

**Digitally Controlled Solar Micro
Inverter using
C2000 Piccolo Microcontroller**

*CCS User Guide
Version 1.0
24 April 2014*

Abstract

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

Note: The micro inverter board design follows a control card concept; hence a different control card can also be used depending on the system requirements.

CAUTION: There are high voltages present on the TI Solar Micro Inverter board. It should only be handled by experienced power supply professionals in a lab environment ONLY. There may also be some components with high temperature when the board is powered on. So the user should take appropriate safety

Contents

Abstract.....	2
1. Introduction.....	4
2. Hardware and Control	7
2.1 Kit Contents.....	7
2.2 Clamped Fly-back DC-DC Stage.....	7
2.3 DC-AC Inverter Stage.....	9
2.3.1 Inverter Phase Locked Loop.....	11
2.3.2 Inverter current control loop.....	14
2.4 Solar Micro Inverter Electrical Specifications.....	16
3. Software Implementation	17
3.1 Project Framework	17
3.2 Project Dependencies & Resources	19
3.3 Control Description	19
3.3.1 DC-DC Flyback with MPPT Control Software.....	19
3.3.2 DC-AC Single Phase Inverter Control Software	20
3.4 DC-DC and DC-AC Integration.....	22
4. Software Setup	23
5. Hardware Setup.....	27
5.1 Equipment Required.....	27
5.2 Jumper & Basic Board Setup	27
5.3 Using Internal Bias Power Supply.....	29
6. Software Builds.....	30
5.1 BUILD = 1	31
5.1.1 Flyback Open Loop Check	31
5.1.2 Inverter Open Loop Check.....	34
5.2 BUILD = 2	40
5.2.1 Flyback Closed Current Loop without MPPT	40
5.2.2 Flyback Closed Current Loop with MPPT	43
5.2.3 Inverter Closed Current Loop without GRID	46
5.2.4 Inverter Closed Current Loop with GRID	49
5.3 BUILD = 3	53
5.3.1 Full PV Inverter System Check without MPPT.....	55
5.3.2 Full PV Inverter System Check with MPPT.....	57
6. References	62

1. Introduction

Energy from renewable sources, such as solar and wind, has been gaining interest as the world's power demand increases and non-renewable resources deplete. A large component of energy expended in the world is used by industries and houses that are connected to the electrical grid. Thus attempts are being made to raise the percentage of energy sourced from renewable sources into the grid. Photovoltaic (PV) energy sources are considered quintessential factor in increasing the renewable content due to their ubiquitous nature and extended life time because of absence of any moving parts.

The PV panel is a non-linear DC source hence an inverter is required to feed current into the grid and a maximum power tracking algorithm is necessary to maximize power from the panel. Thus the key challenge in 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 tied together to a single inverter stage, these are called string inverters. Such PV inverter architecture suffer from partial shading problems hence an emerging architecture is to include an inverter on each panel, Figure 1. The localized MPPT at each panel improves the performance of the system under partial shading and unmatched panels conditions. Texas Instruments C2000 microcontroller family, with its enhanced peripheral set and optimized CPU core for control tasks, is ideal for controlling the power conversion.

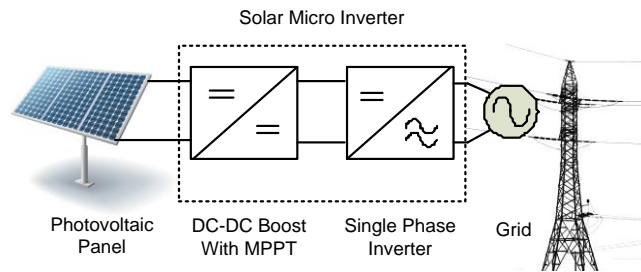


Figure 1 Grid tied PV inverter

This user-guide presents an overview of the hardware and the detailed software implementation of a PV micro inverter system using C2000 MCU on Texas Instrument's solar micro inverter kit (TMDSSOLARUINVKIT). All 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 fly-back converter with secondary voltage multiplier and, (2) a DC-AC inverter. A block diagram of the kits is shown in Figure 2.

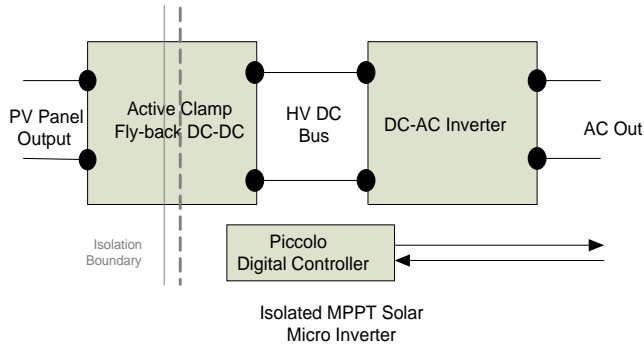


Figure 2 Micro inverter block diagram

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, i.e., 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. Then a current control loop for the fly-back converter ensures that the converter input current tracks the MPPT reference current. The fly-back converter also provides high frequency isolation for the DC-DC stage. The output of the fly-back 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 controlled sine wave current into the grid. The inverter also implements grid synchronization in order to maintain its current waveform locked to phase and frequency of the grid voltage. Figure 4 illustrates the control scheme for a complete grid connected PV micro inverter. All these 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 is able to implement complete digital control of such micro inverter system. Figure 4 shows a simplified diagram of different stages present on the Solar Micro Inverter kit.

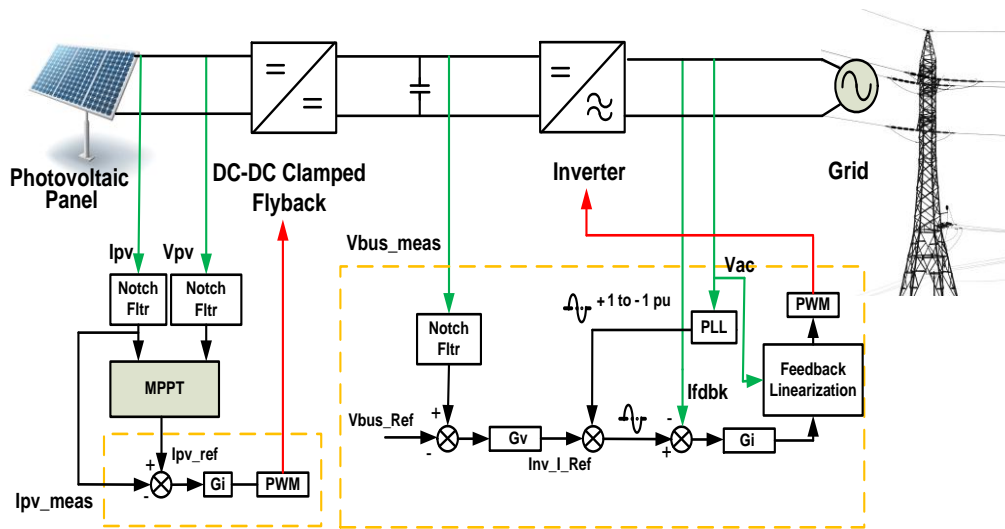


Figure 3 Control of grid connected solar micro inverter

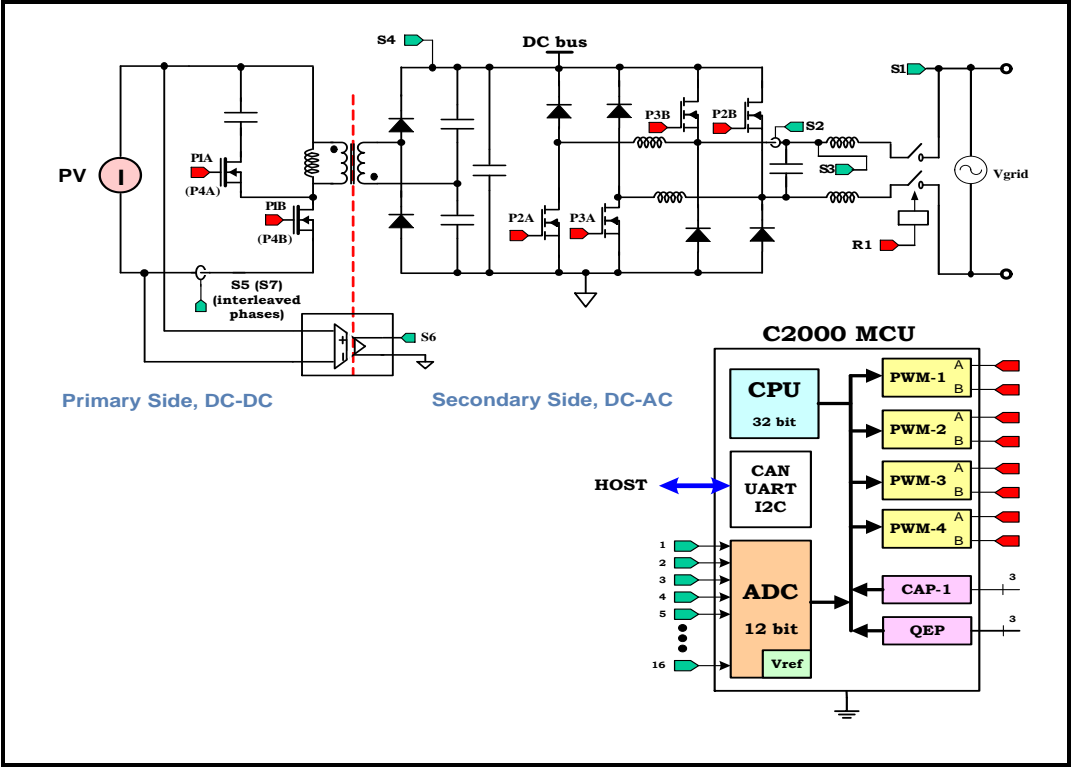


Figure 4 Solar micro inverter kit power stage diagram

2. Hardware and Control

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

2.1 Kit Contents

The kit contents include:

1. TMDSSOLARUINVKIT Base Board
2. TMDCNCD28035ISO control card
3. USB mini to A cable
4. +15V DC external isolated bias supply
5. +12V DC external isolated bias supply
6. 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 fly-back DC-DC power stage implemented on the micro inverter kit.

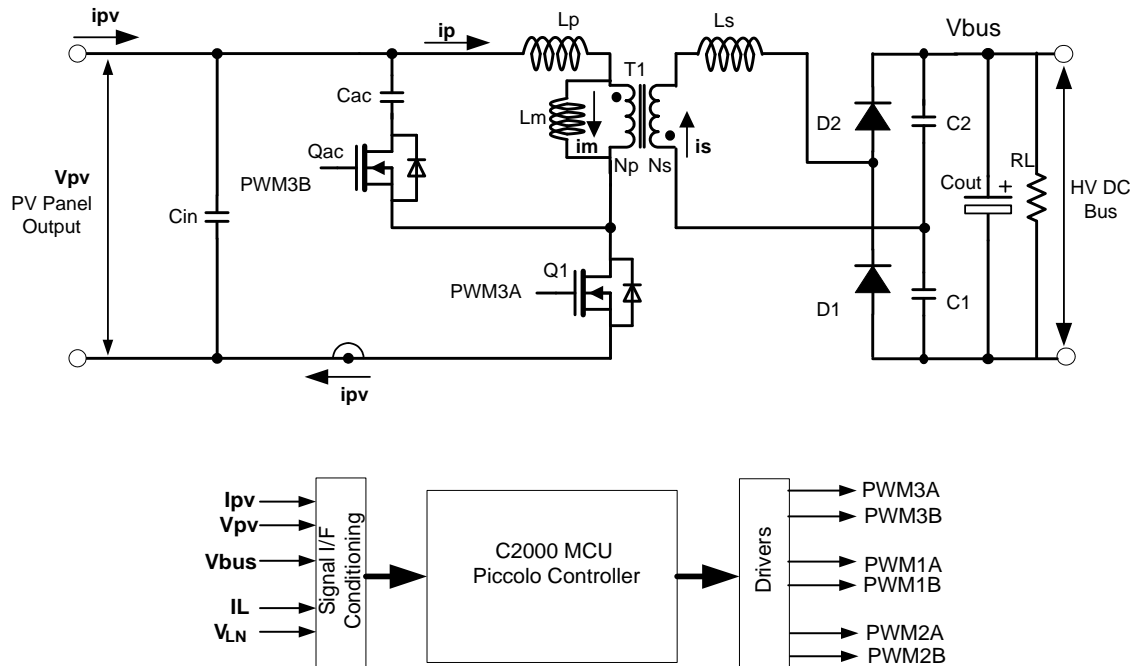


Figure 5 MPPT fly-back DC-DC converter control using C2000 MCU

The PV panel output voltage, V_{pv} , 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 all the interface signals needed for full control of this 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 (V_{pv}), the flyback input current (I_{pv}) and the flyback output voltage (V_{bus}). Since the MCU is ground referenced to secondary ground (-ve terminal of high voltage DC bus V_{bus}) panel output voltage V_{pv} is sensed with a differential isolation amplifier (AMC1200S). I_{pv} is sensed with a current transformer. As a second option I_{pv} is also sensed with an isolated current sensor such as ACS712ELCTR-20A. The sensed signals I_{pv} and V_{pv} 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 worldwide voltage range of 90V_{rm}~260V_{rms}.

Figure 6 shows the DC-DC converter control loop. This uses single current control loop. A Maximum Power Point Tracking (MPPT) algorithm determines the set point current i.e., the reference panel current I_{pv_ref} . The current control loop ensures that the DC/DC input current is regulated at MPPT reference level by adjusting the duty cycles of the power switches Q1 and Qac.

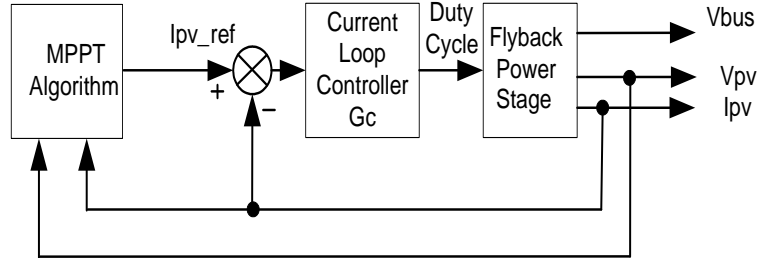


Figure 6 MPPT DC-DC converter control loop

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

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

All the time critical functions related to the DC-DC control loops are 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 C2000 MCU. The DC-DC output voltage, V_{bus} , is applied to the inverter stage input. The inverter output is connected to grid.

The inverter is controlled as a current source and essentially consists of two DC-AC buck converters each operating in one of the half cycles of the AC line voltage V_{LN} . MOSFETs Q9, Q3, diode D3 and inductor L4 together form the first buck converter when V_{LN} is positive. MOSFETs Q8, Q4, diode D4 and inductor L3 together form the second buck converter when V_{LN} 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 high PWM frequency (50kHz PWM) as buck converter switches in their respective line half cycle. PWM1A and PWM1B are used for controlling 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 a LCL filter at the inverter output. The double pole single through (DPST) relay RL1 is controlled by MCU and is used to connect/disconnect the inverter to/from the AC mains.

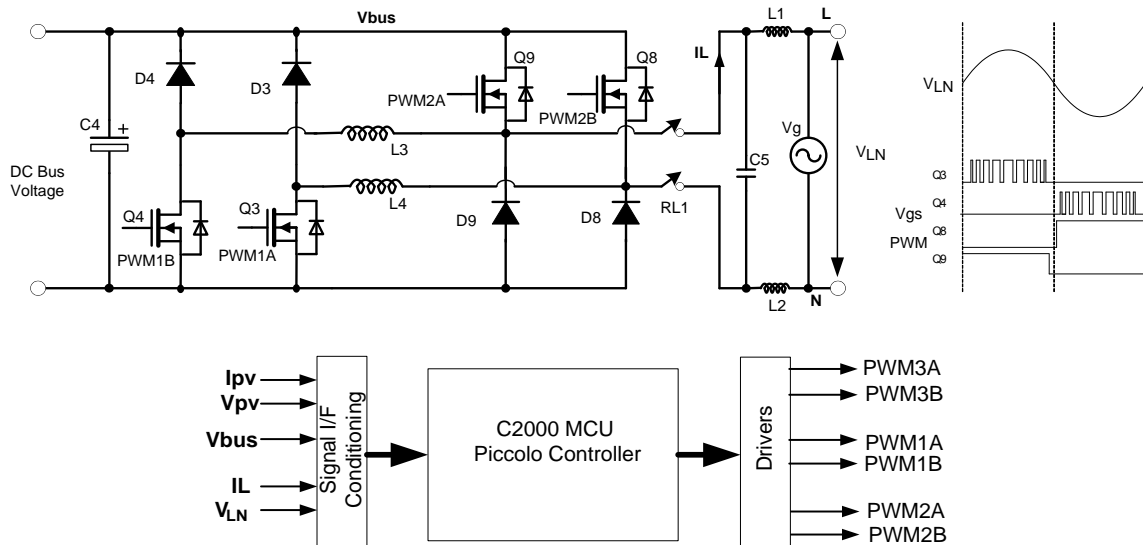


Figure 7 DC-AC inverter control using C2000 MCU

Figure 7 indicates the interface signals needed for control of the inverter using C2000 MCU. The MCU controls the hardware using three feedback signals and four PWM outputs. The signals that are sensed and fed back to the MCU include the AC line voltage (V_{LN}), the DC bus voltage V_{bus} and the main inductor current I_L . The sensed signals are used to implement the V_{bus} control loop, the current control loop and 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 V_{bus_ref} by generating the reference current command for the current loop. The current loop forces the inductor current I_L to track the reference signal I_{L_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 V_{LN} 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. Hence a feedback linearization technique in current control loop to calculate the PWM duty ratios for the inverter switches Q3 and Q4 (See 2.3.2 for details). This linearization technique assumes that the current loop gain is high at the vicinity of grid frequency of 47Hz ~ 65Hz. 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 can be seen from the plots the current loop bandwidth is about 1.2kHz and the gain at 200Hz or below is more than 20dB. A detailed description of the software algorithm is provided in the following chapters.

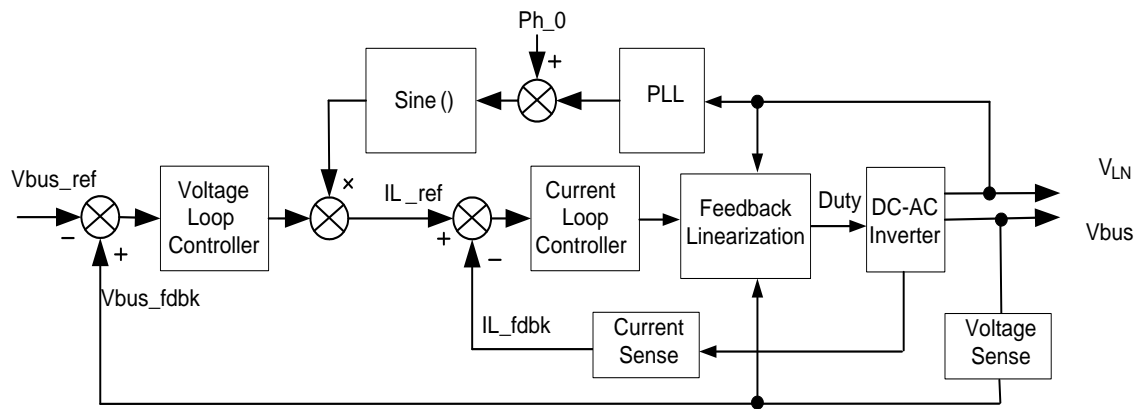


Figure 8 DC-AC inverter control loop



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 the figure below, which consists of a phase detect (PD) consisting of park transform, a loop filter (LPF) and a voltage controlled oscillator(VCO), Figure 10.

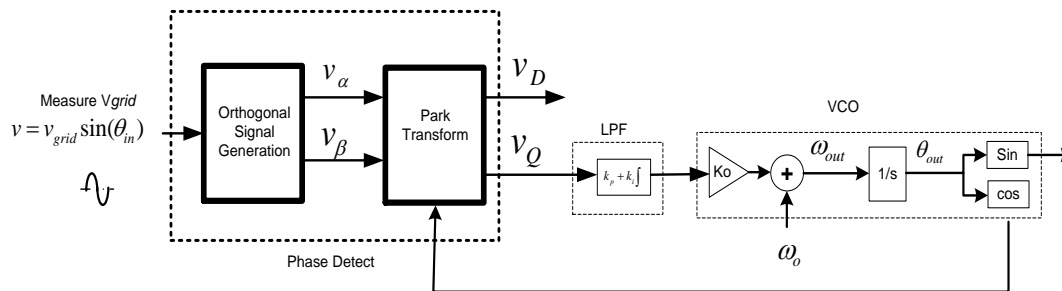


Figure 10 Inverter PLL control

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 has an advantage as it can be used to selectively tune the orthogonal signal generator to reject all frequencies except the grid frequency. Assuming an arbitrary input signal and PLL theta the Phase Detect (PD) output is given by the following equation:

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

Now assuming PLL is closed to being locked i.e. $\theta_{in} - \theta_{out} \approx 0 \Rightarrow v_q = \sin(\theta_{in} - \theta_{out}) \approx \theta_{in} - \theta_{out}$, hence v_q is the error in the PLL angle lock.

The Loop Filter is implemented using a PI controller to keep this error as zero,

$$\frac{y_{lf}(s)}{PhaseDetect(s)} = k_p + \frac{k_p}{T_i} * \frac{1}{s} = K_p + \frac{K_i}{s}$$

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

$$H_o(s) = \frac{\theta_{out}(s)}{\theta_{in}(s)} = \frac{LPF(s) * \frac{1}{s}}{1 + LPF(s) * \frac{1}{s}} = \frac{(k_p s + \frac{k_p}{T_i})}{s^2 + k_p s + \frac{k_p}{T_i}}$$

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

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

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

$$\omega_n = \sqrt{\frac{k_p}{T_i}}$$

$$\zeta = \sqrt{\frac{T_i k_p}{4}}$$

Ignoring the LHP zero from the closed loop transfer equation, step response to a general second order equation i.e.

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

is given as:

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

Where the settling time is given as the time it takes for the response to settle between an error band, let's say this error is δ , then

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

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

Now using settling time to be 30ms and the error band to be 5% and damping ratio to be 0.7 we can obtain the natural frequency to be 119.014 and then back substituting we get $K_p = 166.6$ and $K_i = 27755.55$

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

$$y_{lf}[n] = y_{lf}[n-1] * A1 + PhaseDetect_q[n] * B0 + PhaseDetect_q[n-1] * B1$$

Using Z transform this equation can be re-written as

$$\frac{y_{lf}(z)}{PhaseDetect(z)} = \frac{B0 + B1 * z^{-1}}{1 - z^{-1}} \text{-----(2)}$$

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:

$$\frac{y_{lf}(z)}{PhaseDetect(z)} = \frac{\left(\frac{2 * K_p + K_i * T}{2} \right) - \left(\frac{2 * K_p - K_i * T}{2} \right) z^{-1}}{1 - z^{-1}} \text{-----(3)}$$

Equation 2 and 3 can be compared to map the proportional and integral gain of the PI controller from the analog domain into the digital domain.

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

For example on the micro inverter kit the ISR rate is 50Khz, and hence 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. Since we are controlling the inverter current I_L , the small signal gain for inverter current to duty ratio is,

$$\frac{\tilde{i}_L}{\tilde{d}} = \frac{V_{bus}(Z_c + Z_g)}{Z_g * Z_c + (Z_c + Z_g) * Z_i}$$

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

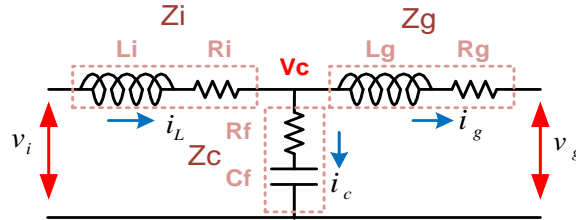


Figure 11 Inverter LCL filter

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 C_f is negligible compared to the inductor current i_L .

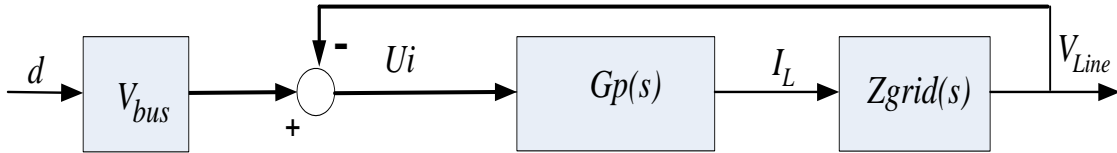


Figure 12 Inverter current loop power stage control

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 , i.e., 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.

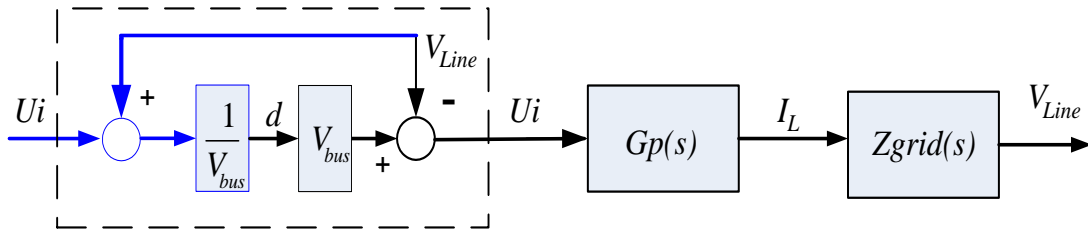


Figure 13 Simplified current loop power stage control

The simplified diagram in Figure 13 shows that the current loop power stage input is U_i which controls the inductor current I_L . This is valid when the input U_i 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 U_i using the measured values of V_{bus} and V_{Line} . This can be easily 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.

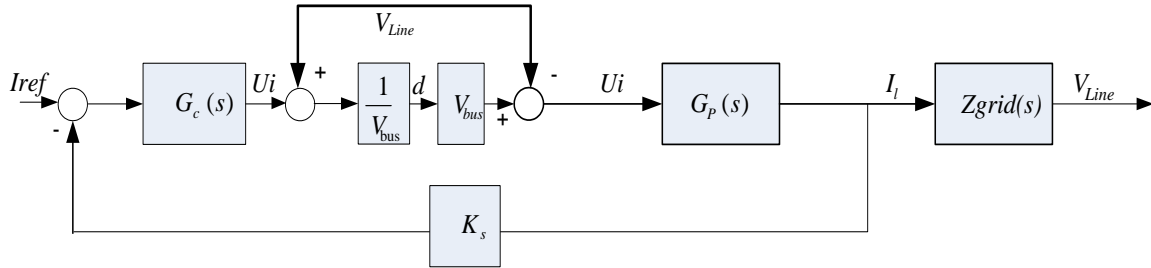


Figure 14 Inverter current control loop

Where, K_s is the current feedback gain. So the loop gain transfer function of the control loop is,

$$G_{loop}(s) = G_c(s)K_sG_P$$

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

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

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

2.4 Solar Micro Inverter Electrical Specifications

Following lists the key highlights of the micro inverter.

- Panel Voltage: 28V (Min) to 45V (Max)
- DC bus voltage: 400V (max)
- Output Power: 280W @ 220Vac, 140W @ 110Vac
- DC-DC & DC-AC combined efficiency ~92% at 50% rated load, THD~5%

DISCLAIMER: For 220V/50Hz grid operation, the EVM was tested ONLY with an emulated grid i.e. an external AC source (at least 600VA rating) with a load resistor (200.00ohm, 600W rating) at the output. Thus the 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 50KHz, ISR is needed for controlling the DC-DC Flyback and the DC-AC inverter stage. A slower ISR (1KHz) is used to run the state machine, MPPT and Power Measurement Functions.

The key framework C files used in the project are:

SolarMicroInv-Main.c – this file is used to initialize, run, and manage the application. In addition, this file also contains ISR for inverter stage control, MPPT algorithm execution, data logging and relay control etc.

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, Watchdog etc. Most of functions in this file are called once during system initialization from *SolarMicroInv-Main.c*.

SolarMicroInv-Settings.h – This file contains of setting such as incremental build option and various defines for PWM frequency, ISR triggers that are 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.

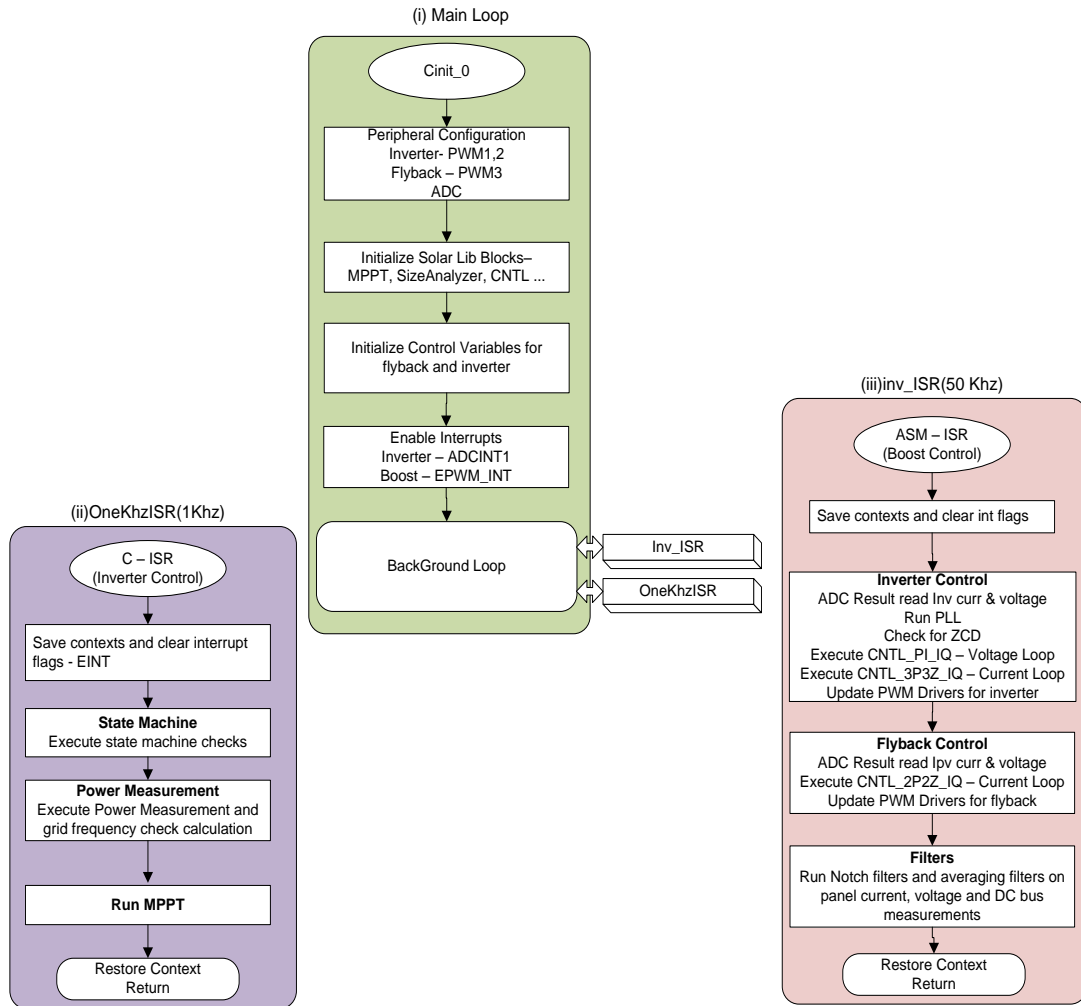


Figure 15 PV inverter software structure (i) Main loop (ii) Inverter ISR (iii) 1KHz ISR

3.2 Project Dependencies & 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

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. Following sections gives details of 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, panel needs to be operated at its maximum power point. 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 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.

Panel current reference is changed at a slow rate $\sim 10\text{Hz}$ and the power stage current regulation is run at 50KHz . The control loop for DC-DC stage does not control the boost stage output voltage. Boost output voltage however is regulated by the DC-AC inverter (described later), 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 AC power ripple seen by the panel.

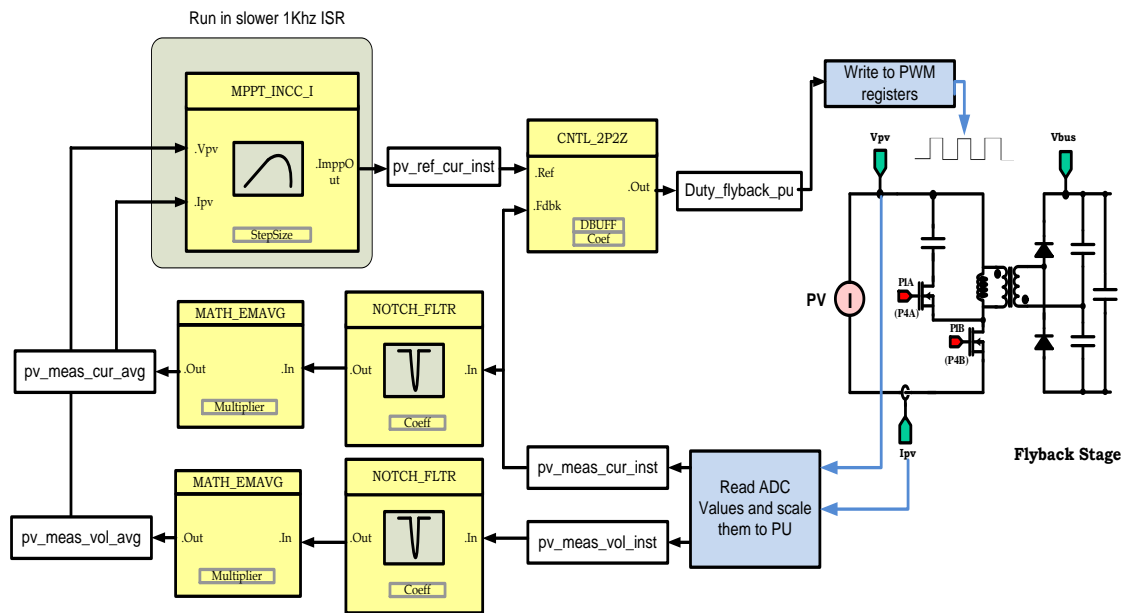


Figure 16 DC-DC 1ph Boost with MPPT software diagram

3.3.2 DC-AC Single Phase Inverter Control Software

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:

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

where D is the duty ratio of the high frequency PWM switches in the inverter power stage.

It is clear from the equation that for the inverter to be able to feed current into the grid V_{dc} must always be greater than the max grid voltage. Also it is known that 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. Voltage loop generates the reference current command for the current loop. In this case increasing 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, i.e. 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. In case of 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 duty ratio for the inverter switches. A notch filter is used to remove the 2nd 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, 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.

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

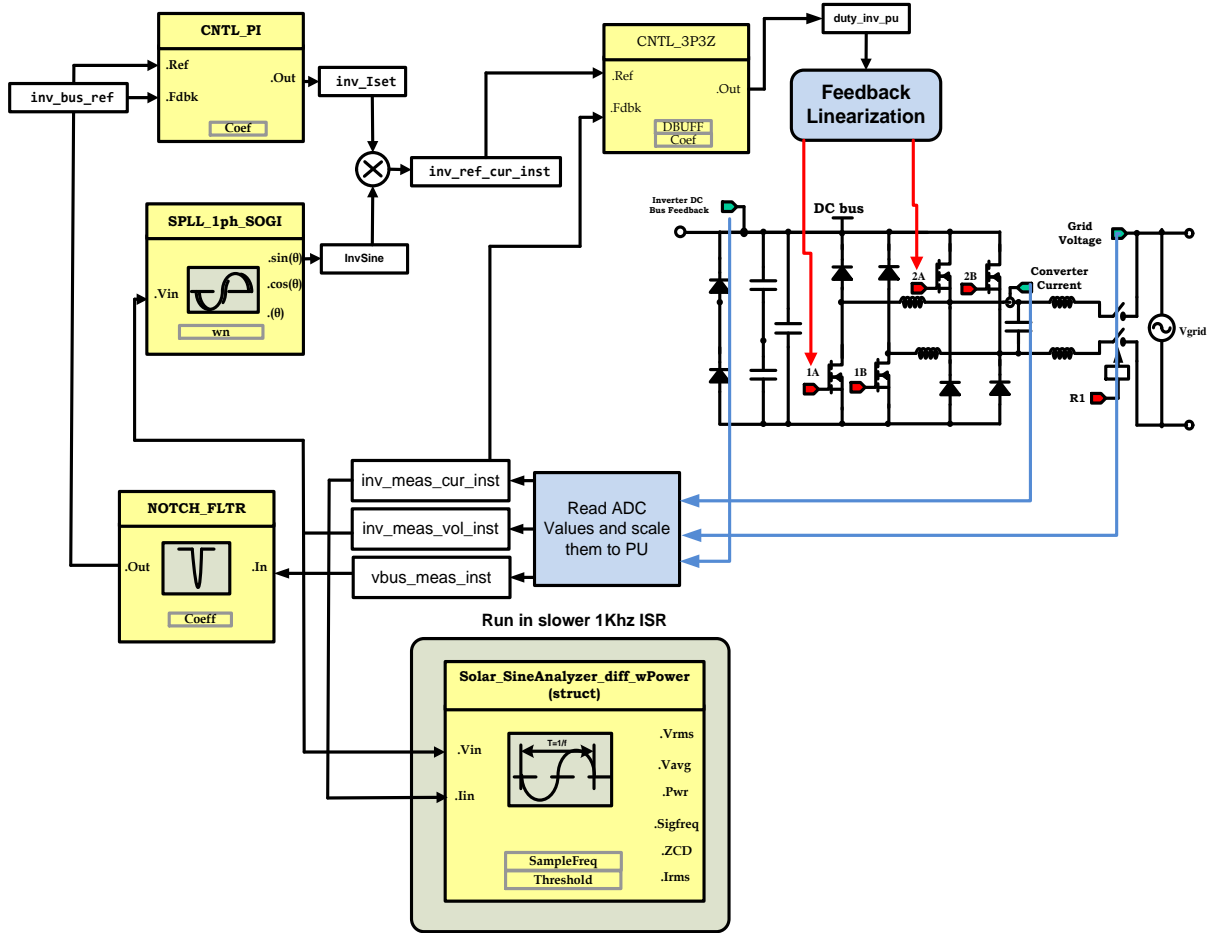


Figure 17 Closed loop current control of inverter with grid connection

3.4 DC-DC and DC-AC Integration

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

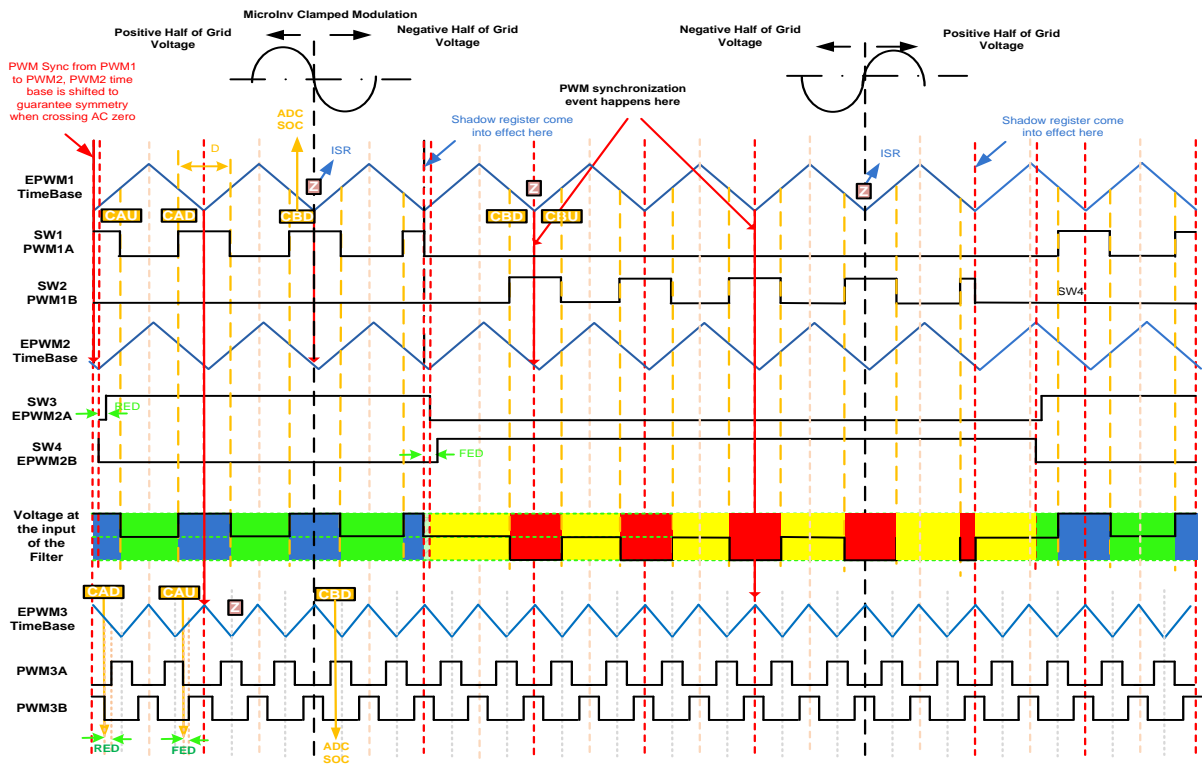


Figure 18 PWM switching scheme for DC-DC and DC-AC stage integration

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

3. Open “Code Composer Studio”.
4. Once Code Composer Studio opens, the workspace launcher may appear that would ask to select a workspace location, Figure 19: (please note workspace is a location on the hard drive where all the user settings for the IDE i.e. which projects are open, what configuration is selected etc. are saved, this can be anywhere on the disk, the location mentioned below is just for reference. Also note that if this is not your first-time running Code Composer this dialog may not appear)
 - Click the “Browse...” button
 - Create the path below by making new folders as necessary.
 - “C:\Documents and Settings\My Documents\ CCS_workspaces\ProjectWorkspace”
 - Uncheck the box that says “Use this as the default and do not ask again”.
 - Click “OK”.

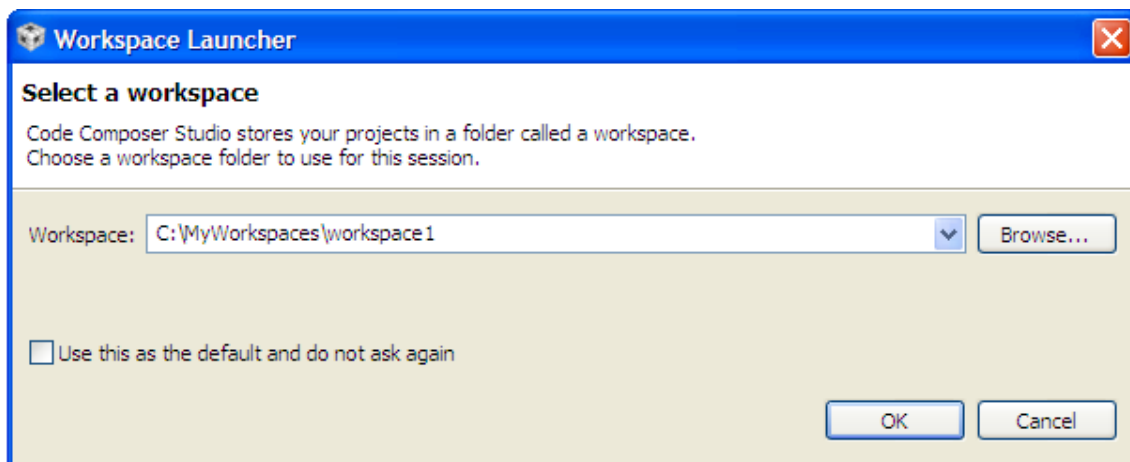


Figure 19 Opening a new workspace

5. Next we will configure Code Composer to know which MCU it will be connecting to. If you have previously connected to F28035 device using XDS100v1 emulator you may skip this step. Click “Target -> New Target Configuration...”. In the popup window, Figure 20, name the new configuration xds100-f28035.ccxml. Make sure that the “Use shared location” checkbox is checked and then click Finish.

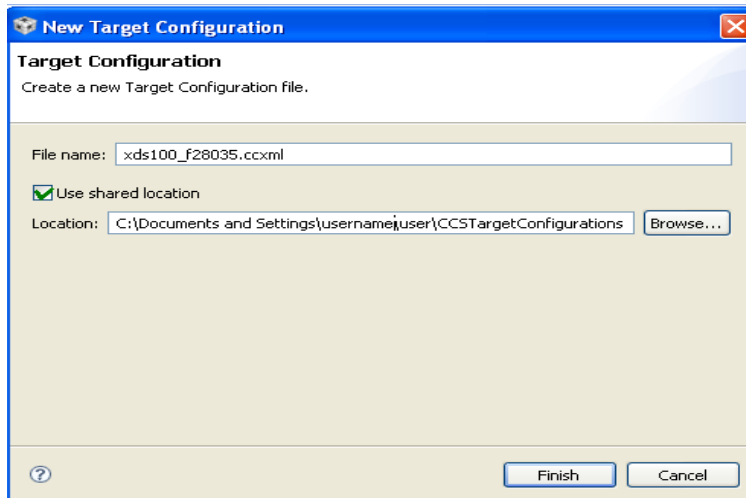


Figure 20 New Target Configuration for XDS100 v1 and F28035 MCU

6. A new tab will now appear, Figure 21. Select and enter the options as shown:
- Connection – Texas Instruments XDS100v1 USB Emulator
 - Device – TMS320F28035
 - Click Save
 - Close the xds100-f28035.ccxml tab

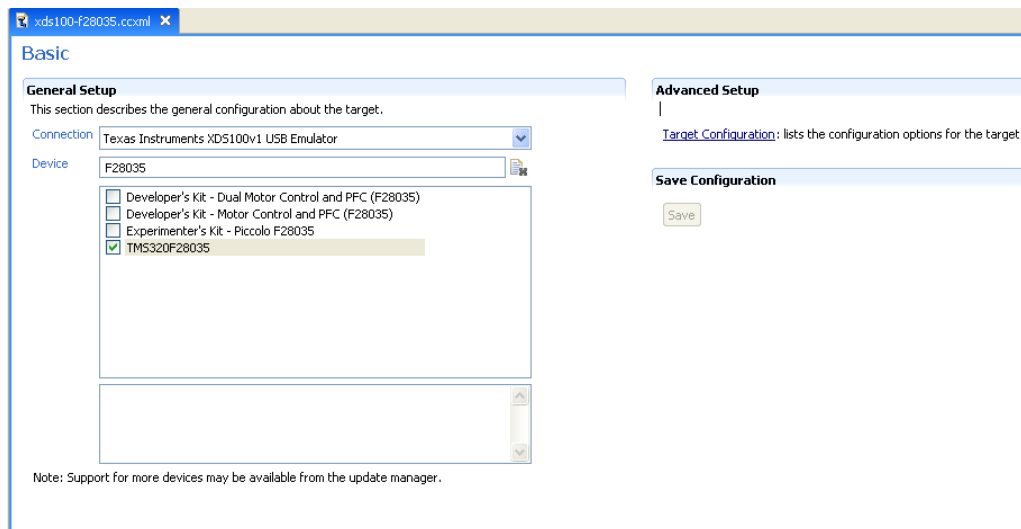


Figure 21 Target configuration

7. Assuming this is your first time using Code Composer, the xds100-F28035 configuration is now set as the default target configuration for Code Composer. Please 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 you to reuse existing target configurations and link them to specific projects.
8. Add all the solar micro inverter project into your current workspace by clicking “Project->Import Existing CCS/CCE Eclipse Project”, Figure 22.
 - Select the root directory of the SolarMicroInverter. This will be: `\controlSUITE\development_kits\TMDSSOLARUINVKIT_vX\MicroInv_F2803x`

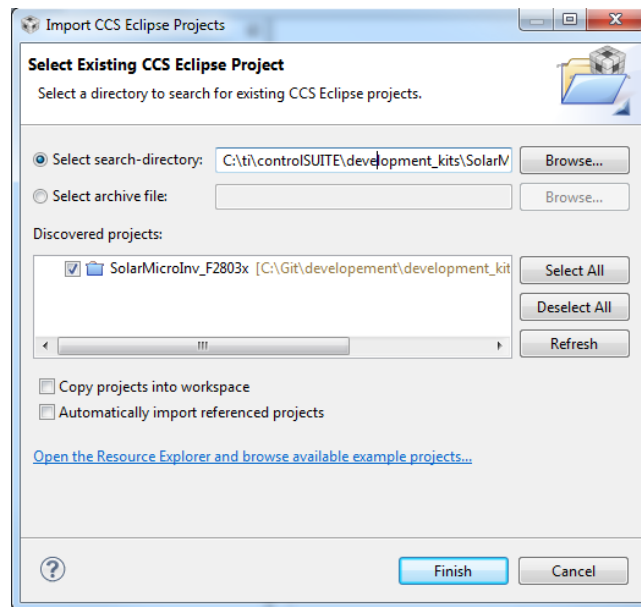


Figure 22 Adding F28035 PV inverter project to the workspace

- Click Finish, this would copy all the projects relevant for the kit into the workspace. If you want only a particular project to be copied uncheck the box next to the other project names.

Configuring a Project

9. Expand the file structure of the project you would like to run from the C/C++ Projects tab. Right-click on this project’s name and select “Set as Active Project”, if this is not already the case.
10. The project includes the target configuration for the XDS100v1 with F28035, make sure this is set as active and default. If not go to “View-> Target Configurations”, you will see the XDS100v1 F28035 target configuration that was configured earlier. Right clicking on the Target configuration name link it to the imported Project.
11. Figure 23 shows the project in the CCS C/C++ Project tab, it shows all the key files used in the project.

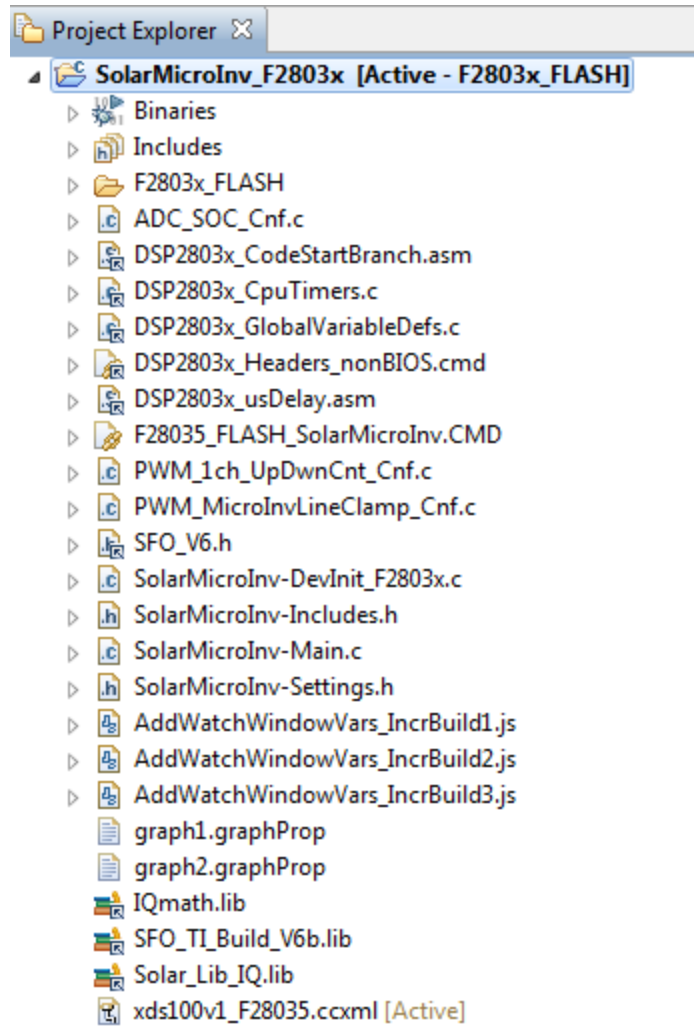


Figure 23 PV inverter project in C/C++ tab

5. Hardware Setup

5.1 Equipment Required

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

1. Input power source, this 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 25-44VDC, 20A and 400W 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 25V~44V Output, rated for ~140W (max) if connecting to grid at 110Vrms or ~240W (max) if connecting to 220Vrms.
2. 200 Ohms, 500W power resistor
3. 100 Ohms, 1KW power resistor
4. 1K Ohms 500W power resistor

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

5. Volt-Meters of up to 500V input range
6. Oscilloscope of ~200MHz bandwidth.
7. High Volt Differential Probe, Tektronix P5205 or similar.
8. Current Probe, Tektronix TCP202 or similar
9. Current Meter for output AC current, Up to 5A (RMS) range
10. Current Meter for input DC current, Up to 12A (DC) range

5.2 Jumper & 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 the section 5.3)

1. **Inspect/Install jumpers:** Before applying any power, ensure that these jumpers are configured properly. Table 1 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 1 List of jumper settings for C2000 Solar Micro inverter EVM

Jumper No.	Jumper Settings	Comments
J15	Install jumper	Jumper for +12V external bias.
J38, J39	Do not install jumpers	

J40	Install jumper	Jumper for +15V external bias.
J41, J42	Do not install jumpers	
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 outputs from C2000
J30, J31	Install both jumpers	Jumpers for Fly-back stage PWM outputs from C2000

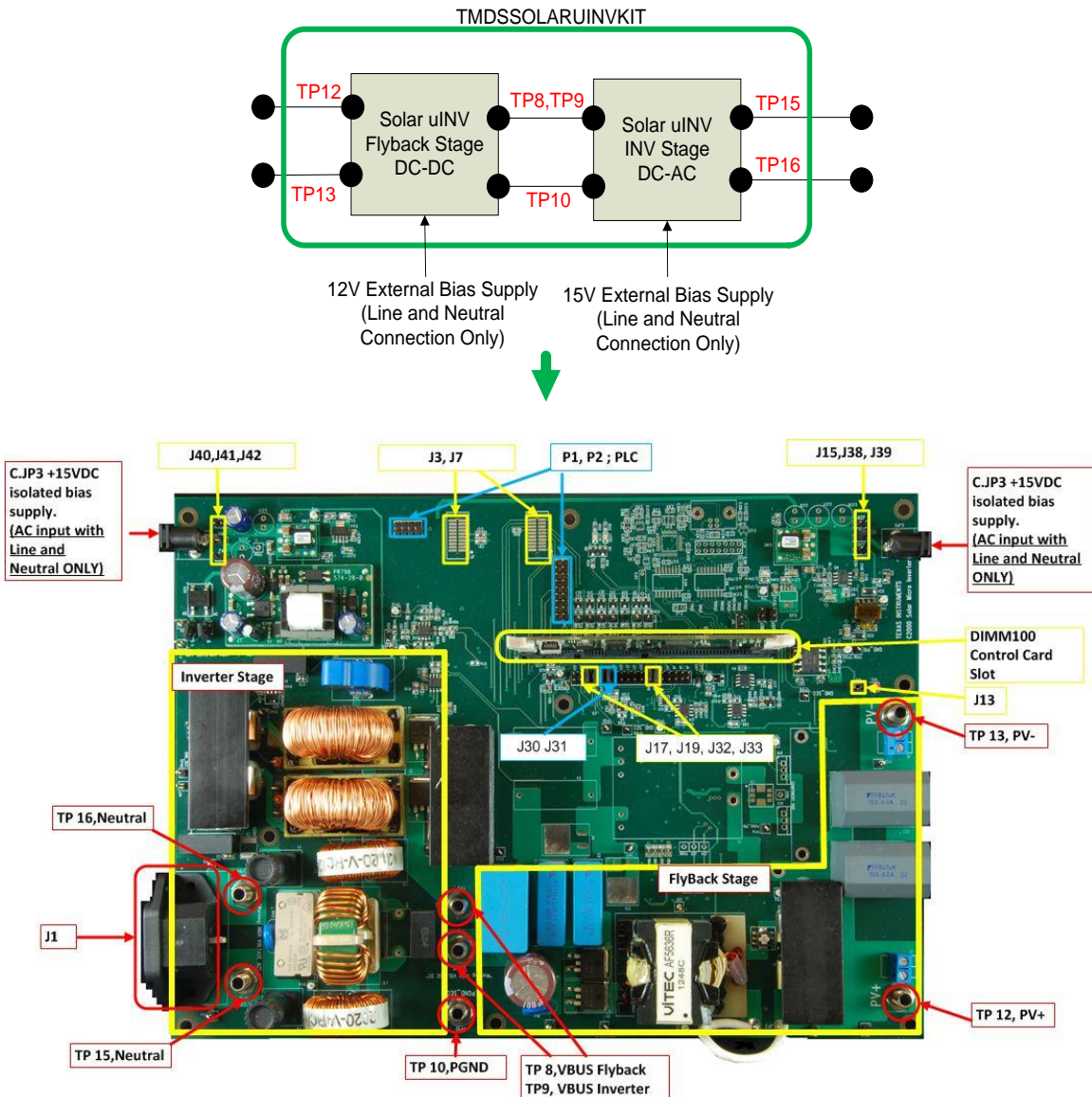


Figure 24 Jumper & Connector locations on C2000 Solar Micro inverter

2. Install / Verify F28035 ISO Control Card is connected to the EVM header U6.

3. Check the **switch SW3 is set to ON position** on the control card, to enable JTAG connection.
4. **Connect a USB cable** from the ISO Control Card to the PC on which GUI needs to run. Verify that the LED LD4 on the control card is ON indicating 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 +12VDC isolated bias supply to C:JP2 connector and power it on using 110Vac or 220Vac input (Figure 2, lower left). **Make sure the supply uses Neutral and Line connection only** (no earth connection).
7. Connect the external +15VDC isolated bias supply to C:JP3 connector and power it on using 110Vac or 220Vac input(Figure 2, lower right). **Make sure 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 option to generate bias supply from the AC source on the secondary side and from the panel on the primary side. Following table lists the jumper connections needed to use the internal bias power supply.

Table 2 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. This makes it easier to learn and get familiar with the board and the software. This approach is also good for debugging/testing 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 rebuild-all compiler option. Next chapters provide more details on how to run each of the build options.

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 together).

For the Inverter Stage GRID_CONNECT #define is used to test incrementally the closed loop performance of the loop without grid connection first and second with grid connection:

GRID_CONNECT 0: Forced ramp angle is used for the inverter control, tests 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, SPLL is used to calculate the grid angle. Inverter current loop control with grid connection is tested in this build. (Note a resistive load can be connected at the output to see grid current inversion) {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 is used to specify if the MPPT is implemented or not.

MPPT 0: Fixed input current is commanded from the flyback stage. User must ensure proper resistive load is connected at the output.

MPPT 1: Flyback stage current command is used to maintain the input panel at MPPT, User must ensure proper resistive load is connected at the output and a panel or a panel emulator is connected at input.

Build 3: Tests the action of 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 the stages together with a resistive load at the output of the inverter. This can be used to test 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 the stages and the state machine when resistive load is at the output and input if from a PV / PV emulator source.

GRID_CONNECT 1, MPPT 0: This option is used to test both the stages together with grid connection at the output, a DC source at the input instead of a PV panel emulator is used hence MPPT is disabled.

GRID_CONNECT 1, MPPT 1: This option tests the combined action of both the stages and the state machine when grid is connected at the inverter output. DC input is from a PV / PV emulator source hence MPPT is enabled. This is the **full PV inverter system build**.

All software files related to this C28x controlled Solar Explorer system i.e., 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

Objective:

The objectives of this build are, (1) evaluate PWM and ADC software driver modules, (2) verify MOSFET gate driver, voltage and current sensing circuit, (3) become familiar with the operation of Code Composer Studio (CCS) (4) Test the SPLN module. Under this build the system runs in open loop for the Inverter power stage and DCDC boost stage. Steps required for building and running a project are explained next.

Overview:

The software in Build1 has been configured so that the user can 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 expressions window in CCS. 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.

6.1.1 Flyback Open Loop Check

Procedure

Assuming the hardware and software setups are followed as described in Section 4 & 5.2, further 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 need to be connected are highlighted in Figure 25.

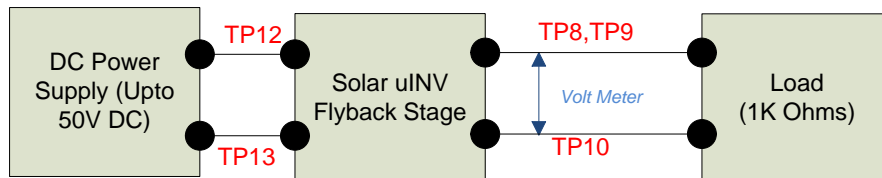


Figure 25 Power stage connections for Flyback Open Loop Check

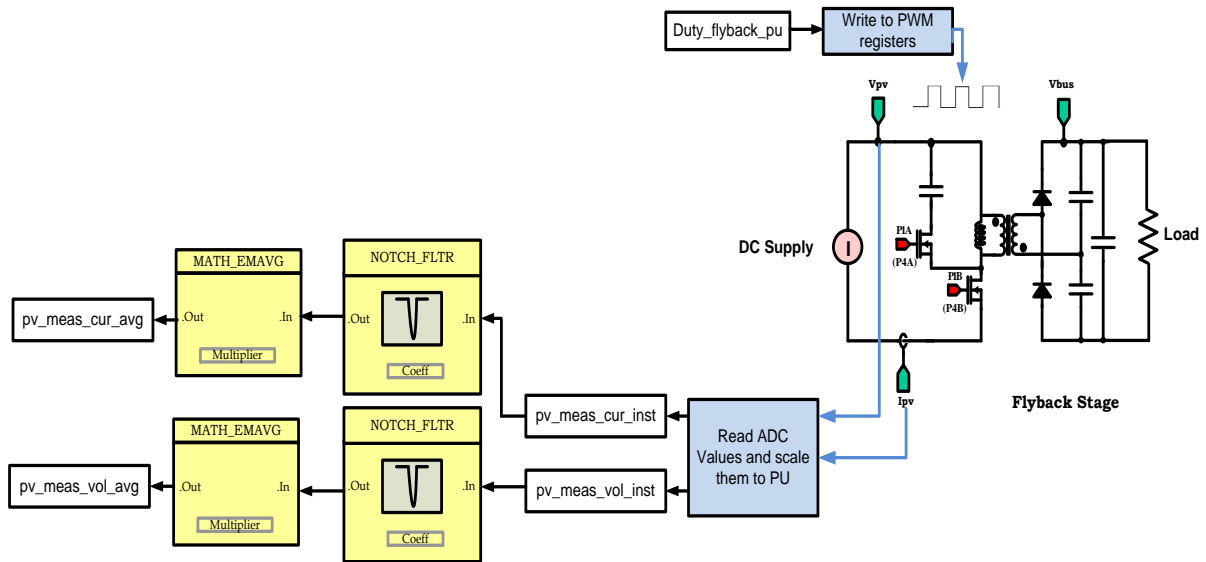


Figure 26 Build 1 Flyback Open Loop Check Software Diagram

1. In CCS click the plus sign (+) sign on the project name in the project window, all the files used in the micro inverter project will be displayed. 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 have been configured for each peripheral.
2. Next open and inspect *SolarMicroInv-Main.c*. Locate the main() function in this file.
3. First is the main() is the device and peripheral initialization, notice the call made to *DeviceInit()* function and other peripheral initialization functions. Inspect the PWM initialization and ADC initialization for the Flyback and the inverter.
4. Following which in the main() is the initialization of the solar lib modules and the control variables.
5. An offset calibration routine is run to calculate analog offset on the ADC input channels for PV current and the grid.
6. In the end the interrupt is configured for a 50KHz ISR which is used for the control loop and 1KHz ISR for background task. The 1KHz ISR is chosen such that 50KHz ISR can interrupt the 1KHz ISR.


Build and Load the Project

7. Now select the incremental build option as 1 in the *SolarMicroInv-Settings.h* file.

Note: Whenever you change the incremental build option always do a “Rebuild All”.

8. Click Project → “Rebuild All” button and watch the tools run in the build window. The project will compile successfully.
9. Then click Target → ”Debug Active Project”. The program will then be loaded into the flash of the F28035 device. You should now be at the start of Main().

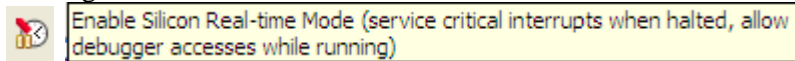
Debug Environment Windows


- 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 will populate the watch window with 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 continuous update of values from the controller, Figure 27.

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.

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




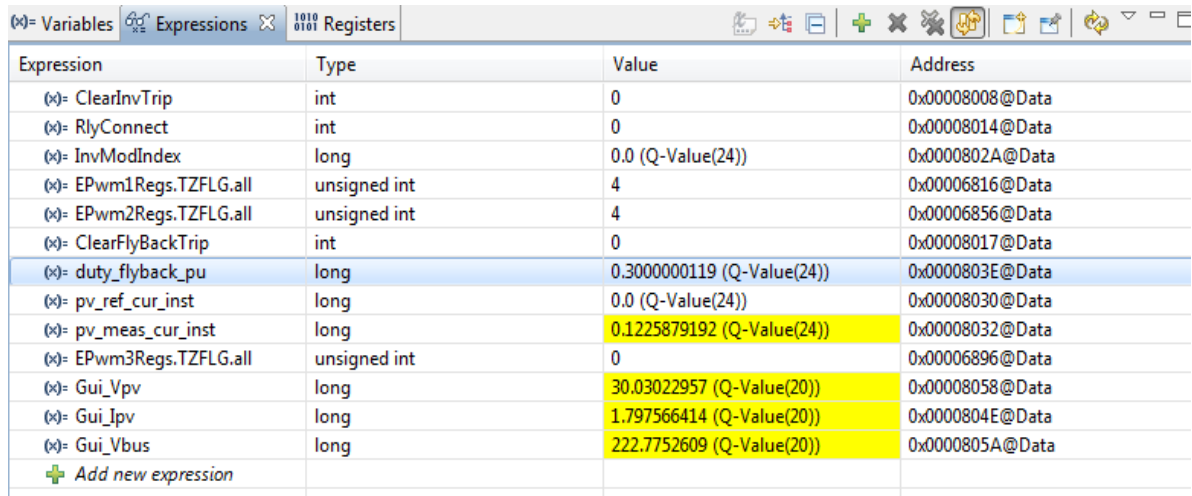
- A message box *may* appear. If so, select YES to enable debug events. This will set bit 1 (DGBM bit) of status register 1 (ST1) to a “0”. The DGBM is the debug enable mask bit. When the DGBM bit is set to “0”, memory and register values can be passed to the host processor for updating the debugger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

Expression	Type	Value	Address
(x)- ClearInvTrip	int	0	0x00008008@Data
(x)- RlyConnect	int	0	0x00008014@Data
(x)- InvModIndex	long	0.0 (Q-Value(24))	0x0000802A@Data
(x)- EPwm1Regs.TZFLG.all	unsigned int	4	0x00006816@Data
(x)- EPwm2Regs.TZFLG.all	unsigned int	4	0x00006856@Data
(x)- ClearFlyBackTrip	int	0	0x00008017@Data
(x)- duty_flyback_pu	long	0.0 (Q-Value(24))	0x0000803E@Data
(x)- pv_ref_cur_inst	long	0.0 (Q-Value(24))	0x00008030@Data
(x)- pv_meas_cur_inst	long	0.001005887985 (Q-Value(24))	0x00008032@Data
(x)- EPwm3Regs.TZFLG.all	unsigned int	4	0x00006896@Data
(x)- Gui_Vpv	long	0.0 (Q-Value(20))	0x00008058@Data
(x)- Gui_Ipv	long	0.003962516785 (Q-Value(20))	0x0000804E@Data
(x)- Gui_Vbus	long	0.01195907593 (Q-Value(20))	0x0000805A@Data
+ Add new expression			

Figure 27 Watch window populated using script

Run the Code

14. Run the project, 
15. In the watch view, check the value of Gui_Vpv this will be the voltage at the input of the board. Now slowly raise this voltage to 30V of the input power supply and see Gui_Vpv change as you increase the input voltage, Figure 28.
16. Now clear the flyback stage trip by writing a 1 to “ClearFlyBacktrip” variable, you can see the EPwm3Regs.TZFLG.all will go to zero as soon as the trip is cleared.
17. Now enter some duty in the “duty_flyback_pu” field. Starting from 0.1 go up to 0.3. Now the watch window will look as following:




Expression	Type	Value	Address
(x)- ClearInvTrip	int	0	0x00008008@Data
(x)- RlyConnect	int	0	0x00008014@Data
(x)- InvModIndex	long	0.0 (Q-Value(24))	0x0000802A@Data
(x)- EPwm1Regs.TZFLG.all	unsigned int	4	0x00006816@Data
(x)- EPwm2Regs.TZFLG.all	unsigned int	4	0x00006856@Data
(x)- ClearFlyBackTrip	int	0	0x00008017@Data
(x)- duty_flyback_pu	long	0.300000119 (Q-Value(24))	0x0000803E@Data
(x)- pv_ref_cur_inst	long	0.0 (Q-Value(24))	0x00008030@Data
(x)- pv_meas_cur_inst	long	0.1225879192 (Q-Value(24))	0x00008032@Data
(x)- EPwm3Regs.TZFLG.all	unsigned int	0	0x00006896@Data
(x)- Gui_Vpv	long	30.03022957 (Q-Value(20))	0x00008058@Data
(x)- Gui_Ipv	long	1.797566414 (Q-Value(20))	0x0000804E@Data
(x)- Gui_Vbus	long	222.7752609 (Q-Value(20))	0x0000805A@Data
 Add new expression			

Figure 28 Watch window during running of BUILD 1 Flyback Stage

18. Check the multi meter 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.
19. This completes the DC Flyback open loop check. If this does not work you can check the PWM and gate drive signals using a differential probe referring to the kit schematics.
20. To end this test, enter duty_flyback_pu to zero and reduce the DC power supply voltage to 0V. One can now 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.

6.1.2 Inverter Open Loop Check

Next the inverter stage is checked 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 need to 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 is later in the document. The software structure used for the open loop test of the inverter is shown in Figure 30.

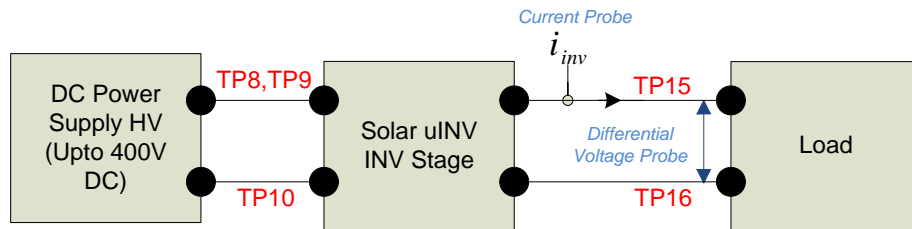


Figure 29 Power stage connections for Inverter Open Loop Check

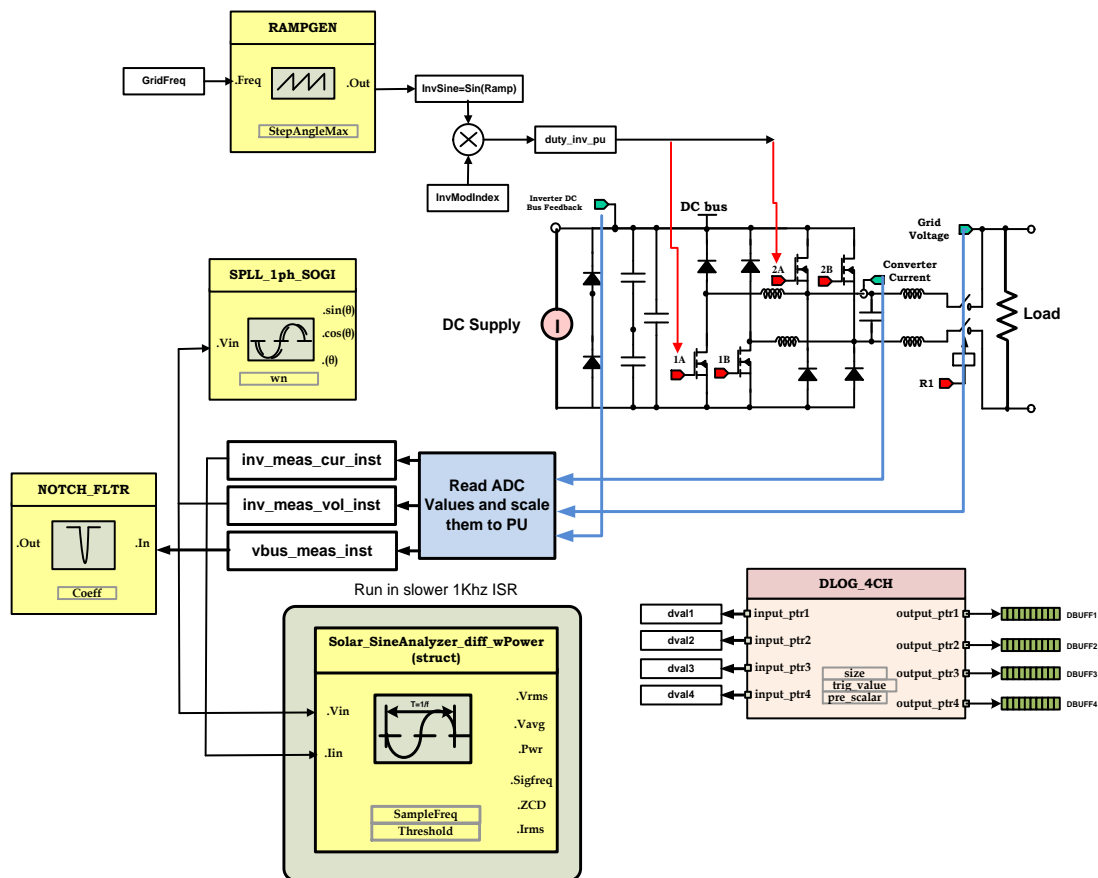


Figure 30 Build 1 Inverter Open Loop Check Software Diagram

Make sure no high voltages are present on the board, input of the flyback stage is zero and no voltages are present at the output. Now 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 connection for a differential probe and current probe as shown in Figure 29.

Run the Code

21. 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/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.
22. 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.
23. Connect the inverter to the load by firing the Relay ON by writing '1' to RelayConnect variable.
24. Now write a 0.5 to InvModIndex.
25. Now turn on the HV DC supply and slowly increase the DC bus voltage. One can observe the DC Bus voltage reading on the GUI as well to verify the connections are right.
26. Go up to 300V at the DC input side and observe the Gui_Vrms, Gui_Irms values in the watch expressions. Also observe the voltage and the currents on the scope, they should match Figure 32 & Figure 32 respectively.

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

Figure 31 Watch Expression Build Level 1 Inverter Open Loop Check

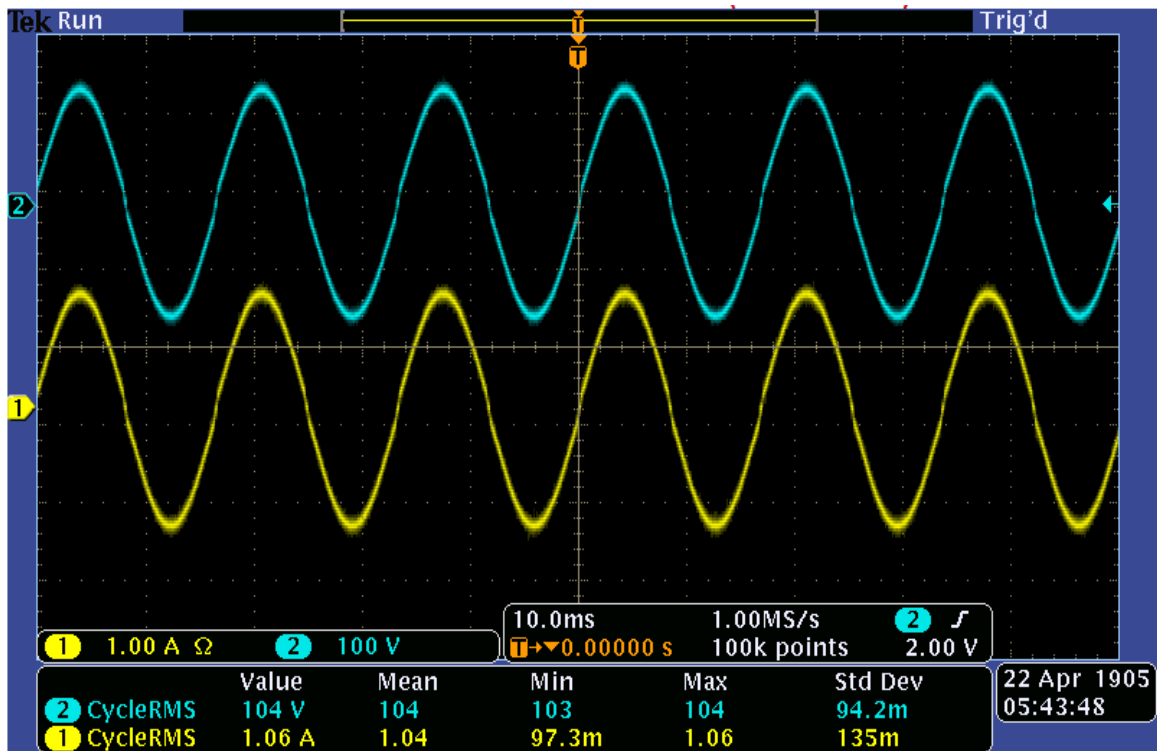


Figure 32 Build Level 1 Inverter Voltage and Current Check

27. 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 will populate the graph properties window as Figure 33. Alternatively the user can enter the values as shown in the Figure 33 manually. Once the entries are verified, click ok. Two graphs will appear in CCS now. Click on continuous refresh on these graphs.

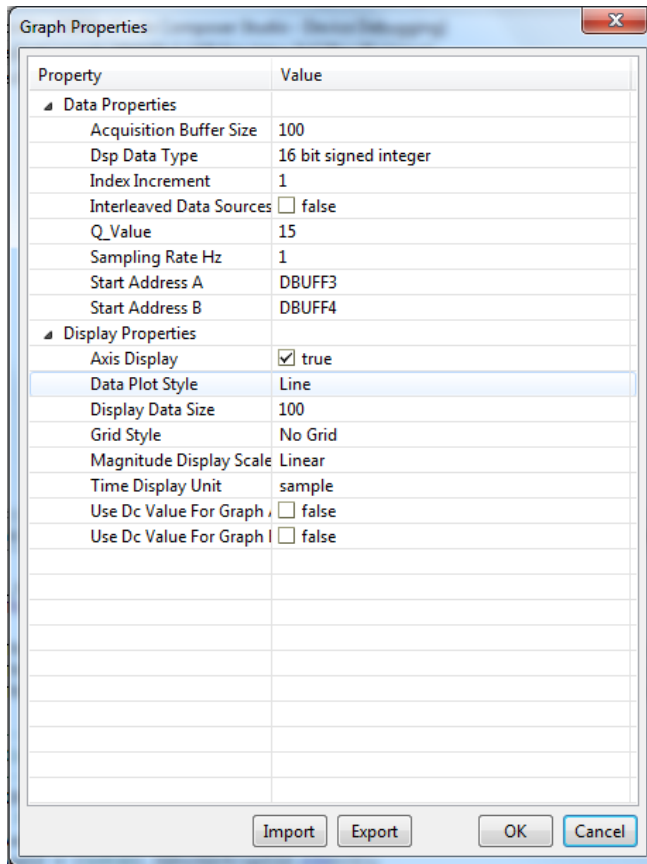


Figure 33 Graph Window Properties

28. The graphs will appear as shown below.

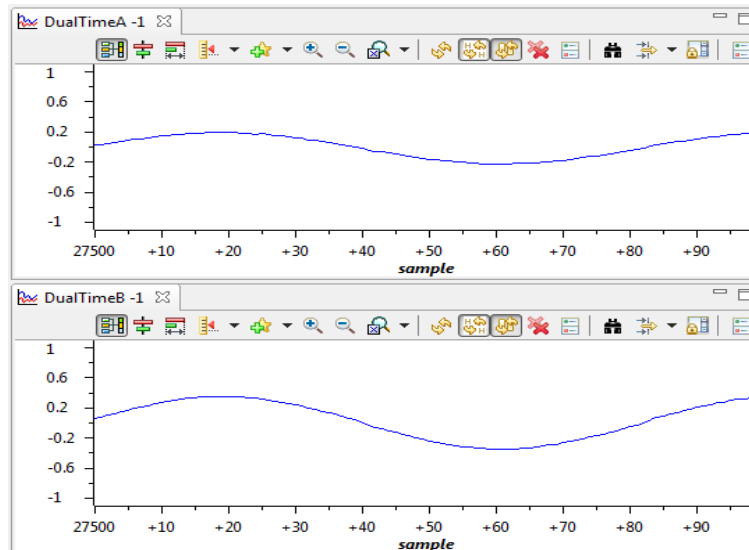






Figure 34 Build 1 Inverter Open Loop CCS Graphs

29. Change the InvModIndex variable and see the voltage and current measurements change in the watch window and the graphs.

30. PLL performance can also be checked by importing the graph1.GraphProp file which plots the invsine which is the forced angle and the PLL output sine value.
31. This completes the inverter open loop check. To end the debug session enter 0 in the InvModIndex and then slowly reduce the DC supply voltage input to the inverter to zero. Put RelayConnect to 0.
32. Fully halting the MCU when in real-time mode is a two-step process. Now, 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 .
33. Close CCS debug session by clicking on Terminate Debug Session  (Target->Terminate all).

End of Exercise

6.2 BUILD = 2

Objective:

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

Overview:

The software in Build2 has been configured so that the user can quickly evaluate the closed current loop performance of the flyback and the inverter stage. Steps required for building and running a project are explained next.

6.2.1 Flyback Closed Current Loop **without** MPPT

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.

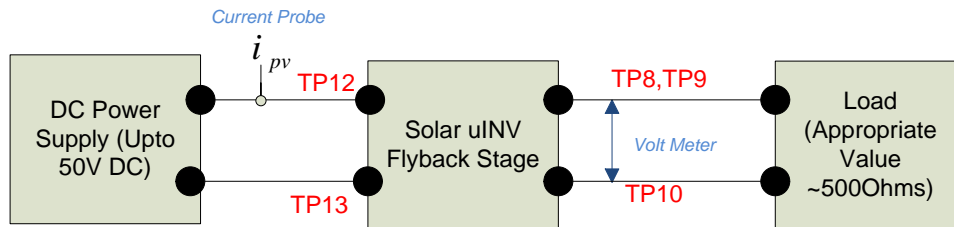


Figure 35 Build Level 2 Flyback Close Current Loop Operation

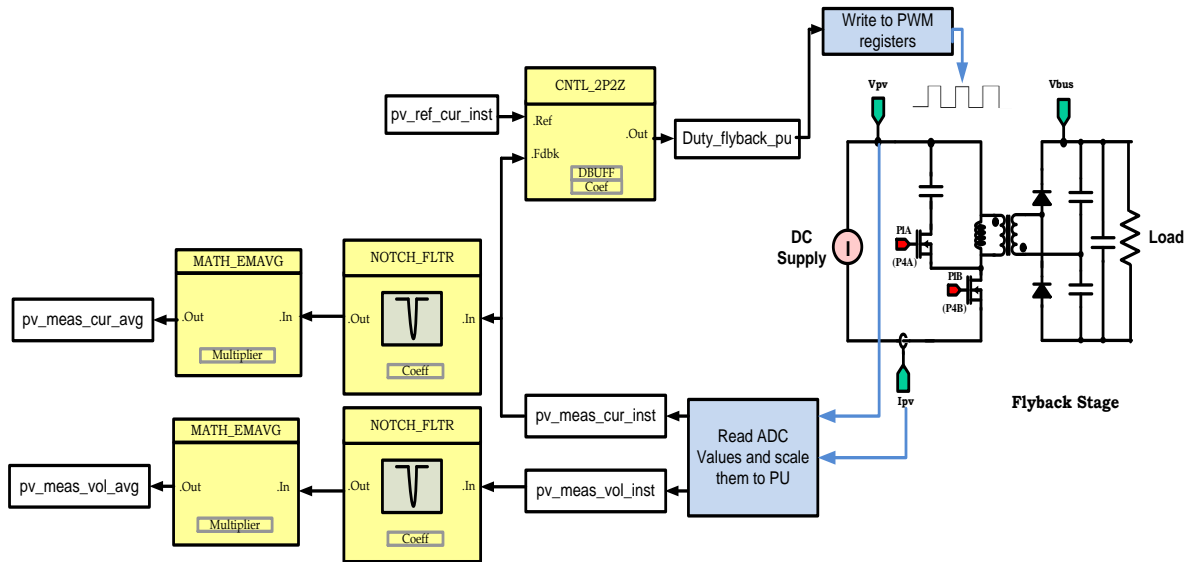


Figure 36 Build 2 Closed Loop Flyback Test without MPPT


1. Follow the steps from the previous sections to import the micro inverter project into CCS.
2. Now change the build level open the *SolarMicroInv-Settings.h* file. Make sure the defines are as below, if not modify to match below:

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

Note: Whenever you change the incremental build option always do a “Rebuild All”.

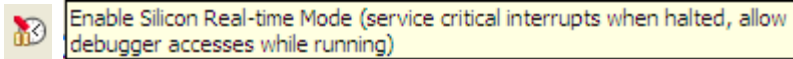
3. Click Project → “Rebuild All” button and watch the tools run in the build window.
4. Click Target → ”Debug Active Project”. The program will be loaded into the flash. You should now be at the start of Main().

Debug Environment Windows

5. Click 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 will populate the watch window with appropriate variables needed to debug the system and the appropriate Q formats. Click on Continuous Refresh  button on the watch window to enable continuous update of values from the controller.

Using Real-time Emulation

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



7. A message box *may* appear. If so, select YES to enable debug events. This will set bit 1 (DGBM bit) of status register 1 (ST1) to a “0”. The DGBM is the debug enable mask bit. When the DGBM bit is set to “0”, memory and register values can be passed to the host processor for updating the debugger windows.
8. Click on Continuous Refresh buttons for the watch view.

Run the Code


9. Run the project,
10. In the watch view, check the value of Gui_Vpv this will be the voltage at the input of the board. Now slowly raise this voltage to 30V and see the Gui_Vpv change as you increase the input voltage.
11. Now clear the flyback stage trip by writing a 1 to “ClearFlyBacktrip”, you can see the EPwm3Regs.TZFLG.all will go to zero as soon as the trip is cleared.
12. Now enter some input current value in pu format in the “pv_ref_cur_inst”. For example for 1.5Amps you may enter _IQ24(0.1), the max current sense is 15Amps 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, Figure 37.

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

Expression	Type	Value	Address
(*)= ClearInvTrip	int	0	0x00008015@Data
(*)= RlyConnect	int	0	0x00008010@Data
(*)= inv_Iset	long	0.0 (Q-Value(24))	0x00008040@Data
(*)= CloselooptInv	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

13. This completes the Flyback Closed Current Loop check without MPPT, now enter ‘0’ for the pv_ref_cur_inst in the watch window and reduce the DC input voltage to zero.
14. Fully halting the MCU when in real-time mode is a two-step process. Now, 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 .

15. Close CCS debug session by clicking on Terminate Debug Session  (Target->Terminate all).

6.2.2 Flyback Closed Current Loop **with** MPPT

Procedure

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

- a. Open Circuit Voltage 40V
- b. Maximum Power Point Voltage 30V
- c. Short Circuit Current 5.2Amps
- d. Maximum Power Point Current 4Amps

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

CAUTION: Panel is an energized source and extreme caution must be taken to connect this to the board, Refer to your organizations safety procedure before connecting the panel.

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

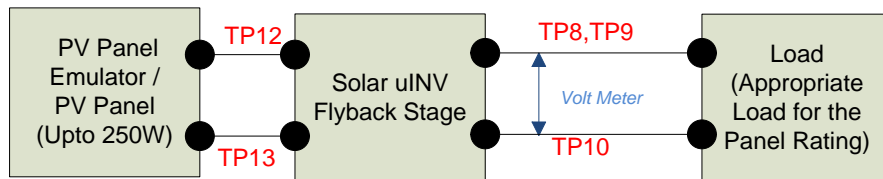


Figure 38 Build 2 Closed Loop Flyback Check with MPPT

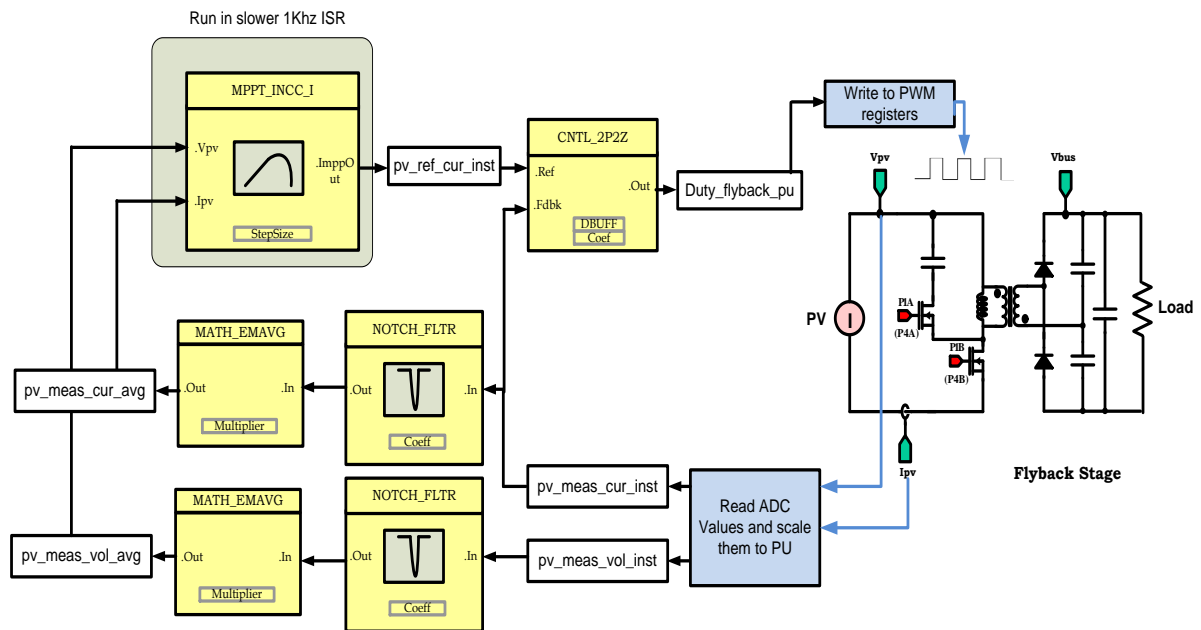


Figure 39 Build 2 Software Diagram for Closed Current Loop Flyback Check with MPPT


1. Follow the steps from the previous sections to import the micro inverter project into CCS.
2. First to change the build level; open the *SolarMicroInv-Settings.h* file. Make sure the defines are as below if not modify to match below:

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

Note: Whenever you change the incremental build option always do a “Rebuild All”.

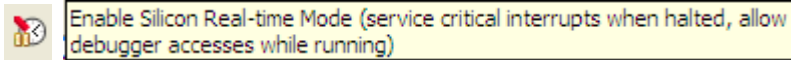
3. Click Project → “Rebuild All” button and watch the tools run in the build window.
4. Click Target → ”Debug Active Project”. The program will be loaded into the flash. You should now be at the start of Main().


Debug Environment Windows

5. Click View->Scripting Console to open the scripting console and open the “AddWatchWindowVars_IncrBuild2.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 Continuous Refresh  button on the watch window to enable continuous update of values from the controller.


Using Real-time Emulation

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



7. A message box *may* appear. If so, select YES to enable debug events. This will set bit 1 (DGBM bit) of status register 1 (ST1) to a “0”. The DGBM is the debug enable mask bit. When the DGBM bit is set to “0”, memory and register values can be passed to the host processor for updating the debugger windows.
8. Click on Continuous Refresh buttons  for the watch view.

Run the Code





9. Run the project, 
10. At this stage switch on the PV panel emulator or connect the panel to the board.
11. In the watch view, check the value of Gui_Vpv this will display the panel open circuit voltage.
12. Now clear the flyback stage trip by writing a 1 to “ClearFlyBacktrip” variable, you can see the EPwm3Regs.TZFLG.all will go to zero as soon as the trip is cleared.
13. Now make MPPT_ENABLE to ‘1’, and observe the input PV current slowly rise to the MPPT point.
14. This verified the MPPT algorithm operation.

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

Expression	Type	Value	Address
ClearInvTrip	int	0	0x00008015@Data
RlyConnect	int	0	0x00008010@Data
inv_Iset	long	0.0 (Q-Value(24))	0x00008040@Data
CloseLoopInv	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	0.241617918 (Q-Value(24))	0x00008044@Data
pv_ref_cur_inst	long	0.138372004 (Q-Value(24))	0x00008034@Data
pv_meas_cur_inst	long	0.1369922161 (Q-Value(24))	0x00008026@Data
EPwm3Regs.TZFLG.all	unsigned int	0	0x00006896@Data
Gui_Vpv	long	29.28743935 (Q-Value(20))	0x0000805A@Data
Gui_Ipv	long	2.05637455 (Q-Value(20))	0x00008058@Data
Gui_Vbus	long	192.4721584 (Q-Value(20))	0x00008054@Data
MPPT_ENABLE	int	1	0x00008018@Data

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

15. This completes the Flyback Closed Current Loop check with MPPT, now enter ‘0’ for MPPT_ENABLE and then ‘0’ for pv_ref_cur_inst in the watch window. Now disconnect the Panel emulator/panel from the board. **CAUTION: Take appropriate precaution when removing the Panel as it is an energized source**

16. Fully halting the MCU when in real-time mode is a two-step process. Now, 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 .
17. Close CCS debug session by clicking on Terminate Debug Session  (Target->Terminate all).

6.2.3 Inverter Closed Current Loop **without GRID**

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, Figure 41. Keep the HV supply off, but make the connections as show below. The software structure used is illustrated in Figure 42.

CAUTION: High Voltage proceeds with extreme caution. Please refer to your organizations safety procedure for handling high voltages.

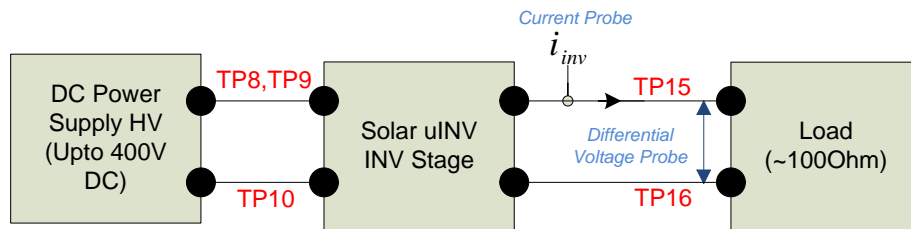


Figure 41 Build Level 2 Power Stage Connections for Inverter Closed Current Loop Check without Grid

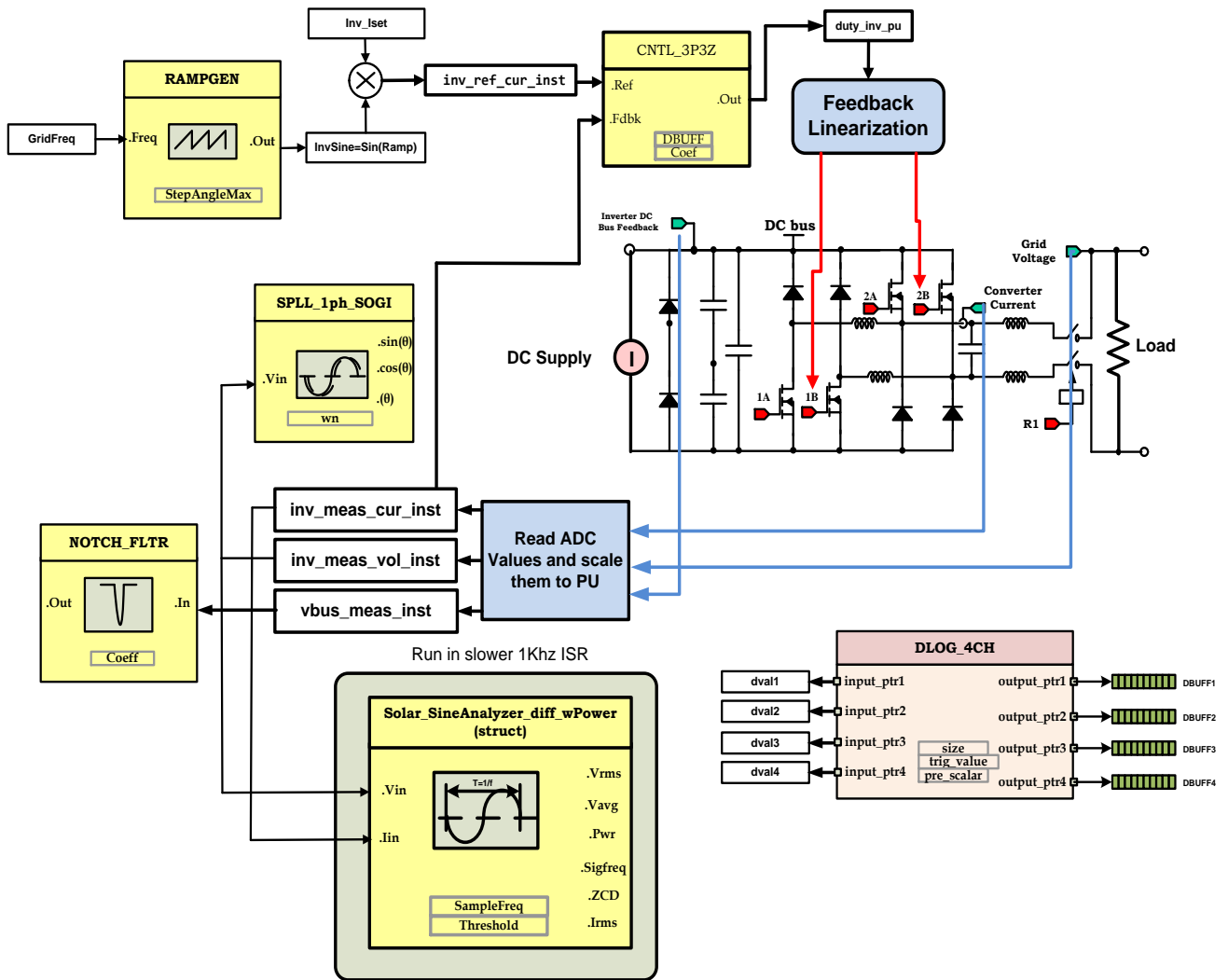


Figure 42 Build Level 2 Software Diagram for Inverter Closed Current Loop Check without Grid


1. Follow the steps from the previous sections to import the micro inverter project into CCS.
2. First to change the build level; open the *SolarMicroInv-Settings.h* file. Make sure the #defines are as below, if not modify to match below.

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

Note: Whenever you change the incremental build option always do a “Rebuild All”.

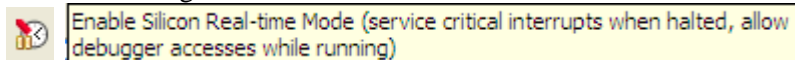
3. Click Project → “Rebuild All” button and watch the tools run in the build window.
4. Click Target → ”Debug Active Project”. The program will be loaded into the flash. You should now be at the start of Main().


Debug Environment Windows

5. Click View->Scripting Console to open the scripting console and open the “AddWatchWindowVars_IncrBuild2.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 Continuous Refresh  button on the watch window to enable continuous update of values from the controller.


Using Real-time Emulation

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



7. A message box *may* appear. If so, select YES to enable debug events. This will set bit 1 (DGBM bit) of status register 1 (ST1) to a “0”. The DGBM is the debug enable mask bit. When the DGBM bit is set to “0”, memory and register values can be passed to the host processor for updating the debugger windows.
8. Click on Continuous Refresh buttons  for the watch view.

Run the Code

9. Run the project, 
10. In the watch view, check the value of Gui_Vbus this will display the input voltage at the inverter bus which is zero.
11. Now turn on the HV DC supply and increase the input voltage to ~300V, verify reading from the watch expressions of the DC Bus to match the input.
12. Now connect the relay by writing a ‘1’ to RlyConnect variable in the watch window
13. Now set the current command for the inverter by writing pu value in the variable inv_Iset. For example write _IQ24(0.1).
14. Clear the inverter trip by writing a ‘1’ to ‘ClearInvTrip’
15. This will start the inverter closed current loop operation. Gradually increase the current command to be 0.25
16. The closed loop operation can further be verified by viewing the DBUFF3 and DBUFF4 values in the graph window as they are displaying 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. The graph below shows the measured and reference current for a 0.25 pu current command. You can observe the graphs change by changing the inv_Iset command. However always make sure the load power rating matches the power you are trying to pump into the resistor.

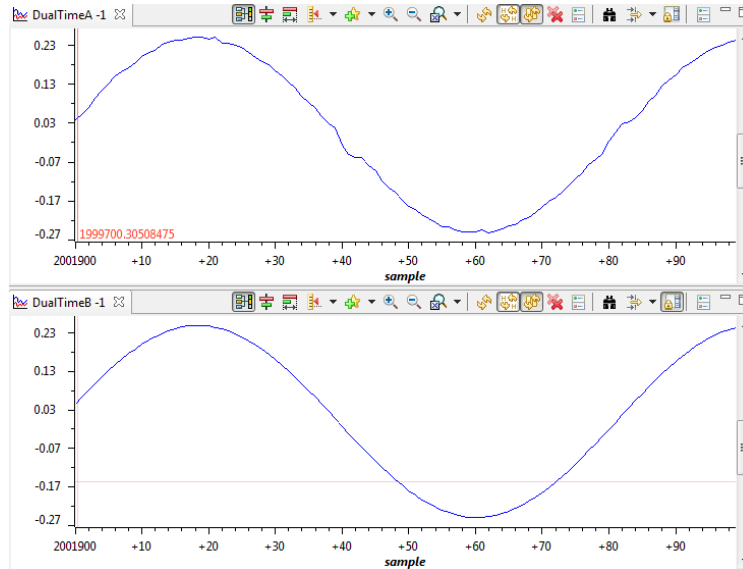






Figure 43 CCS Graph Window of Closed Current Loop Operation

17. This completes the closed current loop test for the inverter.
18. Now set the current command i.e. `inv_Iset` to zero.
19. Disconnect the inverter from the load by writing a '0' to `RlyConnect`
20. Reduce the DC bus to zero.
21. Fully halting the MCU when in real-time mode is a two-step process. Now, 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 .
22. Close CCS debug session by clicking on Terminate Debug Session  (Target->Terminate all).

6.2.4 Inverter Closed Current Loop with GRID

Procedure

In this build the closed current loop operation for the PV inverter is tested when a AC source/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 panel is not used which is a truly floating power supply the ground loop from the DC power supply and the AC source must be disconnected.

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

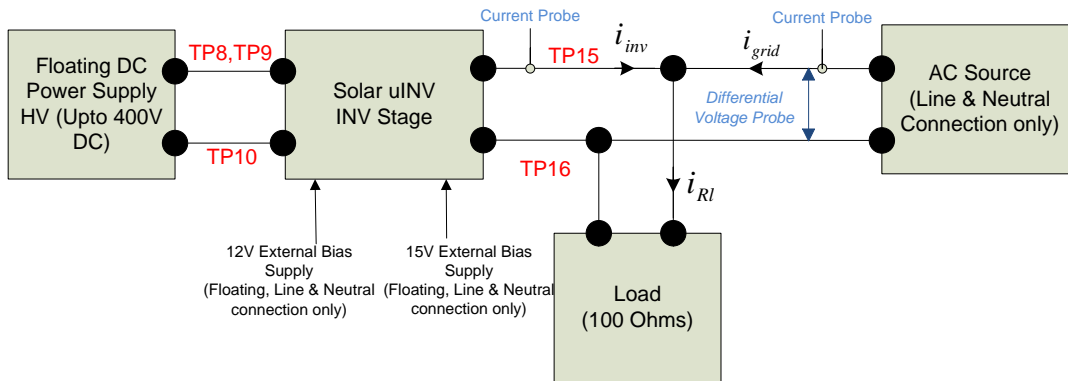


Figure 44 Inverter Closed Loop Current Check with Grid Connection Build Level 2

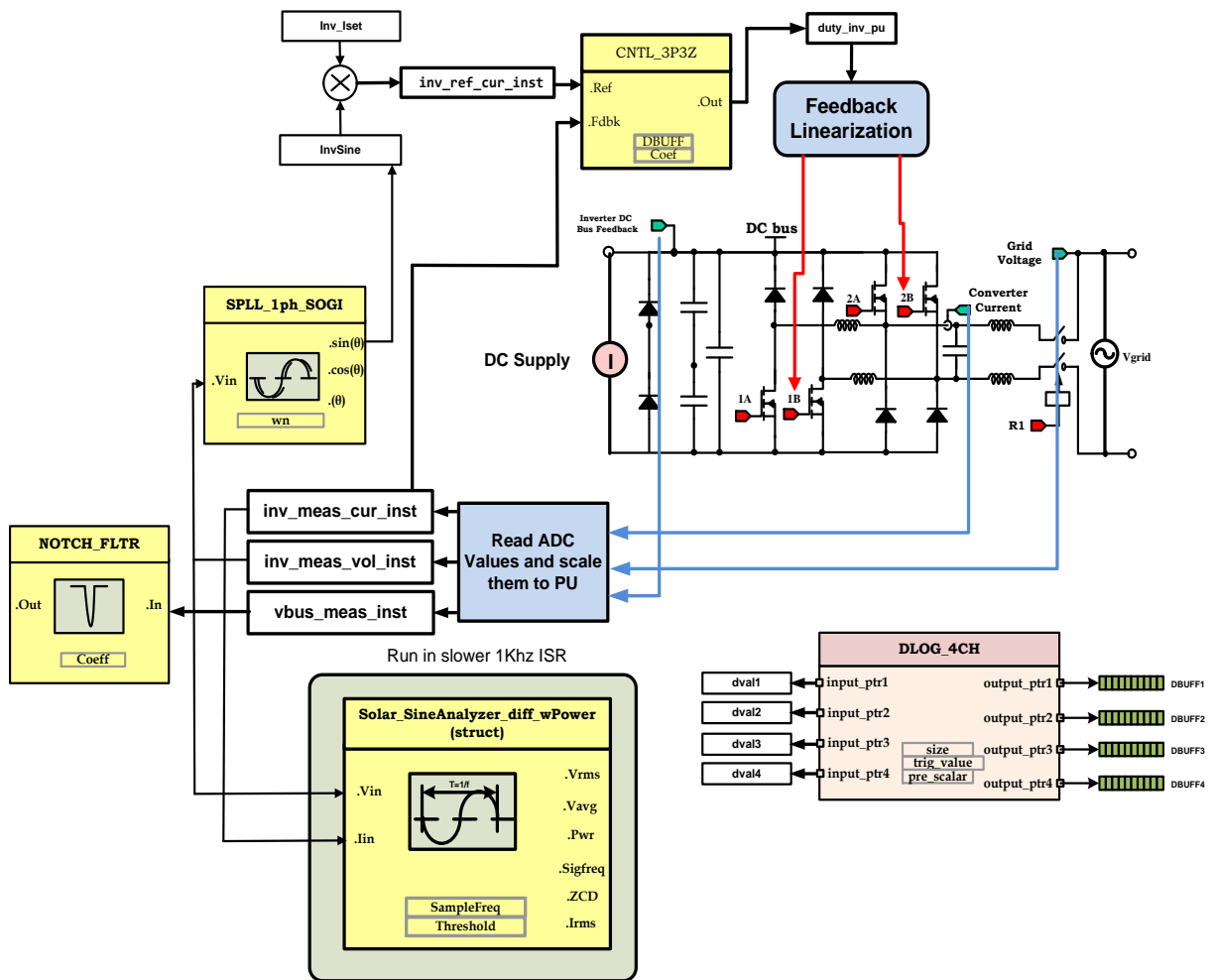


Figure 45 Software Diagram for Closed Current Loop Inverter with Grid Connection

1. Follow the steps from the previous sections to import the micro inverter project into CCS.
2. Now change the build level by opening the *SolarMicroInv-Settings.h* file. Make sure the #defines are as below if not modify to match below. Also modify the grid frequency to be 60Hz.


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

#define GRID_FREQ 60
```

Note: Whenever you change the incremental build option always do a “Rebuild All”.

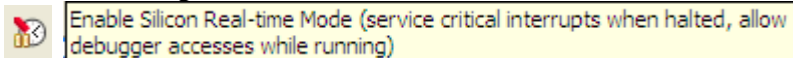
3. Click Project → “Rebuild All” button and watch the tools run in the build window.
4. Click Target → ”Debug Active Project”. The program will be loaded into the flash. You should now be at the start of Main().


Debug Environment Windows

5. Click View->Scripting Console to open the scripting console and open the “AddWatchWindowVars_IncrBuild2.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 Continuous Refresh  button on the watch window to enable continuous update of values from the controller.


Using Real-time Emulation

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







7. A message box *may* appear. If so, select YES to enable debug events. This will set bit 1 (DGBM bit) of status register 1 (ST1) to a “0”. The DGBM is the debug enable mask bit. When the DGBM bit is set to “0”, memory and register values can be passed to the host processor for updating the debugger windows.
8. Click on Continuous Refresh buttons  for the watch view.

Run the Code

9. Run the project, 
10. In the watch view, check the value of Gui_Vrms and Gui_Vbus this will 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.
11. Now slowly increase the input voltage to ~300V, verify that Gui_Vbus reads 300V
12. Now turn the AC source on and make sure the frequency is 60Hz and rms voltage is 110.
13. Gui_Vrms will show ~110V , the power drawn from the DC power supply will be close to 120W.
14. Now set the current command for the inverter by writing pu value in the variable inv_Iset. For example write _IQ24(0.1).
15. Next connect the relay by writing a ‘1’ to the RlyConnect variable in the watch window

16. Clear the inverter trip by writing a '1' to 'ClearInvTrip'
17. This will start the inverter and it will start feeding current into the load, this will make the power drawn from the AC load to drop. NOTE: with 0.1pu for the inv_Iset, the power drawn from the load will drop to be ~66W.

CAUTION: the AC source cannot take reverse power flow hence make sure you do not increase the inv_Iset such that the micro inverter board feeds power into the AC source, this can damage the AC power supply. The max value for the setup described above is ~0.2pu.

18. The closed loop operation can further be verified by viewing the DBUFF3 and DBUFF4 in graph window. DBUFF3 and DBUFF4 are displaying the reference current instantaneous and measured current instantaneous. Real time graphs can be added using the procedure outlined in 6.1.2.
19. The inv_Iset(max is 0.2 pu for the setup described above) can be changed and performance of the grid connected closed current loop inverter verified.
20. This completes the closed current loop with grid test for the inverter.
21. To stop operation put RlyConnect to '0'
22. Now set the current command i.e. inv_Iset to zero.
23. Reduce the AC voltage to zero
24. Reduce the DC bus to zero.
25. Fully halting the MCU when in real-time mode is a two-step process. Now, 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 .
26. Close CCS debug session by clicking on Terminate Debug Session  (Target->Terminate all).

6.3 BUILD = 3

Objective:

The objective of the build is to run the full PV inverter system with closed current loop and DC bus voltage control. Additionally 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 users to first 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.

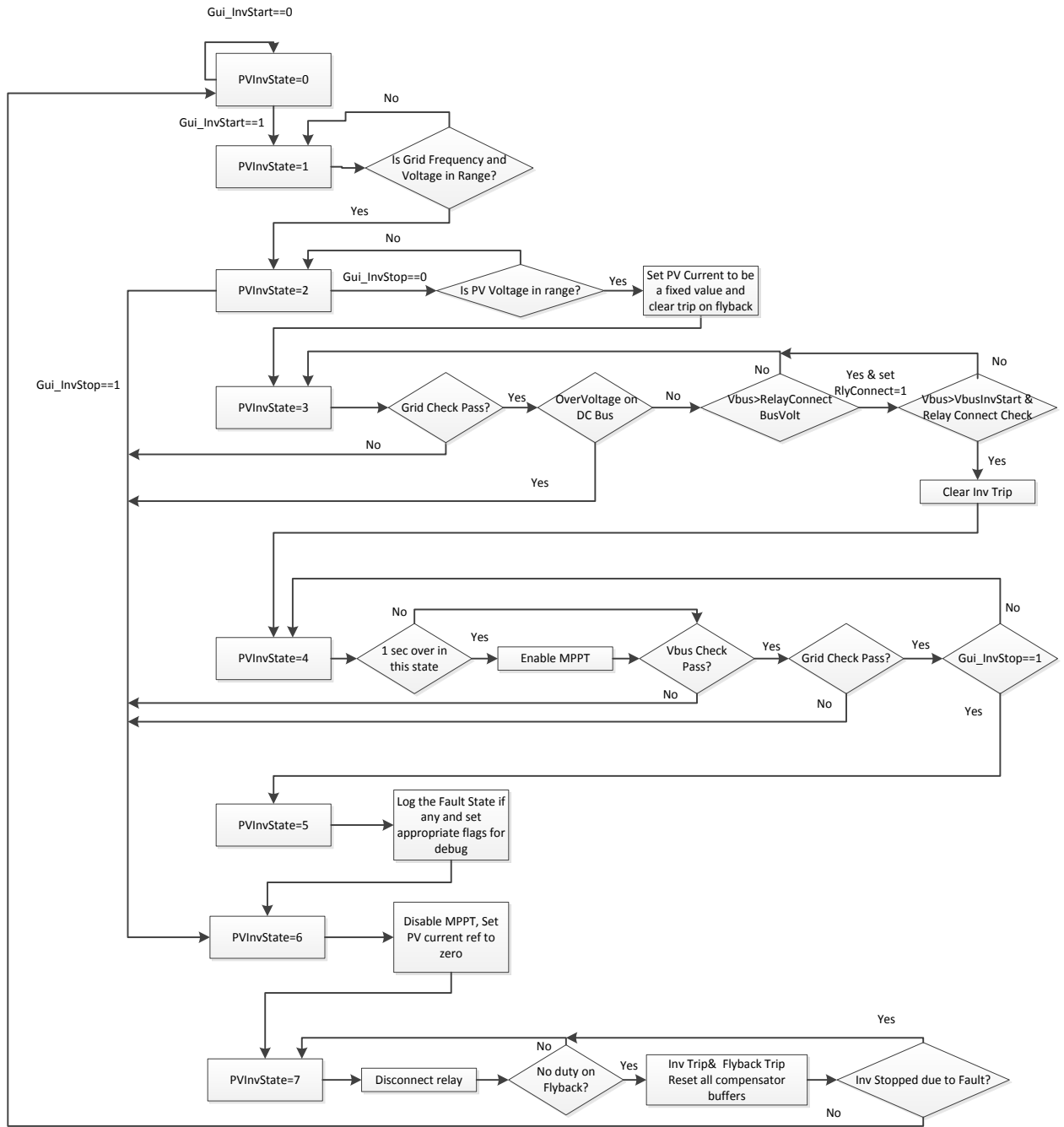


Figure 46 State Machine for Grid Connection

6.3.1 Full PV Inverter System Check **without MPPT**

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 below shows the hardware setup with the two stages connected together.

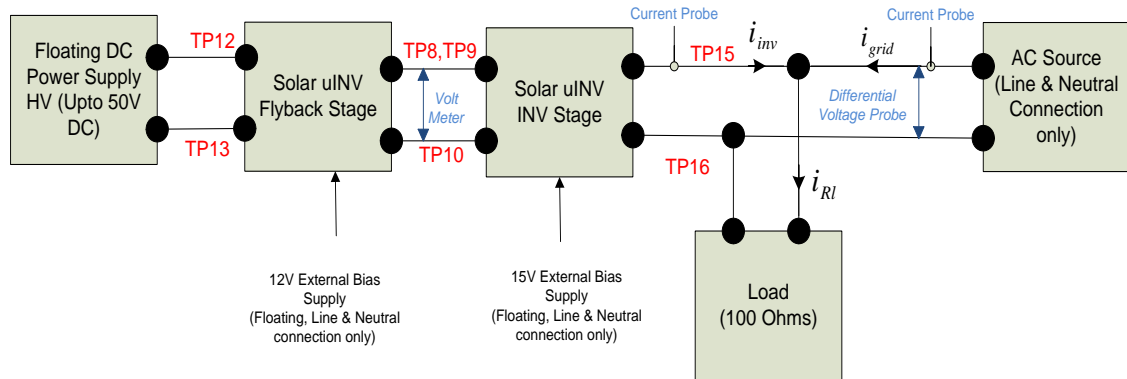


Figure 47 Full PV Inverter System Hardware Check Build 3 without MPPT

1. Follow the steps from the previous sections to import the micro inverter project into CCS.
2. Now change the build level by opening the *SolarMicroInv-Settings.h* file. Make sure the #defines are as below if not modify to match below. Also modify the grid frequency to be 60Hz.


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

#define GRID_FREQ 60
```

Note: Whenever you change the incremental build option always do a “Rebuild All”.

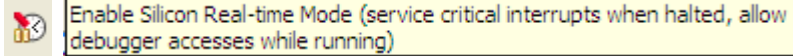
3. Click Project → “Rebuild All” button and watch the tools run in the build window.
4. Click Target → ”Debug Active Project”. The program will be loaded into the flash. You should now be at the start of Main().


Debug Environment Windows

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


Using Real-time Emulation

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



7. A message box *may* appear. If so, select YES to enable debug events. This will set bit 1 (DGBM bit) of status register 1 (ST1) to a “0”. The DGBM is the debug enable mask bit. When the DGBM bit is set to “0”, memory and register values can be passed to the host processor for updating the debugger windows.
8. Click on Continuous Refresh buttons  for the watch view.





Run the Code

9. Run the project, 
10. In the watch view, check the value of Gui_Vrms and Gui_Vpv this will display the input voltage at the flyback stage and the AC voltage source rms value at the micro inverter output.
11. Now turn the AC source on and make sure the frequency is 60Hz and rms voltage is 110
12. Gui_Vrms will show ~110V, if the resistive load is used the power drawn from the DC power supply will be close to 120W.
13. Now turn on the DC power supply and increase the voltage to be 30V. Verify this by inspecting the Gui_Vpv variable in the watch window.
14. The inverter is turned on by writing a ‘1’ to Gui_InvStart
15. This will trigger the state machine of the inverter and the PVInverterState will go to PVInverterActive.

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	0x0006896@Data
EPwm1Regs.TZFLG.all	unsigned int	0	0x0006816@Data
EPwm2Regs.TZFLG.all	unsigned int	0	0x0006856@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

Figure 48 Watch expression for Build 3, PV Inverter Check without MPPT

16. 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. Notice the DC bus is regulated at ~275V. This is specified in the SolarMicroInv-Settings.h file, irrespective of the current command from the DC power supply.
17. To stop the inverter operation write a ‘1’ to Gui_InvStop

18. Reduce the DC power supply to zero
19. Remove the AC power source connected at the output of the micro inverter board.
20. **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 safe voltage level ~3-4V. This may take a few minutes.**
21. This completes the PV inverter evaluation without MPPT.
22. To stop operation put RlyConnect to '0'
23. Now set the current command i.e. inv_Iset to zero.
24. Reduce the AC voltage to zero
25. Reduce the DC bus to zero.
26. Fully halting the MCU when in real-time mode is a two-step process. Now, 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 .
27. Close CCS debug session by clicking on Terminate Debug Session  (Target->Terminate all).

6.3.2 Full PV Inverter System Check **with MPPT**

Overview:

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

- a. Open Circuit Voltage 40V
- b. Maximum Power Point Voltage 30V
- c. Short Circuit Current 5.2Amps
- d. Maximum Power Point Current 4Amps

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

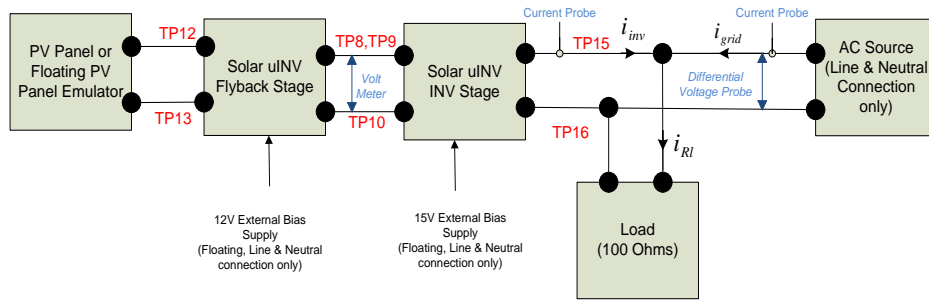


Figure 49 Hardware setup for Build level 4, PV Inverter system check with MPPT

Procedure:

1. Follow the steps from the previous sections to import the micro inverter project into CCS.
2. Now change the build level open the *SolarMicroInv-Settings.h* file. Make sure the defines are as below if not modify to match below. Also modify the grid frequency to be 60Hz.


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

#define GRID_FREQ 60
```

Note: Whenever you change the incremental build option always do a “Rebuild All”.

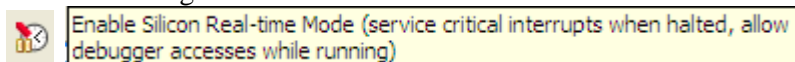
3. Click Project → “Rebuild All” button and watch the tools run in the build window.
4. Click Target → ”Debug Active Project”. The program will be loaded into the flash. You should now be at the start of Main().

Debug Environment Windows


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

Using Real-time Emulation


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

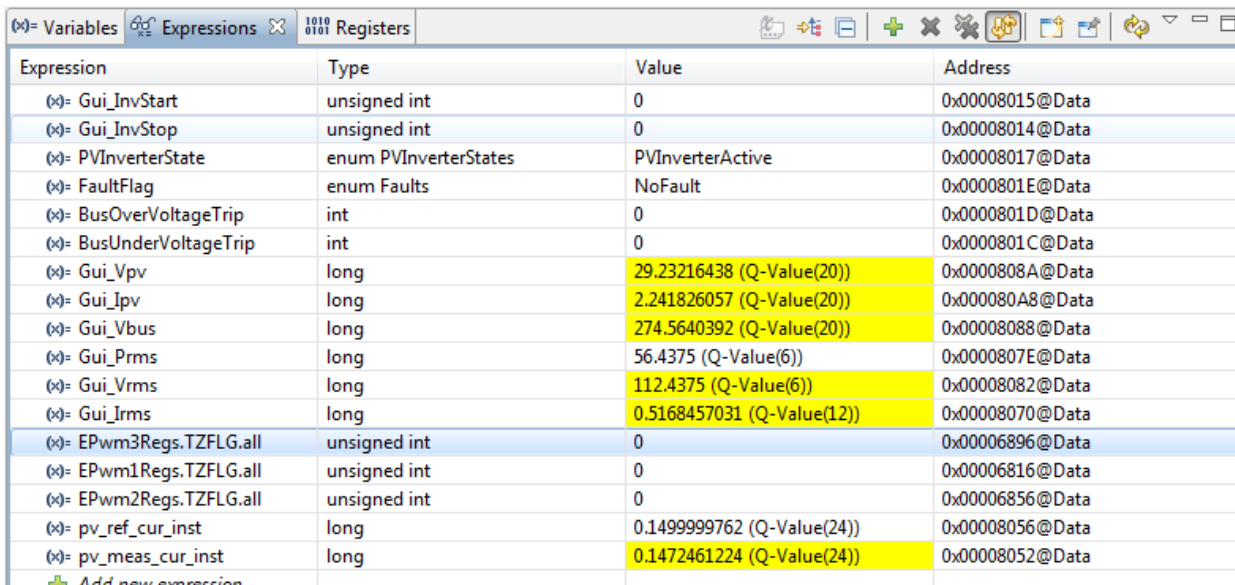


7. A message box *may* appear. If so, select YES to enable debug events. This will set bit 1 (DGBM bit) of status register 1 (ST1) to a “0”. The DGBM is the debug enable mask bit. When the DGBM bit is set to “0”, memory and register values can be passed to the host processor for updating the debugger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

- Click on Continuous Refresh buttons  for the watch view.

Run the Code

- Run the project, 
- In the watch view, check the value of Gui_Vrms and Gui_Vpv this will display the input voltage at the flyback stage and the AC voltage source rms value at the micro inverter output.
- Now turn the AC source on and make sure the frequency is 60Hz and rms voltage is 110
- Gui_Vrms will show ~110V, if the resistive load is used the power drawn from the DC power supply will be close to 120W.
- Now turn on the panel emulator or connect the panel to the input of the board.
- The inverter is turned on by writing a '1' to Gui_InvStart
- This will trigger the state machine of the inverter and the PVInverterState will go to PVInverterActive.



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

Figure 50 Watch expressions for build level 3, PV inverter system check with MPPT

- You will notice the current from the panel slowly increases to the MPP. If PV panel emulator is used different shading conditions can be tested and the transient behavior also evaluate. Notice the DC bus is regulated at ~275V. This is specified in the SolarMicroInv-Settings.h file, irrespective of the current command from the DC power supply. The figure below shows the waveform of Grid Voltage and inverter current fed into the grid when connected to a ~120W MPP panel emulator.

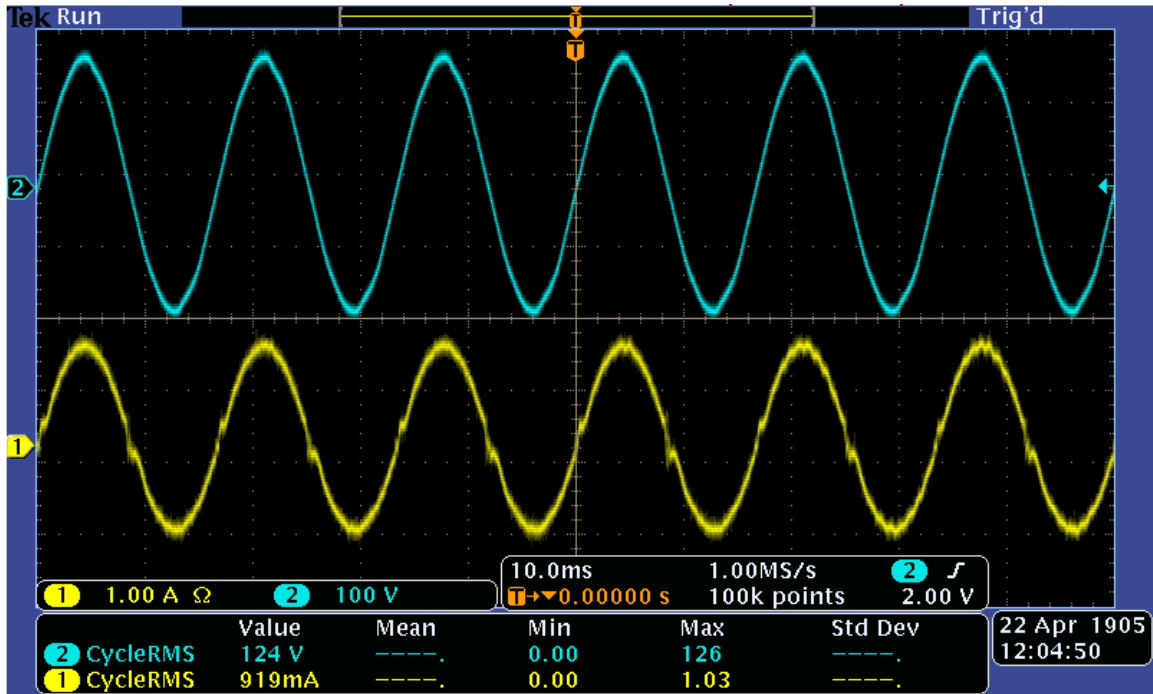


Figure 51 Build level 3, grid connection check voltage and current waveform scope capture

The Figure below is capture of inverter start and inverter stop command.

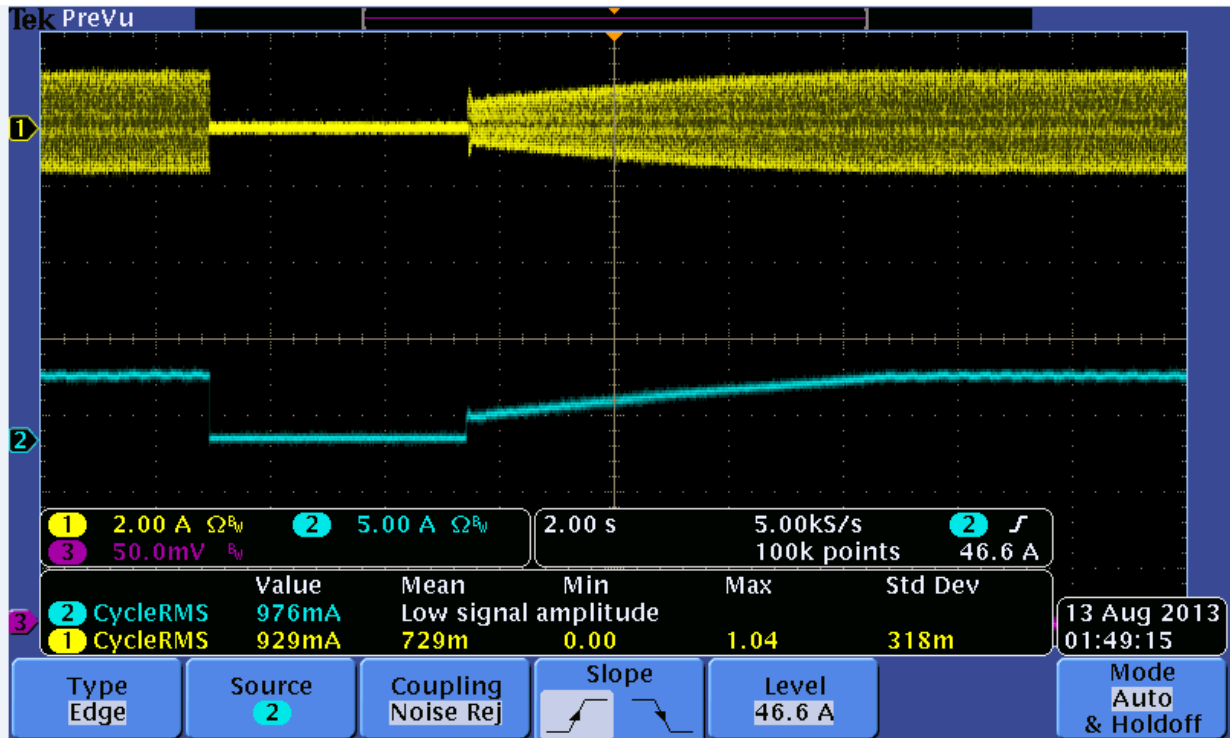






Figure 52 Stop and start sequence of the PV Micro Inverter scope capture

17. To stop the inverter operation write a '1' to Gui_InvStop
18. Reduce the DC power supply to zero

19. Remove the AC power source connected at the output of the uInv board.
20. **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 safe voltage level ~3-4V. This may take a few minutes.**
21. This completes the PV inverter evaluation without MPPT.
22. To stop operation put RlyConnect to '0'
23. Now set the current command i.e. inv_Iset to zero.
24. Reduce the AC voltage to zero
25. Reduce the DC bus to zero.
26. Fully halting the MCU when in real-time mode is a two-step process. Now, 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 .
27. Close CCS debug session by clicking on Terminate Debug Session  (Target->Terminate all).

7 References

For more information please refer to the following guides:

- **Solar Micro Inverter-GUI-QSG** – A quick-start guide for quick demo of the Solar Micro Inverter EVM using a GUI interface.
..\controlSUITE\development_kits\Solar_Micro_Inverter\~Docs\QSG_Solar_Micro_Inverter_GUI_Rev1.0.pdf
- **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\Solar_Micro-Inverter\Solar_Micro-Inverter_HWDevPkg
- **F28xxx User's Guides**
<http://www.ti.com/f28xuserguides>

IMPORTANT NOTICE FOR TI REFERENCE DESIGNS

Texas Instruments Incorporated ("TI") reference designs are solely intended to assist designers ("Buyers") who are developing systems that incorporate TI semiconductor products (also referred to herein as "components"). Buyer understands and agrees that Buyer remains responsible for using its independent analysis, evaluation and judgment in designing Buyer's systems and products.

TI reference designs have been created using standard laboratory conditions and engineering practices. **TI has not conducted any testing other than that specifically described in the published documentation for a particular reference design.** TI may make corrections, enhancements, improvements and other changes to its reference designs.

Buyers are authorized to use TI reference designs with the TI component(s) identified in each particular reference design and to modify the reference design in the development of their end products. HOWEVER, NO OTHER LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE TO ANY OTHER TI INTELLECTUAL PROPERTY RIGHT, AND NO LICENSE TO ANY THIRD PARTY TECHNOLOGY OR INTELLECTUAL PROPERTY RIGHT, 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 components or services are used. Information published by TI regarding third-party products or services does not constitute a license to use such products or services, or a warranty or endorsement thereof. Use of such information 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 REFERENCE DESIGNS ARE PROVIDED "AS IS". TI MAKES NO WARRANTIES OR REPRESENTATIONS WITH REGARD TO THE REFERENCE DESIGNS OR USE OF THE REFERENCE DESIGNS, EXPRESS, IMPLIED OR STATUTORY, INCLUDING ACCURACY OR COMPLETENESS. TI DISCLAIMS ANY WARRANTY OF TITLE AND ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, QUIET ENJOYMENT, QUIET POSSESSION, AND NON-INFRINGEMENT OF ANY THIRD PARTY INTELLECTUAL PROPERTY RIGHTS WITH REGARD TO TI REFERENCE DESIGNS OR USE THEREOF. TI SHALL NOT BE LIABLE FOR AND SHALL NOT DEFEND OR INDEMNIFY BUYERS AGAINST ANY THIRD PARTY INFRINGEMENT CLAIM THAT RELATES TO OR IS BASED ON A COMBINATION OF COMPONENTS PROVIDED IN A TI REFERENCE DESIGN. IN NO EVENT SHALL TI BE LIABLE FOR ANY ACTUAL, SPECIAL, INCIDENTAL, CONSEQUENTIAL OR INDIRECT DAMAGES, HOWEVER CAUSED, ON ANY THEORY OF LIABILITY AND WHETHER OR NOT TI HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES, ARISING IN ANY WAY OUT OF TI REFERENCE DESIGNS OR BUYER'S USE OF TI REFERENCE DESIGNS.

TI reserves the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its components to the specifications applicable at the time of sale, in accordance with the warranty in TI's terms and conditions of sale of semiconductor products. Testing and other quality control techniques for TI components are used to the extent TI deems necessary to support this warranty. Except where mandated by applicable law, testing of all parameters of each component is not necessarily performed.

TI assumes no liability for applications assistance or the design of Buyers' products. Buyers are responsible for their products and applications using TI components. To minimize the risks associated with Buyers' products and applications, Buyers should provide adequate design and operating safeguards.

Reproduction of significant portions of TI information in TI data books, data sheets or reference designs is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Buyer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of TI components in its applications, notwithstanding any applications-related information or support that may be provided by TI. Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards that anticipate dangerous failures, monitor failures and their consequences, lessen the likelihood of dangerous failures and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in Buyer's safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI's goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed an agreement specifically governing such use.

Only those TI components that TI has specifically designated as military grade or "enhanced plastic" are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components that have **not** been so designated is solely at Buyer's risk, and Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI has specifically designated certain components as meeting ISO/TS16949 requirements, mainly for automotive use. In any case of use of non-designated products, TI will not be responsible for any failure to meet ISO/TS16949.