Engineering from Texas A&M University in College Station, TX, in December 1991. He can be reached at sach@ti.com.

## Revision History

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

| Changes from March 1, 2016 to June 30, 2017 (from A Revision (February 2016) to B Revision) | Page |
|---|---|
| • Added Test Resutls section. | 52 |

## IMPORTANT NOTICE FOR TI DESIGN INFORMATION AND RESOURCES

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

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

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

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

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

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

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

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

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