Non Isolated Bi-Directional Buck Boost Converter Design using C2000 Microcontroller

Version 1.1 – January 27, 2015
Manish Bhardwaj
C2000 System Solution

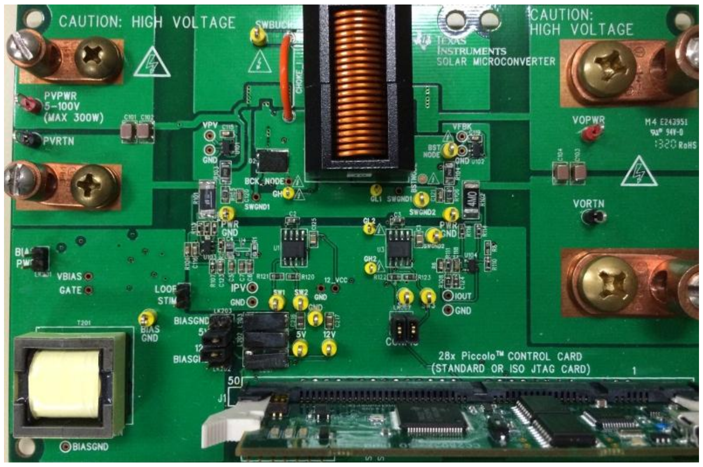


Fig 1 Bi Directional Non Isolated Buck Boost Board

1. Introduction

The C2000 Bidirectional Non Isolated Buck Boost TI Design (TIDM_BUCKBOOST_BIDIR, Fig 1), illustrates the use of C2000 microcontroller for controlling a bi-directional non isolate buck boost power stage. This power stage can also be used for an emerging concept of solar micro converters. The document goes over the design of the board, hardware implementation and software structure that is used to control the power stage.

Contents

2. Getting Familiar with the Design

The design follows a control card concept and hence any device from the C2000 family with DIMM100 control card can be used. The software associated with the design uses the F28035 MCU control card with isolated JTAG.

2.1 Design Overview

Fig 2 illustrates the DC-DC buck boost topology implemented using the UCC27211 driver. The power stage is comprised of 4 FET'S. SW1 & SW2 form the buck pair and SW3 & SW4 form the boost pair and are switched complimentary to each other, when power is being transferred from V1 to V2. When the power flows in the reverse direction, SW4 and SW3 form the buck pair and SW2 and SW1 form the boost pair. The board comprises of an on board power supply that can power the supporting control, power processing and sensing circuit including the microcontroller, this bias is powered from the Vpv/V1 side.

Looking at the power stage the ideal DC gain of the converter is given by equation:

$$G = \frac{V_o}{V_i} = \frac{D_{bu}}{1 - D_{bo}} \text{.............................. Equation 1}$$

Where D_{bu} is the duty cycle of the switch SW1 and D_{bo} is the duty cycle of the switch SW3 when power is transferred from V1 to V2. Because of MOSFET usage and the symmetric power stage the kit can be used for bi-directional buck boost type application.

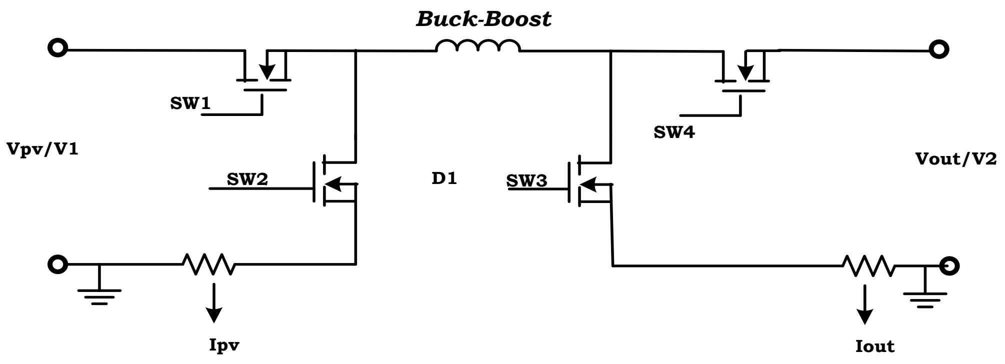


Fig 2 DC-DC Buck Boost Power Stage

Fig 3 shows the positioning of key components on the kit base board.

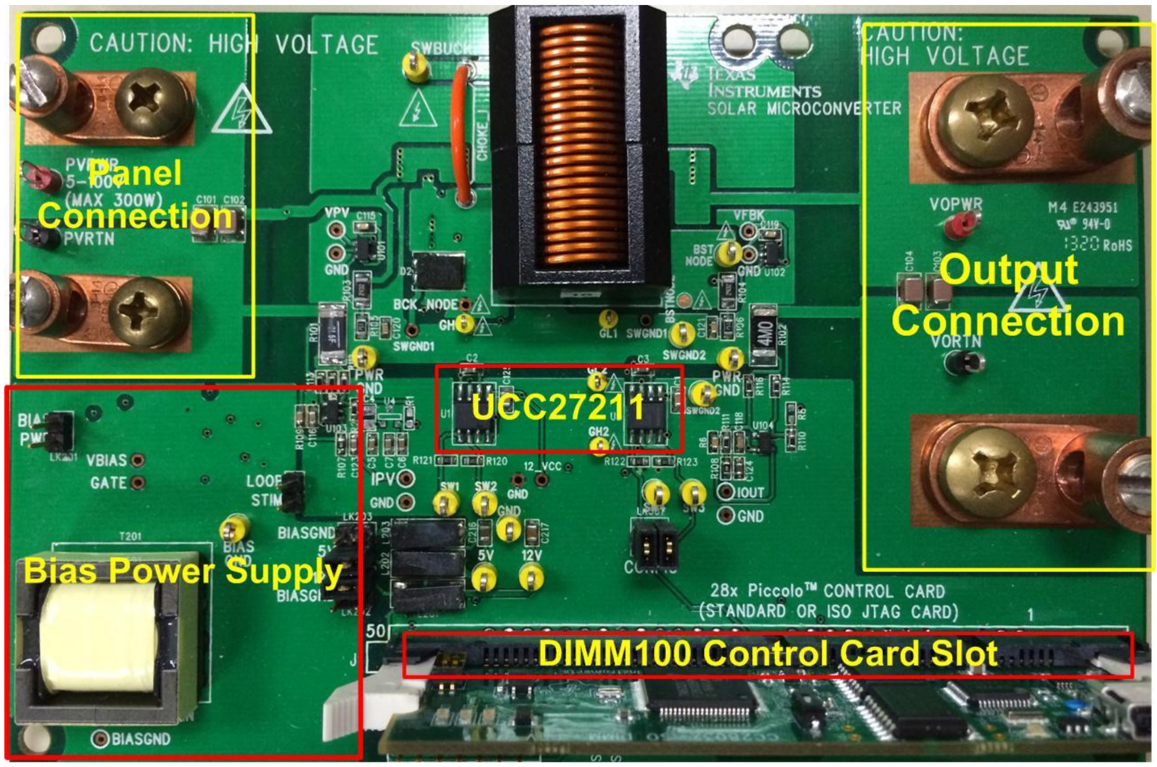

Panel Connection
Output Connection
UCC27211
Bias Power Supply
DIMM100 Control Card Slot
CAUTION: HIGH VOLTAGE
TEXAS INSTRUMENTS SOLAR MICROCONVERTER
28x Piccolo™ CONTROL CARD (STANDARD OR ISO JTAG CARD)

Fig 3 Bi Directional Buck Boost board key sections highlighted

Power Stage Parameters

Input Voltage : 10-100V (Panel Input/ V1) DC max
Input Current : 0-8Amps (Panel Input). Input power has to be <300W.
Output Voltage : 5-100V DC max
Output Current: 0-8Amps.
Power Rating: 300W Max
Fsw : 250Khz

2.2 Switching Scheme

The switch pairs, SW1 & SW2 and SW3 & SW4, are switched complimentary to each other thus giving synchronous buck and boost. The switching scheme for the switches is illustrated in Fig 4.

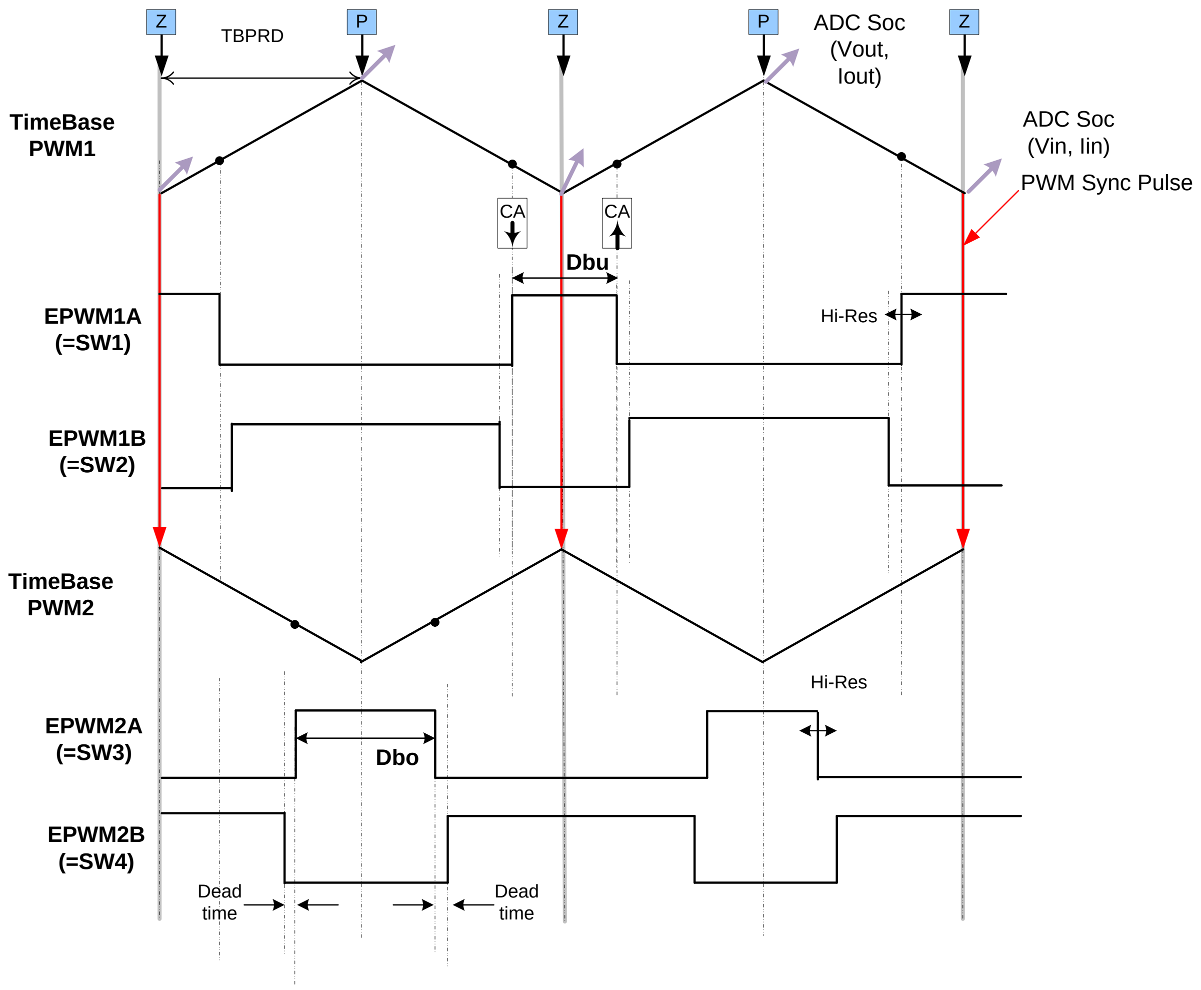


Fig 4 Switching Scheme for Synchronous Buck Boost Power Stage

A novel switching scheme as proposed in [High-Efficiency, Wide-Load Range Buck/Boost Photovoltaic Microconverter," Dave Freeman, Dick Hester et al, APEC 2010] is used to switch the power stage, to get the maximum efficiency. The operation range of the stage is split into 4 regions, Table 1, and the duty value computed for each region such that the gain increases monotonically between the different modes.

Table 1 Mode of Operation of the Buck Boost Stage and Duty Calculation

Duty/ Gain needed	Dbu	Dbo	Mode
0-BuckModeMaxGain	Gain	0	**Buck Mode**
BuckModeMaxGain-ConstantBoostMaxGain	Gain-C1	ConstantBoostDuty	**Constant Boost, Buck Mode**
ConstantBoostMaxGain-ConstantBuckMaxGain	ConstantBuckDuty	Gain-C2	**Constant Buck, Boost Mode**
ConstantBuckMaxGain – up	1.0	Gain –C3	**Boost Mode**

Note that the boundary values for changing from one mode to the other such as BuckModeMaxGain, ConstantBoostMaxGain and ConstantBuckMaxGain are determined empirically to get the maximum efficiency. Also the values to calculate the Duty for the Buck and the Boost switches such as C1, C2, C3 etc are determined empirically to make sure the gain increases monotonically (equation 1 alone cannot provide this information as it does not take into account the minimum on time etc for the switches). The values can be empirically tuned to get the best efficiency as they are readily available as variables in the software. See the Gain_Worksheet.xlsx provided with the design for the values used with this design.

2.3 Understanding bi directional power flow

In this document the two directions of power flow are separate to facilitate the experimentation with a single quadrant voltage supply on one side and a load on the other. In practical application of bi-directional power flow two energy storage elements are connected on either side and the power flow is simply a function of the duty cycle. This is illustrated in this section by means of a synchronous boost power stage, however the reader can extend the concepts to understand power flow in case of the full bridge.

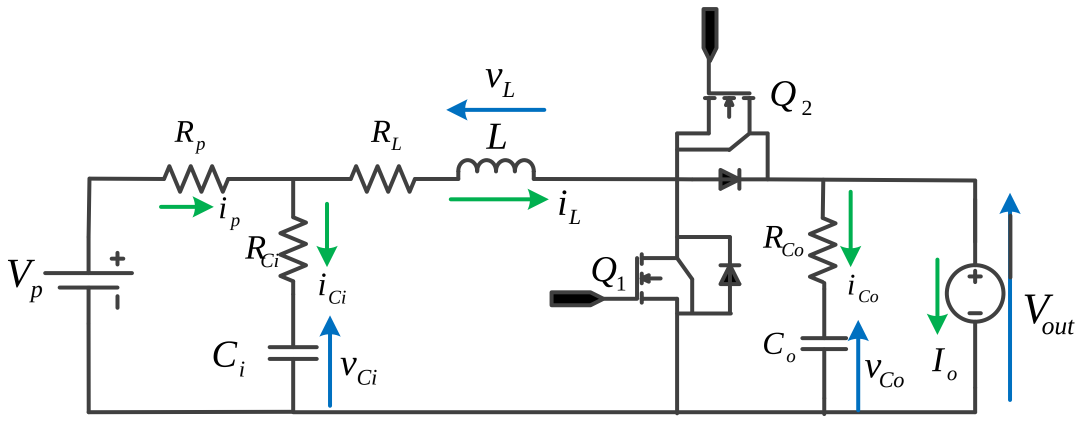


Fig 5 Synchronous Boost Stage with energy storgae elements at input and output

State space averaging is commonly applied to develop models for Switched Mode Power Supplies (SMPS). It is common to choose the **energy storage elements as the state of the system**, which is current for inductors and voltage for capacitors. Irrespective of charging or discharging there are only two sub-intervals (Q1 on and Q1 off), hence a **common model** can be developed for both the modes of power flow.

For the DC-DC converter, shown below, **state** (X) , **input** (U_i) and **output** (Y) vectors are chosen as follows:

$$X = \begin{bmatrix} Output\_Cap\_Voltage \\ Input\_Cap\_Voltage \\ Inductor\_Current \end{bmatrix} = \begin{bmatrix} v_{Co} \\ v_{Ci} \\ i_L \end{bmatrix}$$

$$U_i = \begin{bmatrix} Output\_Load \\ Input\_Voltage \end{bmatrix} = \begin{bmatrix} I_o \\ V_p \end{bmatrix}$$

$$Y = \begin{bmatrix} Input\_Current \\ Inductor\_Current \\ Output\_Voltage \end{bmatrix} = \begin{bmatrix} i_p \\ i_L \\ V_{out} \end{bmatrix}$$

Using KCL and KVL equation for each switch state and using state space averaging and separating the large signal and small signal terms the following equation can be realized:

$$\dot{X} = (A_1\overline{D} + A_2(1-\overline{D}))\overline{X} + (B_1\overline{D} + B_2(1-\overline{D}))\overline{U}_i = 0$$

$$= \begin{bmatrix} 0 & 0 & (1-\overline{D})/C_o \\ 0 & -\frac{1}{C_i(R_p+R_{Ci})} & -\frac{R_p}{C_i(R_p+R_{Ci})} \\ -(1-\overline{D})\frac{1}{L} & \frac{R_p}{L(R_p+R_{Ci})} & -\frac{(R_L R_p + R_L R_{Ci} + R_{Ci} R_P + (1-\overline{D})(R_p R_{Co} + R_{Co} R_{Ci}))}{L(R_p+R_{Ci})} \end{bmatrix} \begin{bmatrix} \bar{v}_{Co} \\ \bar{v}_{Ci} \\ \bar{i}_L \end{bmatrix} + \begin{bmatrix} -\frac{1}{C_o} & 0 \\ 0 & \frac{1}{C_i(R_p+R_{Ci})} \\ (1-D)\frac{R_{Co}}{L} & \frac{R_{Ci}}{L(R_p+R_{Ci})} \end{bmatrix} \begin{bmatrix} \bar{I}_o \\ \overline{V}_p \end{bmatrix}$$

Solving further the output current is given by:

$$\bar{I}_o = \frac{(1-\overline{D})^2}{R_p + R_L + \overline{D}(1-\overline{D})R_{Co}} \left[\frac{\overline{V}_p}{(1-D)} - \overline{V}_{out} \right]$$

From this equation it is clear that duty cycle control itself can force the current to be sinked or sourced from Vout.

3 Hardware Details

3.1 Resource Allocation

The Fig 6 shows the board in a block diagram format and illustrates the major connections and feedback values that are being mapped to the C2000 MCU. Table 2, below lists these resources. Note the table only lists the resources used for power stage and that are mapped to the DIMM100 connector on the board.

Table 2 Resource Mapping: PWM, ADC
(LK301 and LK302 jumpers in vertical (=normal) configuration).

Signal Name	PWM channel / ADC Channel No/ Resource Mapping F2803x	Function
INL1	EPWM-1B	PWM input, Buck Low Side
INH1	EPWM-1A	PWM input, Buck High Side
INL2	EPWM-2A	PWM input, Boost Low Side
INH2	EPWM-2B	PWM input, Boost High Side
VPV	ADCIN-B2	Panel Voltage Feedback
IPV	ADCIN-A2	Panel Current Feedback
VFBK	ADCIN-B4	Output Voltage Feedback
IOUT	ADCIN-A4	Output Current Feedback
12_VCC	ADCIN-B6	12V Board Supply Feedback
5_VCC	ADCIN-A6	5V Board Supply Feedback
3V3	ADCIN-A7	3.3V Board Supply Feedback

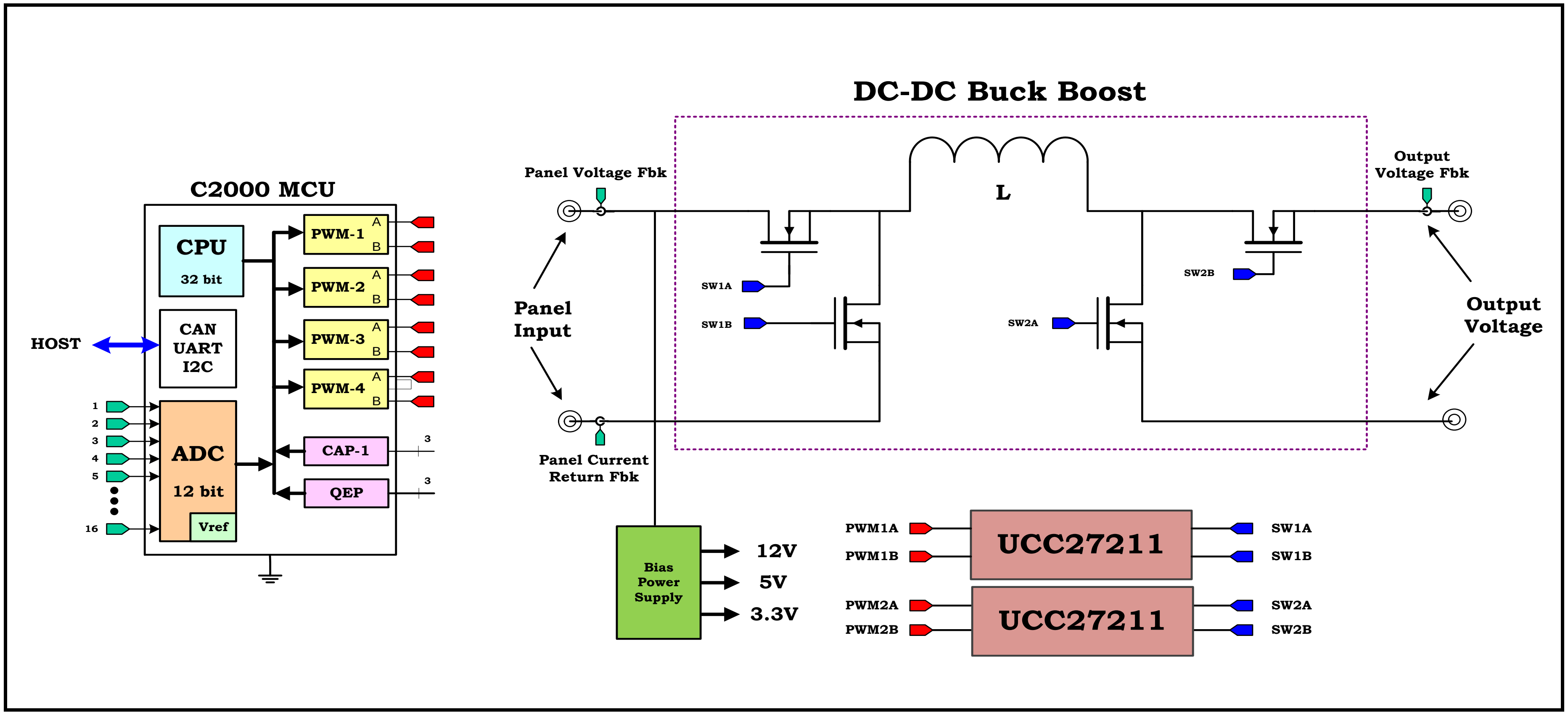


Fig 6 Kit Block Diagram with C2000 MCU (LK301 and LK302 jumpers in vertical (=normal) configuration).
(Connectivity Peripherals may differ from one device to the other including Ethernet, USB, CAN, SPI etc)

3.2 Jumpers and Connectors

Table 3 below lists the jumpers on the board,

Table 3 Jumpers & Connectors on the kit base Board

LK201	**Jumper – BIAS PWR. Connecting this jumper will enable the internal bias power generation on board (also requires LK202 and LK203 to be connected). When enabled, the bias power is drawn from the Panel Voltage (internal bias powers are enabled when PVIN rises above ~9V, and disabled when falls below ~8V).** **If this jumper is not enabled, then the 12V, 5V and 3.3V supplies need to be externally connected to the 5v and 12V test points (3.3V is generated from the 5V supply).** **Default: Connected.**
T201	Transformer for generating the internal bias powers when LK201, LK202, LK203 are connected.
LOOP STIIM	When connected, the internally generated bias supplies are slightly increased. Default: Not connected
LK202, LK203	Connects the Transformer (T201) output voltages to the on-board 12V (LK202), and 5V (LK203). 3.3V is generated from the 5V supply. To enable the voltages, connect the jumpers horizontally at the 5V and 12V labels (requires LK201 to be enabled). Default: Connected horizontally at 5V and 12V labels.
LK301	Jumpers for alternative connections. Vertical configuration (=normal): EPWM-2A controls Boost Low Side MOSFET. EPWM-2B controls Boost High Side MOSFET. Horizontal configuration (=alternative): EPWM-2A controls Boost High Side MOSFET. EPWM-2B controls Boost Low Side MOSFET. Default: Vertical connection.

4 Software

This section describes the details of software.

4.1 Project Framework

Power stage control requires a real time ISR's for the closed loop control of the DC-DC stage. The project makes use of the "C-background/C-ISR/ASM-ISR" framework. The fast ISR, controlling DC-DC Boost stage, runs in assembly environment using the Digital Power Library. The project uses C-code as the main supporting program for the application, and is responsible for all system management tasks, decision making, intelligence, and host interaction.

The project framework also integrates the Software Frequency Response Analyzer(SFRA) which the user can use to measure the frequency response of the power supply and verify if the control design has enough margin of stability. For more details on the SFRA refer to:

http://www.ti.com/tool/sfra

The key framework C files used in the project are:

BuckBoostBiDir-Main.c – this file is used to initialize, run, and manage the application. This contains the interrupt service routine that is used to run the control loop of the power stage.

BuckBoostBiDir-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.

BuckBoostBiDir-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.

BuckBoostBiDir-Includes.h – This file contains of all the header files used by the project.

BuckBoostBiDir-DPL.asm – This file contains time critical "control type" code. This file has an initialization section (one time execute) and a run-time section which executes at control loop execution rate. This routine is called from the C based ISR in Main.c file.

Fig 7 gives the structure of the software, with the main background loop, and the DC-DC ISR.

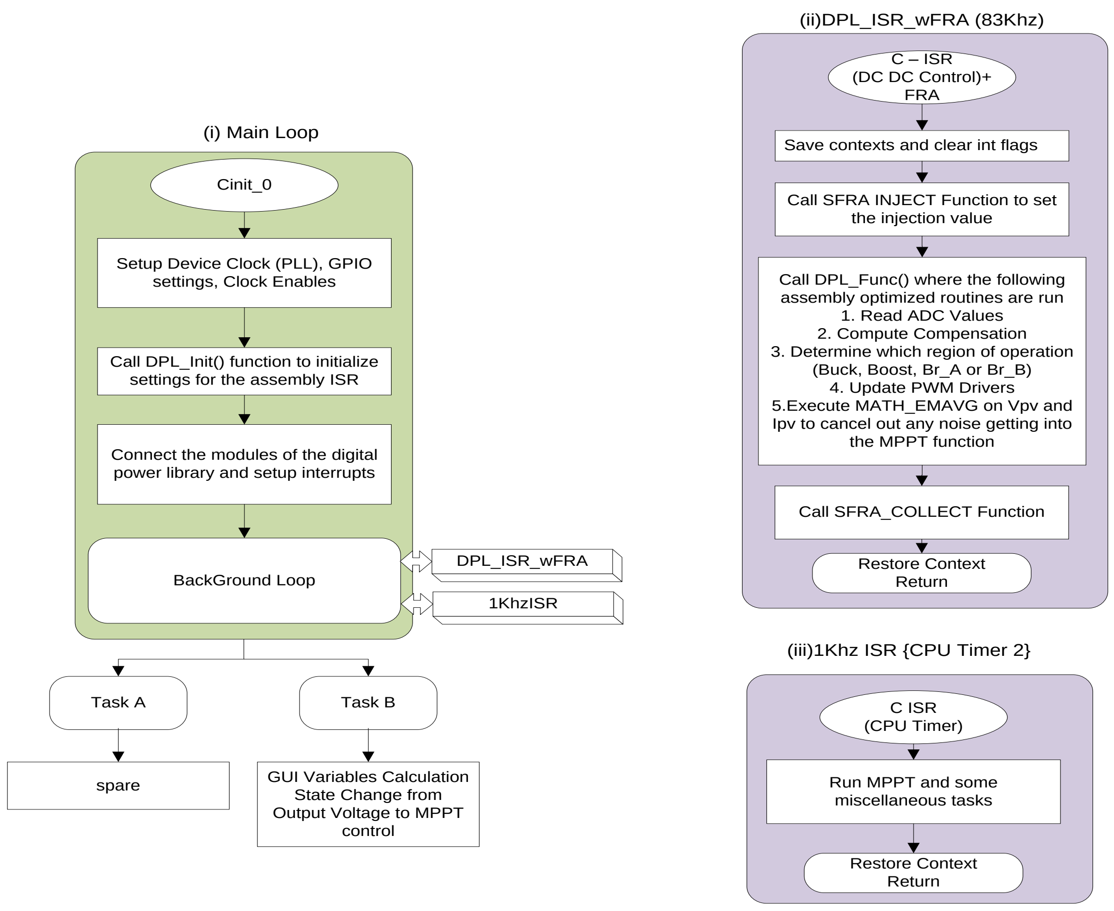

(i) Main Loop
Cinit_0
Setup Device Clock (PLL), GPIO settings, Clock Enables
Call DPL_Init() function to initialize settings for the assembly ISR
Connect the modules of the digital power library and setup interrupts
BackGround Loop
DPL_ISR_wFRA
1KhzISR
Task A
Task B
spare
GUI Variables Calculation State Change from Output Voltage to MPPT control
(ii)DPL_ISR_wFRA (83Khz)
C – ISR (DC DC Control)+ FRA
Save contexts and clear int flags
Call SFRA INJECT Function to set the injection value
Call DPL_Func() where the following assembly optimized routines are run 1. Read ADC Values 2. Compute Compensation 3. Determine which region of operation (Buck, Boost, Br_A or Br_B) 4. Update PWM Drivers 5.Execute MATH_EMAVG on Vpv and Ipv to cancel out any noise getting into the MPPT function
Call SFRA_COLLECT Function
Restore Context Return
(iii)1Khz ISR {CPU Timer 2}
C ISR (CPU Timer)
Run MPPT and some miscellaneous tasks
Restore Context Return

Fig 7 Software structure (i) Main loop (ii) DCDC ISR (iii) 1Khz ISR

4.2 Project Dependencies & Resources

Hardware Kit	: TIDM_BUCKBOOSTBIDR [Rev 1D]
Control Card	: Iso Jtag F28035
Software IDE	: CCSv6 or later
Control Suite Dependencies	:
Device Support (F28035 Header Files)	: controlSUITE\device_support\f2803x\v128
IQMath Library	: controlSUITE\libs\math\IQmath\v160

Digital Power Library :
controlSUITE\app_libs\digital_power\f2803x_v3.4

Solar Library :
controlSUITE\app_libs\solar\v1.2\IQ

SFRA Library
controlSUITE\app_libs\SFRA\v1_00_00_00\IQ

4.3 Control Description

The design provides for sensing the input and output voltage and input and output current. Several control schemes can be implemented. For simplicity the associated software implements the following three modes :

1. Output Voltage Control, Fig 8
2. MPPT Control, Input Current Control, Fig 11
3. Voltage control in the reverse direction, Fig 13

Output Voltage Control

For controlling the output voltage control scheme shown in Fig 8 is used. To output voltage is sensed, (V_{out}) and compared with the reference value (V_{out_ref}) and the 2p2z compensator calculated the effort needed. The voltage controller is executed at a rate of 83 kHz (1/3rd of the PWM switching frequency).

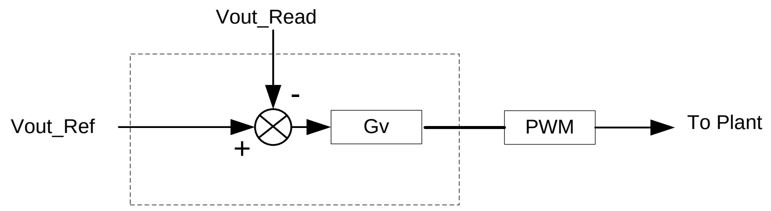


Fig 8 Output Voltage Control

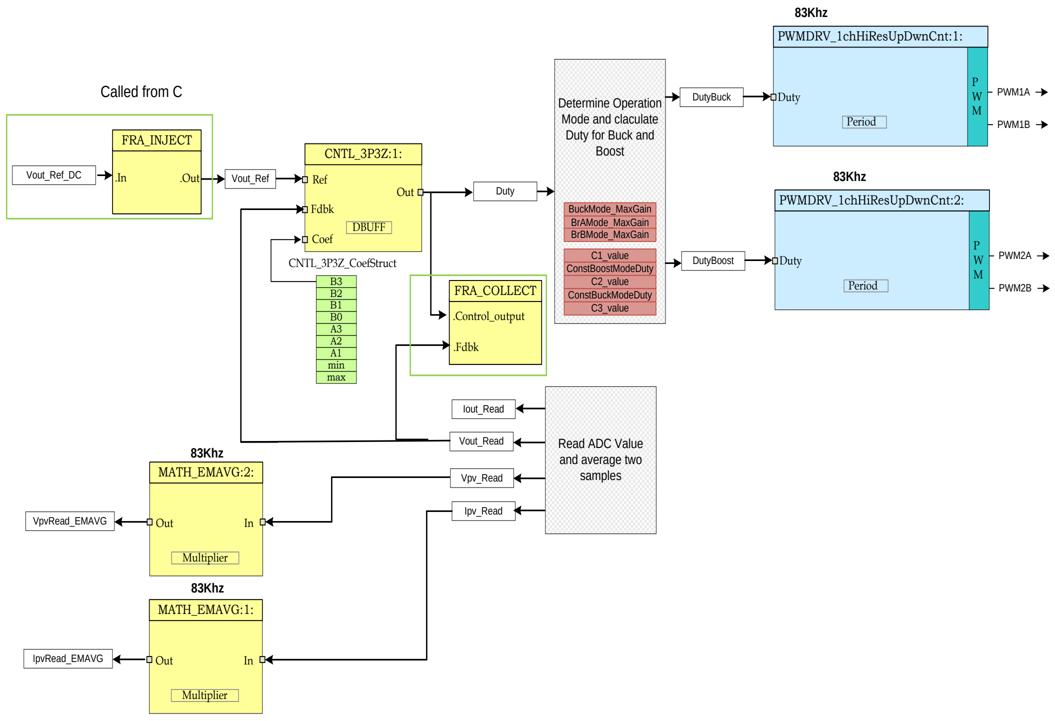


Fig 9 DC-DC output voltage control

As switching rate of the DC-DC stage is fairly high 250Khz, a section of the control ISR for the DC-DC is implemented in an optimized assembly function which is called from the C ISR. In the project the DC-DC ISR (DPL_ISR_wFRA()) is invoked every third switching cycle. Fig 9 gives the software diagram for the DC-DC stage using the optimized blocks from the digital power library and SFRA routine in C.

The ADC result registers are read and converted to normalized values and stored in variables I_{pvRead}, V_{pvread}, $I_{outread}$ and $V_{outread}$. Two 2-pole 2-zero controllers(CNTL_2P2Z) is used to close the DC-DC voltage loop. The voltage controller decides the amount of gain needed in the system. The output of the 2p2z is limited to 0-1. This is then multiplied by 2 and the Duty for the buck side switches and boost side switches is determined in an assembly routine according to the Table 1. The routine also imparts a refresh cycle on the high side drivers if need be. (Note as the control is executed every 3rd PWM cycle 3 refresh pulses will be generates, this is not the ideal situation however this saves any additional software overhead and is seen to work appropriately). The PWMDRV_1ch_UpDwnCnt, PWMDRV_1ch_UpDwnCntCompl block is used to drive the DC-DC stage buck and boost side switches (Note the original digital power library block does not configure the channel B to be switched complimentary of channel A, this is taken care of in the Main.c where the deadband for the PWM is configured.

Notice the color coding for the software blocks. The blocks in 'dark blue' represent the hardware modules on the C2000 controller. The blocks in 'blue' are the software drivers for these modules. Blocks in 'yellow' are the controller blocks for the control loop. Code snippet below shows the Input/Output connections between the different blocks used from the Digital Power Library to implement the DC-DC MPPT control software, this can directly be related to the control diagram above.

```
// Connect the PWM Driver
PWMDRV_1chHiResUpDwnCnt_Duty1 = &DutyBuck;
PWMDRV_1chHiResUpDwnCnt_Duty2 = &DutyBoost;

ADCDRV_4ch_RltPtrA=&Ipv_Read;
ADCDRV_4ch_RltPtrB=&Vpv_Read;
ADCDRV_4ch_RltPtrC=&Vout_Read;
ADCDRV_4ch_RltPtrD=&Iout_Read;


//2p2z connections for outpur voltage loop
CNTL_3P3Z_Ref1 = &Vout_Ref;
CNTL_3P3Z_Out1 = &Duty;
CNTL_3P3Z_Fdbk1= &Vout_Read;

CNTL_3P3Z_Coef1 = &CNTL_2P2Z_CoefStruct1.b2;
```

The Run time ISR calls the FRA functions and then DPL_Func(), which consists of just calling the run time macros from the digital power library.

```
interrupt void DPL_ISR_wFRA()
{
..
        Vout_Ref=SFRA_IQ_INJECT(Vout_Ref_DC);
        DPL_Func();
        SFRA_IQ_COLLECT(&Duty,&Vout_Read);
…
}
```

MPPT Mode (Solar MicroConverter Mode)

The Solar Panel or PhotoVoltaic (PV) panel, as it’s more commonly called, is a DC source with a non linear V vs I characteristics. The key challenges in PV system design are to extract maximum power from the panel by operating the panel at the maximum power point (MPP) of this non linear V vs I curve, and to convert the power such that it can be used to charge batteries, run DC loads, run AC loads, or feed power into the electrical grid. 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. An emerging system architect that supplements the String-inverter paradigm involves dc-dc converters (referred to here as microconverters) dedicated to individual PV modules, Fig 10 . The localized MPPT at each panel improves the performance of the system under partial shading and unmatched panels conditions.

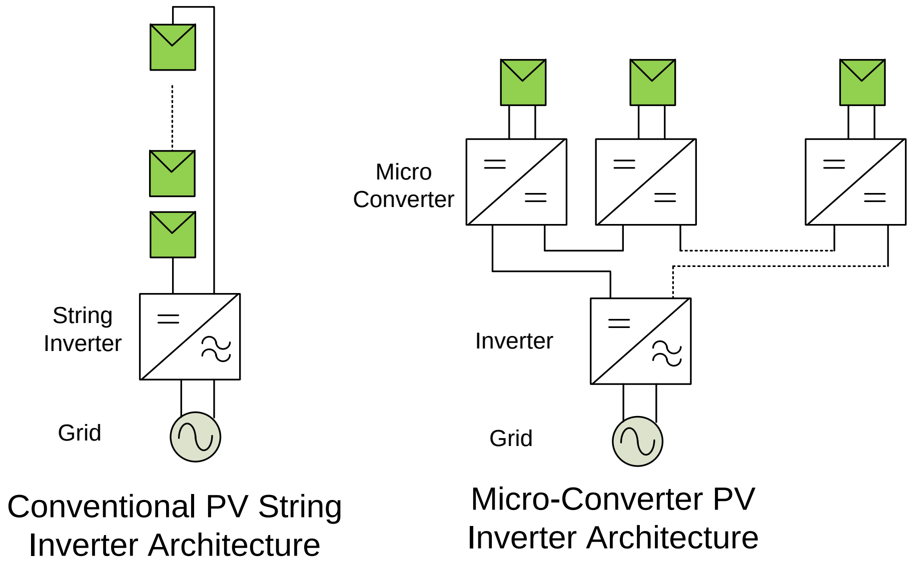

String Inverter
Grid
Micro Converter
Inverter
Grid
Conventional PV String Inverter Architecture
Micro-Converter PV Inverter Architecture

Fig 10 PV Inverter Architectures

To control the power stage in this mode control scheme shown in Fig 11 is used.

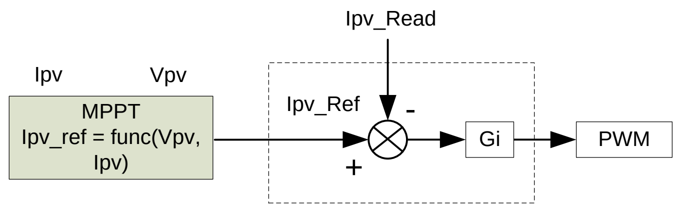


Fig 11 MPPT Control

Maximum power point however is not fixed due to the non linear nature of the PV cell and changes with temperature, light intensity etc. Thus different techniques are used to track maximum power point of the panel like Perturb and Observe, incremental conductance algorithms. These techniques try to track the maximum power point of the panel under given operating conditions and are thus referred to as Maximum Power Point Tracking (MPPT) techniques/algorithms. The kit has a DC-DC buck boost stage that can take input voltage from the solar panel and provide maximum power possible at the output,.

To track the MPP, input voltage (V_{pv}) and Input Current (I_{pv}) are sensed. The MPPT is realized using, a current loop that regulates input current (I_{pv}). Increasing the current reference increases the power drawn from the panel and the panel voltage drops. the current controller is executed at a rate of 83 kHz (1/3rd of the PWM switching frequency) while the MPPT controller is executed at a much slower rate ~ 10Hz. It is noted that the DC DC stage output voltage is not being controlled through software thus an appropriate load must be connected to the output such that the output voltage does not rise above its specified limits.

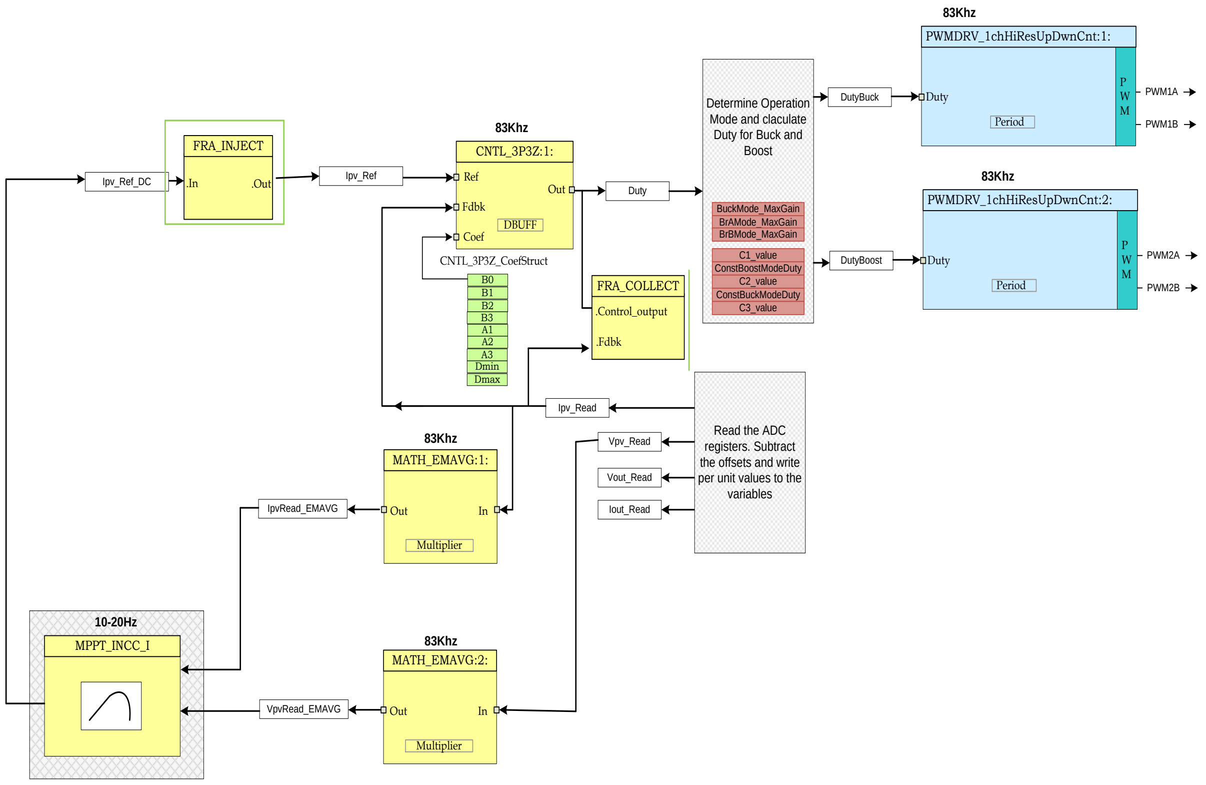


Fig 12 DC-DC with MPPT Software Diagram

As switching rate of the DC-DC stage is fairly high 250Khz, the control ISR for the DC-DC is implemented in an optimized assembly ISR (ASM – ISR) which uses components from the Digital Power Library. In the project the DC-DC ISR is invoked every third switching cycle, this is done because PV panel output does not change very fast. Fig 12 gives the software diagram for the DC-DC stage using the optimized blocks from the digital power library.

The ADC result registers are read and normalized values stored in variables I_{pvRead}, V_{pvread}, $I_{outread}$ and $V_{outread}$. A 2-pole 2-zero controllers(CNTL_2P2Z) is used to close the DC-DC current loop.. MPPT algorithm provides reference input current which is used as the reference for the controller. Current loop controller's output decides the amount of gain needed in the system. The output of the 2p2z is limited to 0-1. This is then multiplied by 2 and the Duty for the buck side switched and boost side switched is determined in an assembly routine according to the Table 1. The routine also imparts a refresh cycle on the high side drivers if need be. (Note as the control is executed every 3rd PWM cycle 3 refresh pulses will be generates, this is not the ideal situation however this saves any additional software overhead and is seen to work appropriately). The PWMDRV_1ch_UpDwnCnt, PWMDRV_1ch_UpDwnCntCompl block is used to drive the DC-DC stage buck and boost side switches (Note the original digital power library block does not configure the channel B to be switched complimentary of channel A, this is taken care of in the Main.c where the deadband for the PWM is configured. Panel current and voltage are filtered using the MATH_EMAVG block, this is done to remove any noise on the panel current and voltage sensing that may confuse the MPPT algorithm.

Notice the color coding for the software blocks. The blocks in ‘dark blue’ represent the hardware modules on the C2000 controller. The blocks in ‘blue’ are the software drivers for these modules. Blocks in ‘yellow’ are the controller blocks for the control loop. Although a 2-pole 2-zero controller is used here, the controller could very well be a PI/PID, a 3-pole 3-zero or any other controller that can be suitably implemented for this application. Similarly for MPP tracking, users can choose to use a different algorithm.

Code snippet below shows the Input/Output connections between the different blocks used from the Digital Power Library to implement the DC-DC MPPT control software, this can directly be related to the control diagram above.

```
// Connect the PWM Driver
PWMDRV_1chHiResUpDwnCnt_Duty1 = &DutyBuck;
PWMDRV_1chHiResUpDwnCnt_Duty2 = &DutyBoost;

ADCDRV_4ch_RltPtrA=&Ipv_Read;
ADCDRV_4ch_RltPtrB=&Vpv_Read;
ADCDRV_4ch_RltPtrC=&Vout_Read;
ADCDRV_4ch_RltPtrD=&Iout_Read;

//3p3z connections for current loop
CNTL_3P3Z_Ref1 = &Ipv_Ref;
CNTL_3P3Z_Out1 = &DutyZero;
CNTL_3P3Z_Fdbk1= &Ipv_Read;
CNTL_3P3Z_Coef1 = &CNTL_2P2Z_CoefStruct1.b2;


// MATH_EMAVG1 block connections
MATH_EMAVG_In1=&Ipv_Read;
MATH_EMAVG_Out1=&Ipv_Read_EMAVG;
MATH_EMAVG_Multiplier1=_IQ30(0.001);

// MATH_EMAVG2 block connections
MATH_EMAVG_In2=&Vpv_Read;
MATH_EMAVG_Out2=&Vpv_Read_EMAVG;
MATH_EMAVG_Multiplier2=_IQ30(0.001);
```

The run time ISR calls the FRA functions and then DPL_Func(), which consists of just calling the run time macros from the digital power library. The MPPT algorithm is called from a background task in the background C framework.

Reverse Power Flow

The DC-DC buck boost stage can be used in bi-directional mode by simple switch in the software. Control scheme used is illustrated in Fig 13 . To demonstrate the reverse flow the input voltage is controlled.

Note: Typically in bi directional application current control scheme is used. This mode is just to illustrate the hardware capability in reversing the power flow.

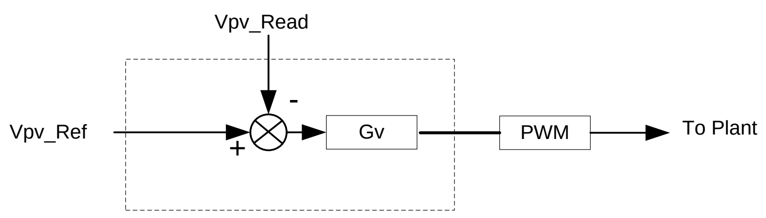


Fig 13 Reverse Power Flow

.
The control of the stage is described in Fig 13. To enable the reverse flow change the DIRECTION define in the "Settings.h" file.

```
#define DIRECTION 2     // 1 for FORWARD
                        // 2 for Backward
```

Notice that with the reverse direction the PWM switches are swapped. The piccolo PWM module is very versatile and offers the option to swap the high side and low side switches when operating in buck boost mode.

```
#elif (DIRECTION==2)
        PWMDRV_1chHiResUpDwnCnt_Duty1 = &DutyBoost;
        PWMDRV_1chHiResUpDwnCnt_Duty2 = &DutyBuck;
        EALLOW;
        EPwm1Regs.HRCNFG.bit.SWAPAB=1;
        EPwm2Regs.HRCNFG.bit.SWAPAB=1;
        EDIS;
#endif
```

Additionally the refresh pulse also needs to be applied in the opposite way i.e. refresh on PWM1 when buck mode and refresh on PWM2 when in boost mode. This is also taken care of in the 1Khz ISR routine.

The control loop net terminals are also changed as follows:

```
#elif(DIRECTION==2)
        CNTL_2P2Z_Ref1 = &Vpv_Ref;

        #if (OPEN_LOOP == 1)
        CNTL_2P2Z_Out1 = &DutyZero;
        #else
        CNTL_2P2Z_Out1 = &Duty;
        #endif

        CNTL_2P2Z_Fdbk1= &Vpv_Read;
        CNTL_2P2Z_Coef1 = &CNTL_2P2Z_CoefStruct1.b2;
#endif
```

To output voltage (panel input voltage in this case) is sensed, (V_{pv}) and compared with the reference value (V_{pv_ref}) and the 2p2z compensator calculated the effort needed. The voltage controller is executed at a rate of 83 kHz (1/3rd of the PWM switching frequency).

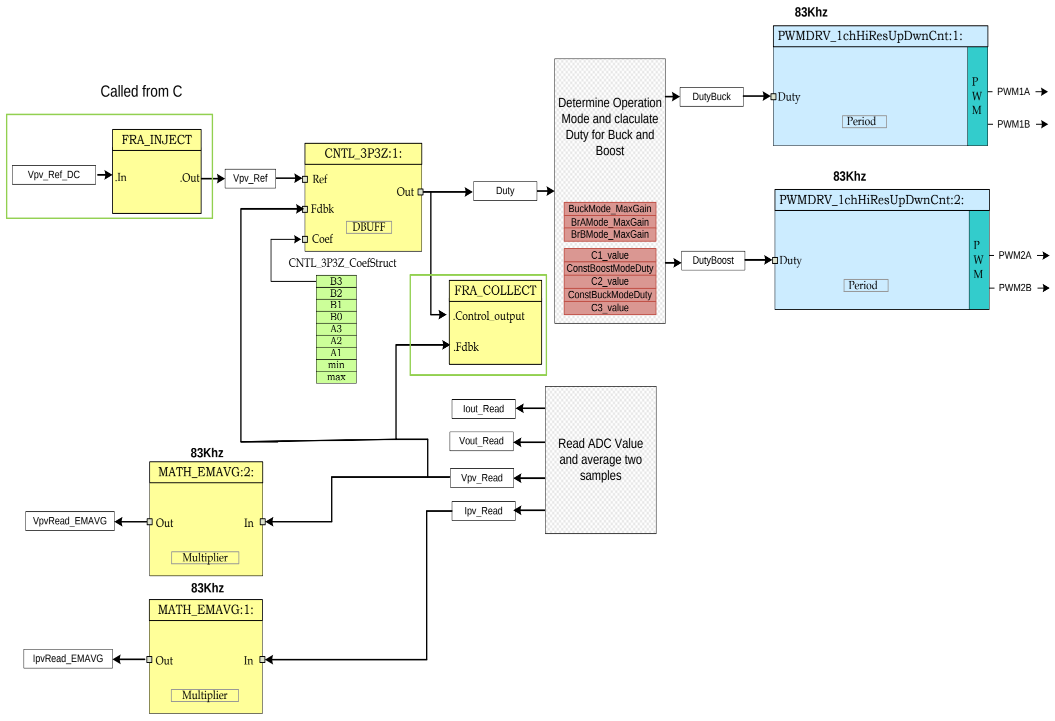


Fig 14 DC-DC output voltage control in reverse direction

5. Running the Software using CCS

5.1 HW Setup Instructions

Note: Do not power up the board before you have verified these settings!
Before starting the labs the user must make sure the following settings are correct.

1) Make sure nothing is connected to the board, and no power is being supplied to the board.

2) Insert the controlCARD into the controlCARD connector if it is not already installed (F28035 ISO DIMM CARD).

3) Do the following switch settings on the controlCARD:
    a. Control Card SW3 is in the ON position (JTAG Connection)
    b. Control Card SW2, Position 1 = ON, Position 2 = ON

4) Connect a DC Load between VOPWR and VORTN (user must make sure the output load is of the appropriate value needed for the test)

5) Connect a USB cable (mini to A Cable) from control card to the host computer. LD4 on the control card will light up indication USB connection.

6) Make sure the following jumers are connected/disconnected:
    a. LK201: Connected (enables on-board bias power)
        i. If desired to supply 5V and 12V externally, then leave this jumper open, and supply 5V and 12V externally to their test points.
    b. LOOP STIM: leave jumper open.
    c. LK202, LK203: Connect jumpers horizontally at 5v and 12V labels.
        i. If 5V and 12V are supplied externally, leave these jumpers open.
    d. LK301, LK302: Connect jumpers vertically for normal configuration.

7) For verifying different mode of operation slightly different setup is needed
    a. Output Voltage Control

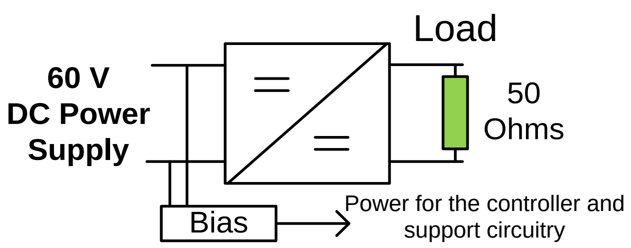


Fig 15 Output Voltage Control Setup

    b. MPPT Mode

Connect a solar panel / panel emulator between PVPWR and PVRTN. If the solar panel is producing enough energy the bias supply will power up the controller and you will see the green LED, LD1 on the control card light up indication power. The bias supply will kick in if the input voltage exceeds ~9V.

Alternatively a current limited DC supply can be used but the user must be careful when operating in the MPPT mode, a resistor must be connected in series to create a non linear V vs I curve, Fig 16. For example a Voltage of 30V and a series resistance of 15 Ohms will provide the curve shown in Fig 17.

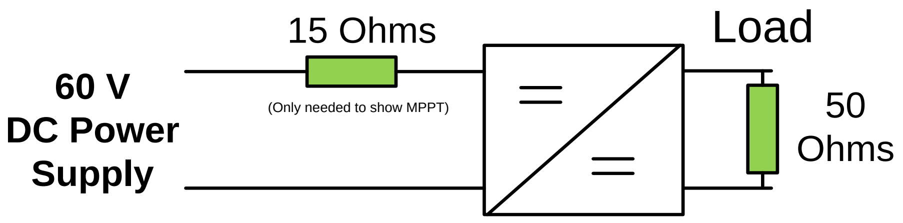


Fig 16 PV Emulator using a DC power supply for testing MPPT Mode

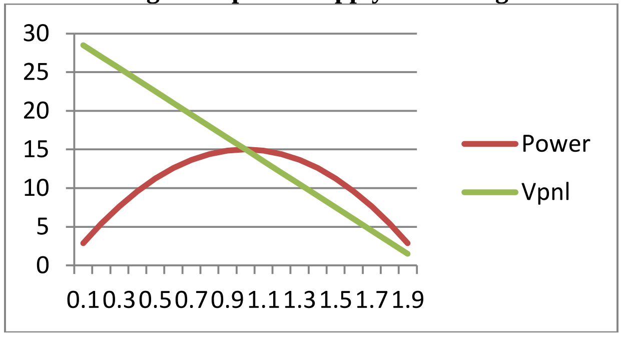


Fig 17 PV Curve using DC power supply, 30V and 15 Ohm Resistance

c. Reverse Power Flow

For seeing power flow in the opposite direction the connections are shown in the figure below. An external 9V supply is needed to connect to the bias power supply input for this mode.

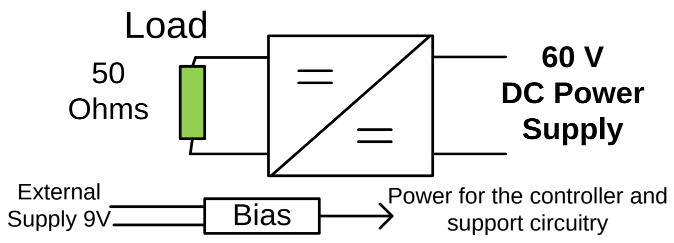


Fig 18 Setup to test reverse power flow

5.2 Software Setup

Installing Code Composer and controlSUITE

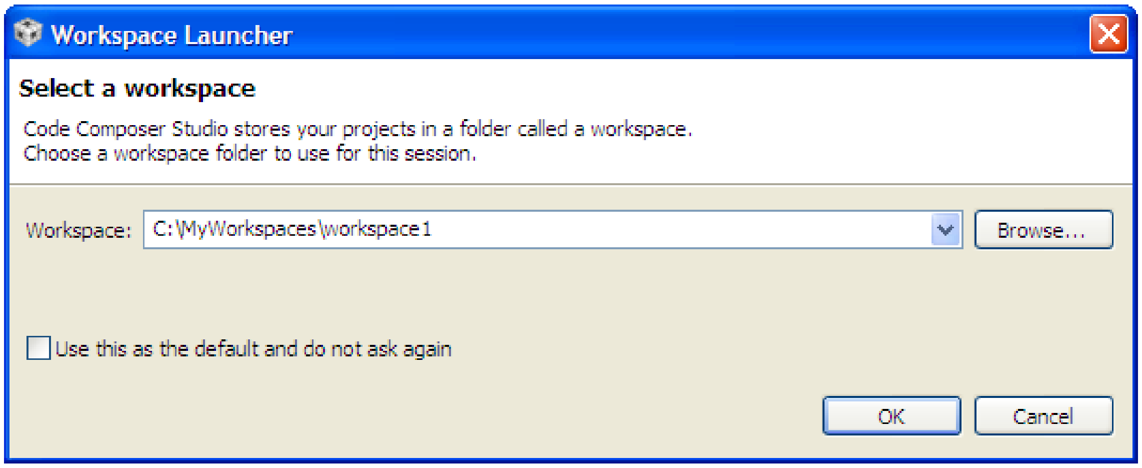
1. If not already installed, please install Code Composer v6 from http://processors.wiki.ti.com/index.php/Category:Code_Composer_Studio_v6

2. Go to http://www.ti.com/controlsuite and run the controlSUITE installer. Select to install the "Bidirectional Buck Boos TI Design".

Setup Code Composer Studio to Work with kit

3. Open "Code Composer Studio v6".

4. Once Code Composer Studio opens, the workspace launcher may appear that would ask to select a workspace location,: (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:\MyWorkspaces\workspace1"
   - Uncheck the box that says "Use this as the default and do not ask again".
   - Click "OK".



Workspace Launcher

Select a workspace

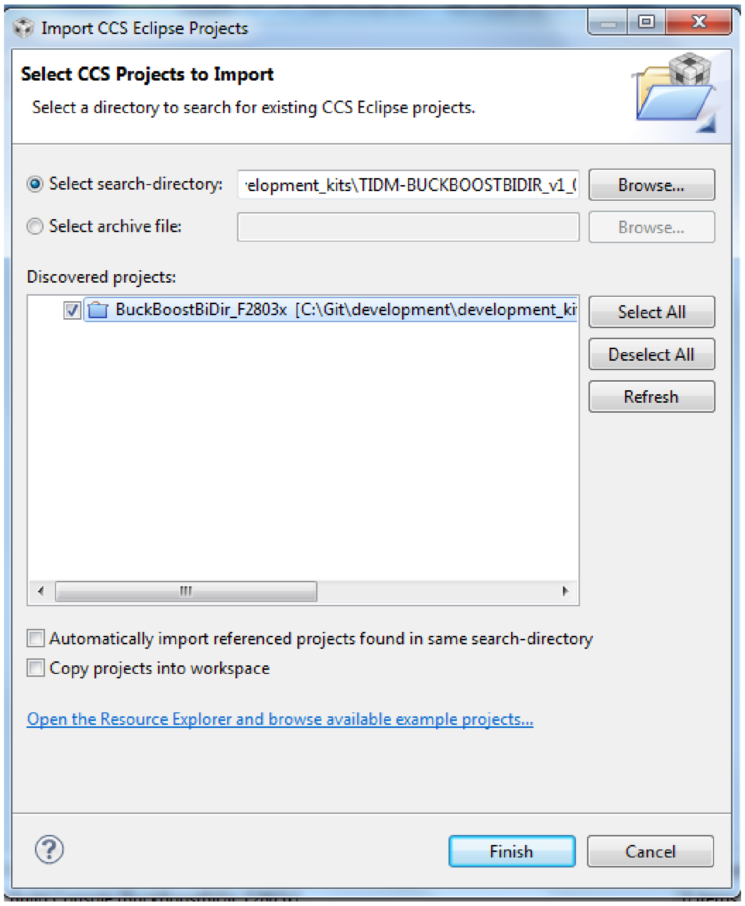
Code Composer Studio stores your projects in a folder called a workspace.
Choose a workspace folder to use for this session.

Workspace: C:\MyWorkspaces\workspace1

Fig 19 Open new workspace

5. Add the project into your current workspace by clicking "Project->Import Existing CCS/CCE Eclipse Project".

   - Select the root directory: "\controlSUITE\development_kits\TIDM_BUCKBOOST_BIDIR \vX \BuckBoostBiDir_F2803x\"


Import CCS Eclipse Projects

Select CCS Projects to Import

Select a directory to search for existing CCS Eclipse projects.

(●) Select search-directory: 'elopment_kits\TIDM-BUCKBOOSTBIDIR_v1_(

( ) Select archive file:

Discovered projects:

☑ BuckBoostBiDir_F2803x [C:\Git\development\development_ki

☐ Automatically import referenced projects found in same search-directory

☐ Copy projects into workspace

Open the Resource Explorer and browse available example projects...

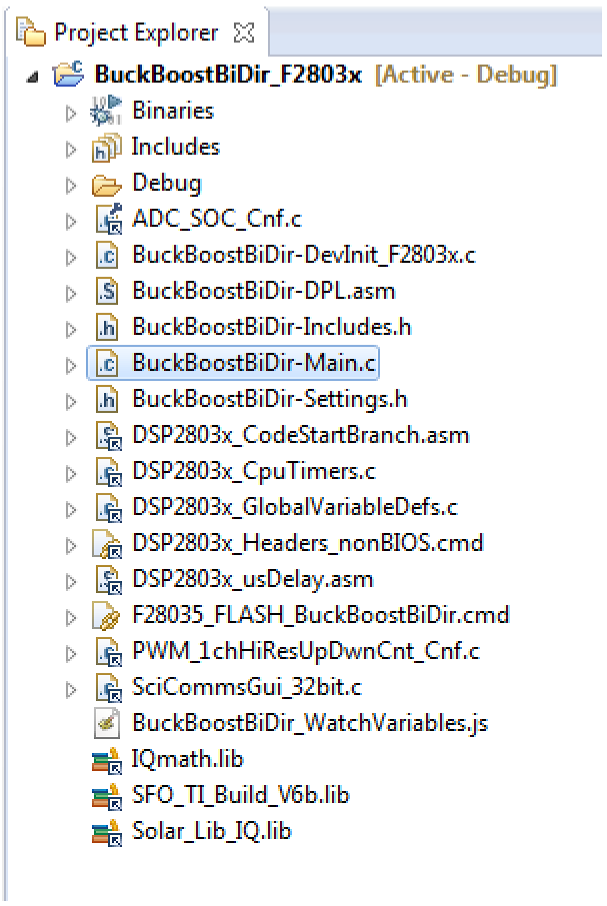
Fig 20 Adding F28035 project to the workspace

- 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

6. 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.

7. Fig 21 shows the project in the CCSv4 C/C++ Project tab, it shows all the key files used in the project.



Project Explorer

BuckBoostBiDir_F2803x [Active - Debug]
- Binaries
- Includes
- Debug
- ADC_SOC_Cnf.c
- BuckBoostBiDir-DevInit_F2803x.c
- BuckBoostBiDir-DPL.asm
- BuckBoostBiDir-Includes.h
- BuckBoostBiDir-Main.c
- BuckBoostBiDir-Settings.h
- DSP2803x_CodeStartBranch.asm
- DSP2803x_CpuTimers.c
- DSP2803x_GlobalVariableDefs.c
- DSP2803x_Headers_nonBIOS.cmd
- DSP2803x_usDelay.asm
- F28035_FLASH_BuckBoostBiDir.cmd
- PWM_1chHiResUpDwnCnt_Cnf.c
- SciCommsGui_32bit.c
- BuckBoostBiDir_WatchVariables.js
- IQmath.lib
- SFO_TI_Build_V6b.lib
- Solar_Lib_IQ.lib

Fig 21 Project Explorer Tab

5.3 Procedure

1. Open and inspect *BuckBoostBiDir-DevInit_F2803x.c* by double clicking on the filename in the project window. Note that system clock, peripheral clock prescale, and peripheral clock enables have been setup. Next, notice how the shared GPIO pins have been configured.
2. Open and inspect *BuckBoostBiDir-Main.c*. Notice the call made to *DeviceInit()* function and other variable initialization.
3. Locate and inspect the code in the main file under initialization code. Observe functions used for EPWM module initialization (PWM_1ch_UpDwnCnt_CNF, PWM_1ch_UpDwnCntCompl_CNF) and ADC module initialization (*ADC_SOC_CNF)* blocks. This is common for all incremental builds. Note the Digital Power Library functions do not configure the deadband module thus this is done in the main file.
4. Also locate and inspect the following code in the main file under initialization code. This is where the ADC channels for different feedback signals are assigned and the start-of-conversion triggers are programmed.

```
#define Ipv_FB                          AdcResult.ADCRESULT1
#define Vpv_FB                          AdcResult.ADCRESULT2
#define Vout_FB                         AdcResult.ADCRESULT4
#define Iout_FB                         AdcResult.ADCRESULT5


//Map channel to ADC Pin
// the dummy reads are to account for first sample issue in Rev 0 silicon. Please refer to the Errata
and the datasheet, this would be fixed in later versions of the silicon
 ChSel[0] = 2;                 // A2 - Ipv-FB Dummy,
 ChSel[1] = 2;                 // A2 - Ipv-FB,
 ChSel[2] = 10;    // B2 - Vpv-FB,
 ChSel[3] = 12;    // B4 - Vout_FB,Dummy,
 ChSel[4] = 12;    // B4 - Vout_FB,
 ChSel[5] = 4;      // A4 - Iout_FB,

// Select Trigger Event
TrigSel[0]= ADCTRIG_EPWM1_SOCA;
TrigSel[1]= ADCTRIG_EPWM1_SOCA;
TrigSel[2]= ADCTRIG_EPWM1_SOCA;
TrigSel[3]= ADCTRIG_EPWM2_SOCA;
TrigSel[4]= ADCTRIG_EPWM2_SOCA;
TrigSel[5]= ADCTRIG_EPWM2_SOCA;

ADC_SOC_CNF(ChSel,TrigSel,ACQPS,-1,3); //ADC auto clear mode, no interrupts
```

Build and Load the Project

5. Now hit the debug button, this will load the project into the controller. The perspective will now change to "Debug". Once program load is complete the debugger will halt the code at the beginning of the Main() routine

6. Click View->Scripting Console to open the scripting console and open the "WatchVariables.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

Real-time emulation is a special emulation feature that allows the windows within Code Composer Studio to be updated at a rate up to 10 Hz *while the MCU is running*. This not only 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. This is very useful when tuning control law parameters on-the-fly, for example.

7. Enable real-time mode by hovering your mouse on the buttons on the horizontal toolbar and clicking button Enable Silicon Real-time Mode (service critical interrupts when halted, allow debugger accesses while running) .

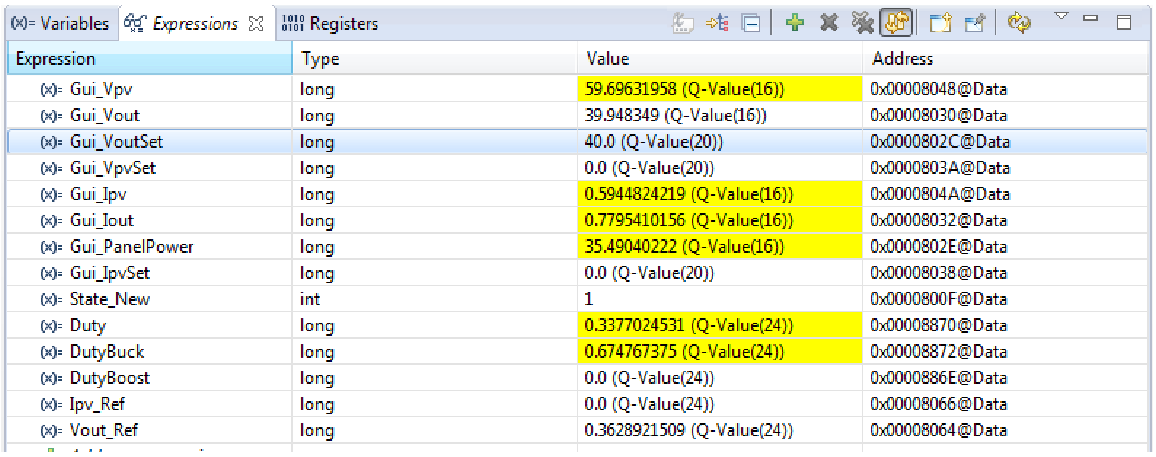
8. A message box *may* appear. If so, select YES to enable debug events. This will set bit 1 (DGBM bit) of status register 1 (ST1) to a "0". The DGBM is the debug enable mask bit. When the DGBM bit is set to "0", memory and register values can be passed to the host processor for updating the debugger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

9. Click on Continuous Refresh buttons for the watch view.

10. Run the project,

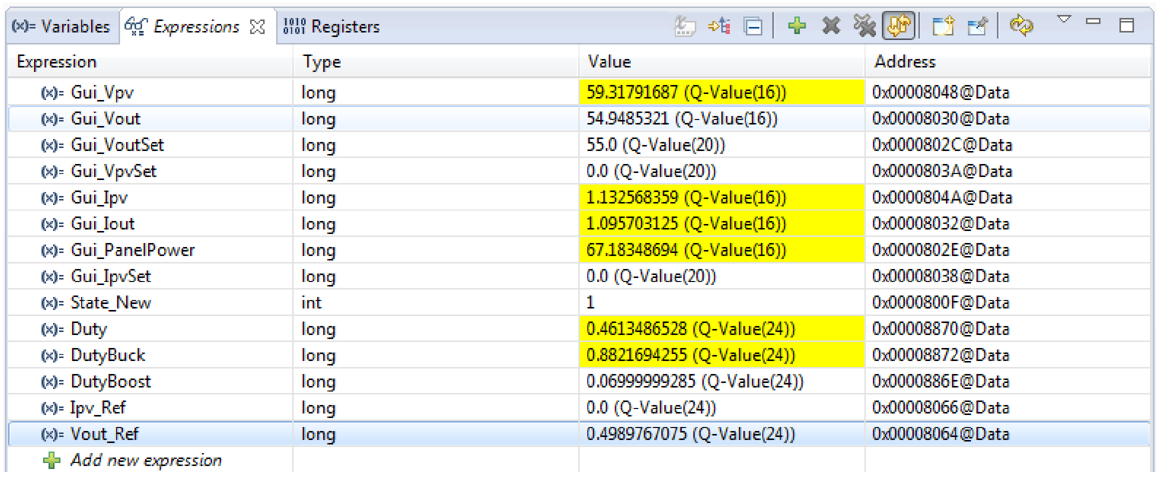
Check Output Voltage Control

11. In the watch view, check the value of Gui_Vpv this will be the voltage at the input of the board. The board will show some power drawn because of the bias power supply and offsets on the ADC signals. The output voltage of the stage will be zero. By default the mode of operation in Output Voltage Control(State_New=1). Set a new voltage by entering a value in Gui_Vout, for example 9V. The voltage will be regulated at the set point voltage and because input voltage is at ~60V slowly increasing the set point changes the mode of operation of the converter from buck to br-A to br-B and boost, this can be observed through the watch window



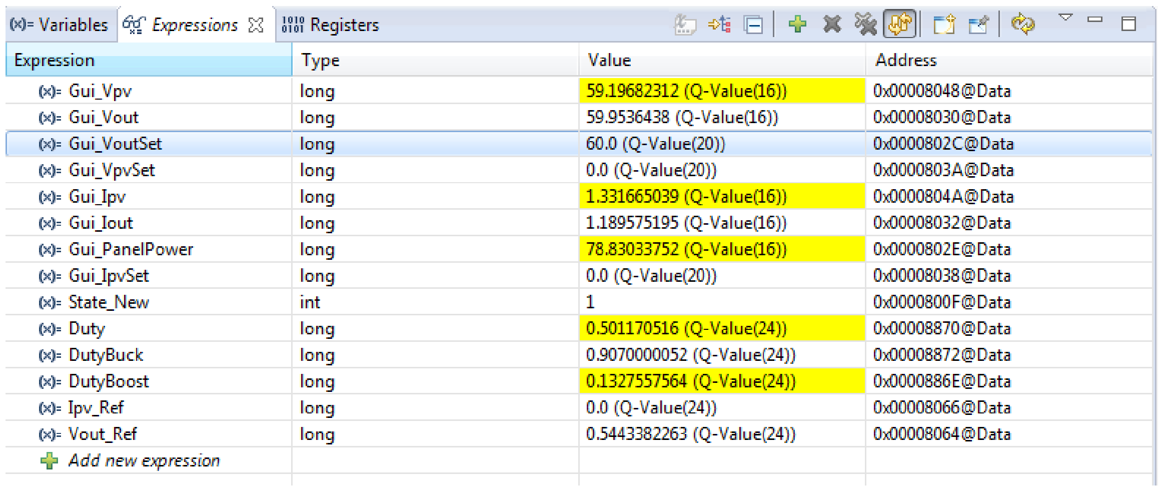
Expression	Type	Value	Address
Gui_Vpv	long	59.69631958 (Q-Value(16))	0x00008048@Data
Gui_Vout	long	39.948349 (Q-Value(16))	0x00008030@Data
Gui_VoutSet	long	40.0 (Q-Value(20))	0x0000802C@Data
Gui_VpvSet	long	0.0 (Q-Value(20))	0x0000803A@Data
Gui_Ipv	long	0.5944824219 (Q-Value(16))	0x0000804A@Data
Gui_Iout	long	0.7795410156 (Q-Value(16))	0x00008032@Data
Gui_PanelPower	long	35.49040222 (Q-Value(16))	0x0000802E@Data
Gui_IpvSet	long	0.0 (Q-Value(20))	0x00008038@Data
State_New	int	1	0x0000800F@Data
Duty	long	0.3377024531 (Q-Value(24))	0x00008870@Data
DutyBuck	long	0.674767375 (Q-Value(24))	0x00008872@Data
DutyBoost	long	0.0 (Q-Value(24))	0x0000886E@Data
Ipv_Ref	long	0.0 (Q-Value(24))	0x00008066@Data
Vout_Ref	long	0.3628921509 (Q-Value(24))	0x00008064@Data

Buck Mode



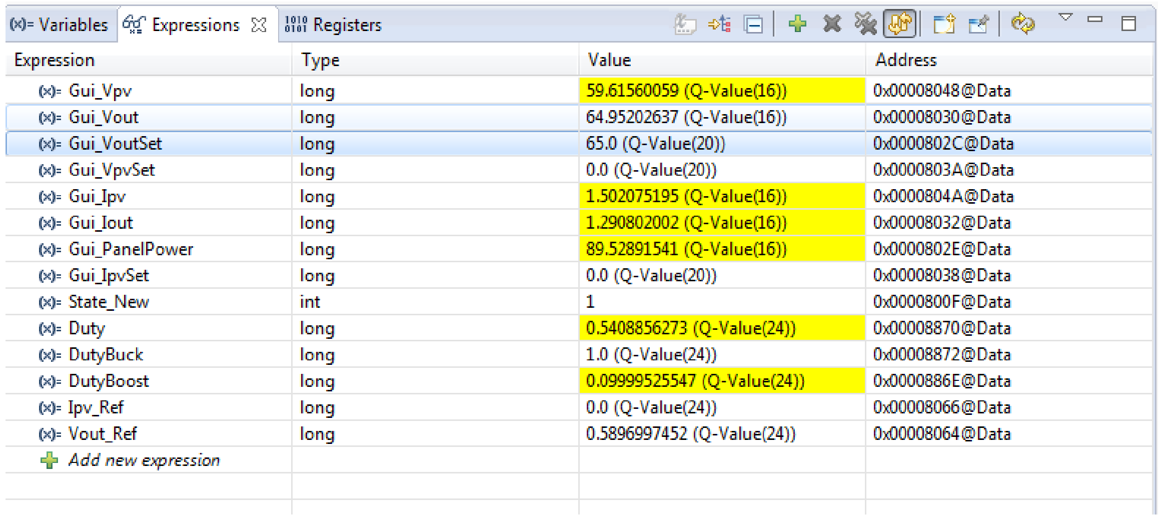
Expression	Type	Value	Address
(x)= Gui_Vpv	long	59.31791687 (Q-Value(16))	0x00008048@Data
(x)= Gui_Vout	long	54.9485321 (Q-Value(16))	0x00008030@Data
(x)= Gui_VoutSet	long	55.0 (Q-Value(20))	0x0000802C@Data
(x)= Gui_VpvSet	long	0.0 (Q-Value(20))	0x0000803A@Data
(x)= Gui_Ipv	long	1.132568359 (Q-Value(16))	0x0000804A@Data
(x)= Gui_Iout	long	1.095703125 (Q-Value(16))	0x00008032@Data
(x)= Gui_PanelPower	long	67.18348694 (Q-Value(16))	0x0000802E@Data
(x)= Gui_IpvSet	long	0.0 (Q-Value(20))	0x00008038@Data
(x)= State_New	int	1	0x0000800F@Data
(x)= Duty	long	0.4613486528 (Q-Value(24))	0x00008870@Data
(x)= DutyBuck	long	0.8821694255 (Q-Value(24))	0x00008872@Data
(x)= DutyBoost	long	0.06999999285 (Q-Value(24))	0x0000886E@Data
(x)= Ipv_Ref	long	0.0 (Q-Value(24))	0x00008066@Data
(x)= Vout_Ref	long	0.4989767075 (Q-Value(24))	0x00008064@Data
Add new expression			

Constant Boost + Buck Mode



Expression	Type	Value	Address
(x)= Gui_Vpv	long	59.19682312 (Q-Value(16))	0x00008048@Data
(x)= Gui_Vout	long	59.9536438 (Q-Value(16))	0x00008030@Data
(x)= Gui_VoutSet	long	60.0 (Q-Value(20))	0x0000802C@Data
(x)= Gui_VpvSet	long	0.0 (Q-Value(20))	0x0000803A@Data
(x)= Gui_Ipv	long	1.331665039 (Q-Value(16))	0x0000804A@Data
(x)= Gui_Iout	long	1.189575195 (Q-Value(16))	0x00008032@Data
(x)= Gui_PanelPower	long	78.83033752 (Q-Value(16))	0x0000802E@Data
(x)= Gui_IpvSet	long	0.0 (Q-Value(20))	0x00008038@Data
(x)= State_New	int	1	0x0000800F@Data
(x)= Duty	long	0.501170516 (Q-Value(24))	0x00008870@Data
(x)= DutyBuck	long	0.9070000052 (Q-Value(24))	0x00008872@Data
(x)= DutyBoost	long	0.1327557564 (Q-Value(24))	0x0000886E@Data
(x)= Ipv_Ref	long	0.0 (Q-Value(24))	0x00008066@Data
(x)= Vout_Ref	long	0.5443382263 (Q-Value(24))	0x00008064@Data
Add new expression			

Constant Buck + Boost Mode



Expression	Type	Value	Address
(x)= Gui_Vpv	long	59.61560059 (Q-Value(16))	0x00008048@Data
(x)= Gui_Vout	long	64.95202637 (Q-Value(16))	0x00008030@Data
(x)= Gui_VoutSet	long	65.0 (Q-Value(20))	0x0000802C@Data
(x)= Gui_VpvSet	long	0.0 (Q-Value(20))	0x0000803A@Data
(x)= Gui_Ipv	long	1.502075195 (Q-Value(16))	0x0000804A@Data
(x)= Gui_Iout	long	1.290802002 (Q-Value(16))	0x00008032@Data
(x)= Gui_PanelPower	long	89.52891541 (Q-Value(16))	0x0000802E@Data
(x)= Gui_IpvSet	long	0.0 (Q-Value(20))	0x00008038@Data
(x)= State_New	int	1	0x0000800F@Data
(x)= Duty	long	0.5408856273 (Q-Value(24))	0x00008870@Data
(x)= DutyBuck	long	1.0 (Q-Value(24))	0x00008872@Data
(x)= DutyBoost	long	0.09999525547 (Q-Value(24))	0x0000886E@Data
(x)= Ipv_Ref	long	0.0 (Q-Value(24))	0x00008066@Data
(x)= Vout_Ref	long	0.5896997452 (Q-Value(24))	0x00008064@Data
Add new expression			

Boost Mode

Fig 22 Snap shot of CCS Watch Window for Build 1using forward Power Flow at different operation modes Buck, Br-A, Br-B and Boost

12. As a new voltage is set you will see the Gui_Vout rise to the set point. You can vary the output voltage set point and see the buck, boost or bridge mode operation. Now set Gui_Vout equal to zero.

13. The software is configured to connect to the SFRA GUI. Open the SFRA gui.exe, located at controlSUITE/libs/app_libs/SFRA/version/GUI

14. Select fixed point math.

15. Click Setup Connection and set the baud rate to be 57600 on the pop up window.

16. Uncheck boot on connect and select the appropriate COM port.If this is the first time you are connecting that control card to the computer it will be the highest number.

17. Click "OK" to close the pop-up window and return to the main screen.

18. On Main Window click "Connect". Once connected the GUI will parse the current settings for the FRA sweep from the controller, these include the Start Frequency of the sweep, the length of the frequency sweep array (this is fixed in the code and hence cannot be changed through the GUI), injection amplitude and steps per decade. Leave these as default for now.

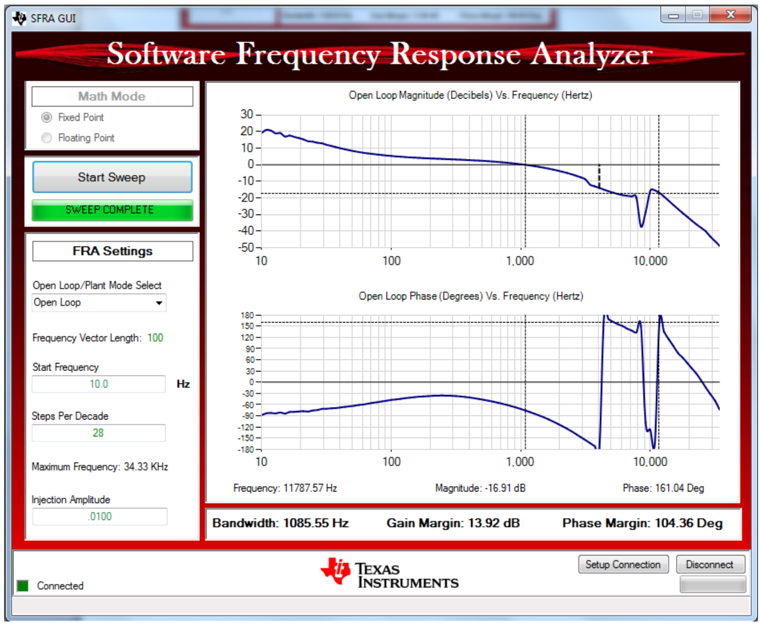
19. Press the "Start Sweep" button.

20. Wait for the status bar in the GUI to change to "Sweep Complete"

21. The results of the SFRA Sweep are now displayed on the window. One can check the SFRA result at different operating point of the board.

    Note: the code uses a fixed set of coefficients, a bank of coefficients can be used to maintain better margin across the load and operating range.

Vin = 60V , Output Load=50Ohms, Vout =40V



Open Loop/Plant Mode Select: Open Loop

Frequency Vector Length: 100

Start Frequency: 10.0 Hz

Steps Per Decade: 28

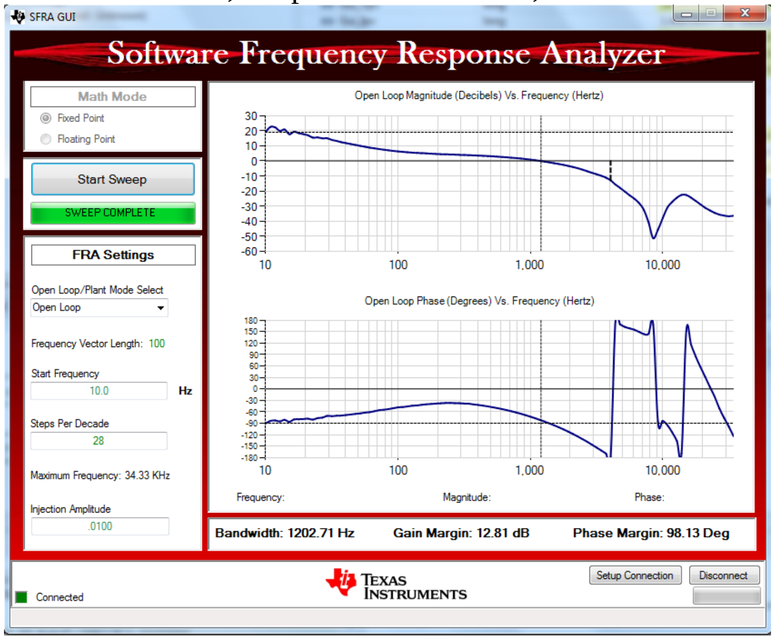
Maximum Frequency: 34.33 KHz

Injection Amplitude: .0100

Frequency: 11787.57 Hz Magnitude: -16.91 dB Phase: 161.04 Deg

Bandwidth: 1085.55 Hz Gain Margin: 13.92 dB Phase Margin: 104.36 Deg

Vin = 60V , Output Load=50Ohms, Vout =60V



Open Loop/Plant Mode Select: Open Loop

Frequency Vector Length: 100

Start Frequency: 10.0 Hz

Steps Per Decade: 28

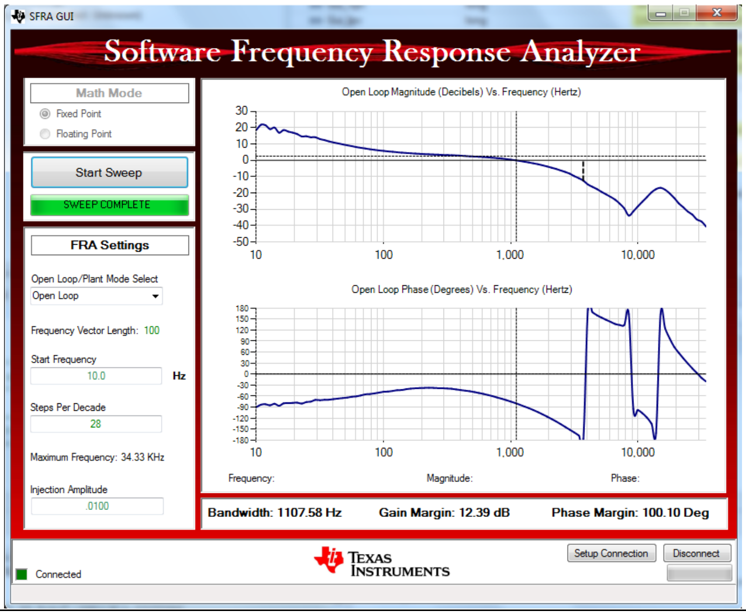
Maximum Frequency: 34.33 KHz

Injection Amplitude: .0100

Frequency: Magnitude: Phase:

Bandwidth: 1202.71 Hz Gain Margin: 12.81 dB Phase Margin: 98.13 Deg

Vin = 60V , Output Load=50Ohms, Vout =65V



Math Mode
Fixed Point
Floating Point

SWEEP COMPLETE

FRA Settings

Open Loop/Plant Mode Select: Open Loop

Frequency Vector Length: 100

Start Frequency: 10.0 Hz

Steps Per Decade: 28

Maximum Frequency: 34.33 KHz

Injection Amplitude: .0100

Frequency:

Magnitude:

Phase:

Bandwidth: 1107.58 Hz Gain Margin: 12.39 dB Phase Margin: 100.10 Deg

Connected

Check MPPT Mode

22. To check MPPT mode make sure the input is connected to a panel , a panel emulator or to a power supply with a series connected resistor to emulate the PV characteristics. To enable MPPT mode just put State_new = 2, make sure that if power supply is used for the input stage it is current limited, the supply will reach this max current limit as soon as MPPT state is enabled. In case of PV the panel the MPP will be reached.

23. To end the experiment write a 0 to Gui_Vout, now set the State_new =1. Now wait till the Gui_Vout is equal to zero. 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 .

Check Reverse Power Flow

1. Note the bias supply is only from one direction i.e. the PV side, hence to see the reverse direction of power flow we will need to disconnect the bias from the PV side i.e. depopulate the jumper LK201, and connect a controller power source of 9V DC between PWR pin and the Bias Ground to externally supply the controller power.

2. Also the software uses Vpv as the term for the input terminal in the forward direction, however in reality the Vpv should never sink in current!! This is just used for software compatibility of the board that is designed for solar application.

3. Now to begin reverse power test, disconnect everything from the board and make sure there is no energized source.

4. Now remove the LK201 jumper and connect a power source DC 10V between PWM pin of this jumper and the Control Ground Test point, this will now provide the controller power and you will see the controller power light come up.

5. Now connect a resistive load of 30 Ohms on the Panel Input side and Connect a controller DC source on the output side.

6. Go to the (ProjectName}-settings.h file and make the changes for DIRECTION==2. Also make sure the OPEN_LOOP==0

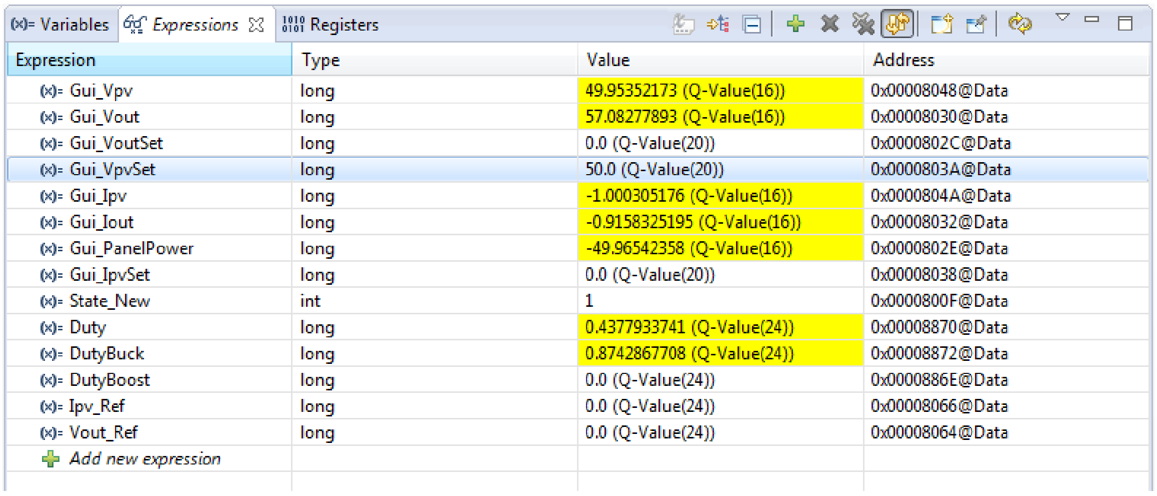
7. Rebuild the code and load the code.

8. Now once the code is loaded enable real time mode and run the code.

9. Enter Gui_VpvSet in _IQ20 format in the watch window, this should be 0.0 intially.

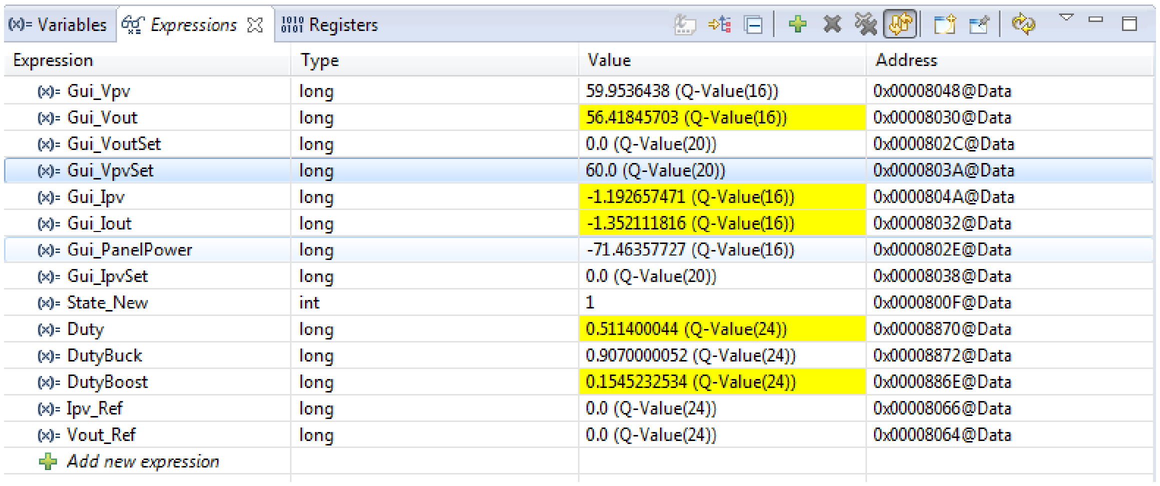
10. Now slowly raise the input voltage (connected at the output terminal), see the Gui_Vout matches this value.

11. Now set the Gui_VpvSet to _IQ20(10.0), and see the Gui_Vpv value regulate at that voltage. As a 60V supply is connected to the terminal hence varying the Gui_VpvSet point will take the power supply across Buck, Br-A, Br-B and Boost modes. This can be seen through the watch window. Also note the negative value of current and power illustrating reverse power flow.



Expression	Type	Value	Address
Gui_Vpv	long	49.95352173 (Q-Value(16))	0x00008048@Data
Gui_Vout	long	57.08277893 (Q-Value(16))	0x00008030@Data
Gui_VoutSet	long	0.0 (Q-Value(20))	0x0000802C@Data
Gui_VpvSet	long	50.0 (Q-Value(20))	0x0000803A@Data
Gui_Ipv	long	-1.000305176 (Q-Value(16))	0x0000804A@Data
Gui_Iout	long	-0.9158325195 (Q-Value(16))	0x00008032@Data
Gui_PanelPower	long	-49.96542358 (Q-Value(16))	0x0000802E@Data
Gui_IpvSet	long	0.0 (Q-Value(20))	0x00008038@Data
State_New	int	1	0x0000800F@Data
Duty	long	0.4377933741 (Q-Value(24))	0x00008870@Data
DutyBuck	long	0.8742867708 (Q-Value(24))	0x00008872@Data
DutyBoost	long	0.0 (Q-Value(24))	0x0000886E@Data
Ipv_Ref	long	0.0 (Q-Value(24))	0x00008066@Data
Vout_Ref	long	0.0 (Q-Value(24))	0x00008064@Data
Add new expression			

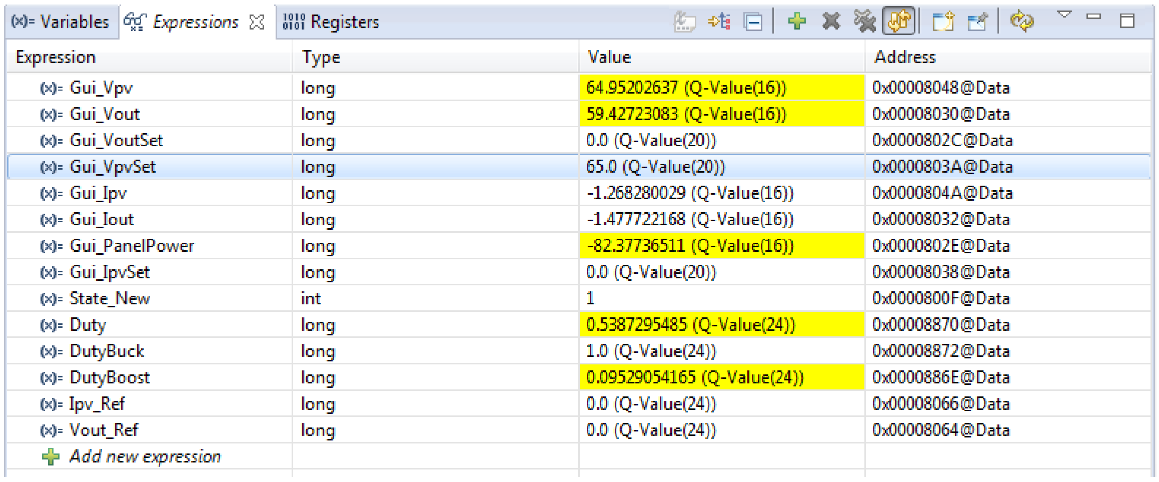
Buck



Variables | Expressions | Registers

Expression	Type	Value	Address
Gui_Vpv	long	59.9536438 (Q-Value(16))	0x00008048@Data
Gui_Vout	long	56.41845703 (Q-Value(16))	0x00008030@Data
Gui_VoutSet	long	0.0 (Q-Value(20))	0x0000802C@Data
Gui_VpvSet	long	60.0 (Q-Value(20))	0x0000803A@Data
Gui_Ipv	long	-1.192657471 (Q-Value(16))	0x0000804A@Data
Gui_Iout	long	-1.352111816 (Q-Value(16))	0x00008032@Data
Gui_PanelPower	long	-71.46357727 (Q-Value(16))	0x0000802E@Data
Gui_IpvSet	long	0.0 (Q-Value(20))	0x00008038@Data
State_New	int	1	0x0000800F@Data
Duty	long	0.511400044 (Q-Value(24))	0x00008870@Data
DutyBuck	long	0.9070000052 (Q-Value(24))	0x00008872@Data
DutyBoost	long	0.1545232534 (Q-Value(24))	0x0000886E@Data
Ipv_Ref	long	0.0 (Q-Value(24))	0x00008066@Data
Vout_Ref	long	0.0 (Q-Value(24))	0x00008064@Data
Add new expression			

Constant Buck



Variables | Expressions | Registers

Expression	Type	Value	Address
Gui_Vpv	long	64.95202637 (Q-Value(16))	0x00008048@Data
Gui_Vout	long	59.42723083 (Q-Value(16))	0x00008030@Data
Gui_VoutSet	long	0.0 (Q-Value(20))	0x0000802C@Data
Gui_VpvSet	long	65.0 (Q-Value(20))	0x0000803A@Data
Gui_Ipv	long	-1.268280029 (Q-Value(16))	0x0000804A@Data
Gui_Iout	long	-1.477722168 (Q-Value(16))	0x00008032@Data
Gui_PanelPower	long	-82.37736511 (Q-Value(16))	0x0000802E@Data
Gui_IpvSet	long	0.0 (Q-Value(20))	0x00008038@Data
State_New	int	1	0x0000800F@Data
Duty	long	0.5387295485 (Q-Value(24))	0x00008870@Data
DutyBuck	long	1.0 (Q-Value(24))	0x00008872@Data
DutyBoost	long	0.09529054165 (Q-Value(24))	0x0000886E@Data
Ipv_Ref	long	0.0 (Q-Value(24))	0x00008066@Data
Vout_Ref	long	0.0 (Q-Value(24))	0x00008064@Data
Add new expression			

Boost

Fig 21 Snap shot of CCS Watch Window for Build 1using Reverse Power Flow at different operation modes Buck, Br-A, Br-B and Boost

12. This completes the reverse direction demonstration.

13. To end the experiment write a 0 to Gui_VpvSet. Now wait till the Gui_VpvSet equal to zero. Now make the input voltage zero. 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 .

14. Finally the controller power can be removed as well if no further experiments are to be carried out.

SCHEMATIC DISCLAIMER AND WARNINGS

TI provides the High Voltage DMC and PFC kit schematic drawings to help users develop C2000 based reference design products. Application safety, safety of the High Voltage DMC and PFC kit and design integrity of such reference designs are solely responsibility of the user. Any reference designs generated off these schematics must take into account necessary product safety design requirements, including interface components and load motors in order to avoid user risks including potential for fire hazard, electrical shock hazard and personal injury, including considerations for anticipated agency certification compliance requirements.

Such product safety design criteria shall include but not be limited to critical circuit creepages and clearances, component selection, ratings compatibility of controlled motor loads, and required protective means (ie output fusing) depending on the specific loads being controlled.

TI accepts no responsibility for design integrity of any reference designs based on supplied schematic drawings and the schematics are strictly for development purposes.

EVALUATION BOARD/KIT IMPORTANT NOTICE

Texas Instruments (TI) provides the enclosed product(s) under the following conditions:
This evaluation board/kit is intended for use for **ENGINEERING DEVELOPMENT, DEMONSTRATION, OR EVALUATION PURPOSES ONLY** and is not considered by TI to be a finished end-product fit for general consumer use. Persons handling the product(s) must have electronics training and observe good engineering practice standards. As such, the goods being provided are not intended to be complete in terms of required design-, marketing-, and/or manufacturing-related protective considerations, including product safety and environmental measures typically found in end products that incorporate such semiconductor components or circuit boards. This evaluation board/kit does not fall within the scope of the European Union directives regarding electromagnetic compatibility, restricted substances (RoHS), recycling (WEEE), FCC, CE or UL, and therefore may not meet the technical requirements of these directives or other related directives.
Should this evaluation board/kit not meet the specifications indicated in the User's Guide, the board/kit may be returned within 30 days from the date of delivery for a full refund. THE FOREGOING WARRANTY IS THE EXCLUSIVE WARRANTY MADE BY SELLER TO BUYER AND IS IN LIEU OF ALL OTHER WARRANTIES, EXPRESSED, IMPLIED, OR STATUTORY, INCLUDING ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE.
The user assumes all responsibility and liability for proper and safe handling of the goods. Further, the user indemnifies TI from all claims arising from the handling or use of the goods. Due to the open construction of the product, it is the user's responsibility to take any and all appropriate precautions with regard to electrostatic discharge.
EXCEPT TO THE EXTENT OF THE INDEMNITY SET FORTH ABOVE, NEITHER PARTY SHALL BE LIABLE TO THE OTHER FOR ANY INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES.
TI currently deals with a variety of customers for products, and therefore our arrangement with the user **is not exclusive**.
TI assumes **no liability for applications assistance, customer product design, software performance, or infringement of patents or services described herein**.
Please read the User's Guide and, specifically, the Warnings and Restrictions notice in the User's Guide prior to handling the product. This notice contains important safety information about temperatures and voltages. For additional information on TI's environmental and/or safety programs, please contact the TI application engineer or visit www.ti.com/esh.
No license is granted under any patent right or other intellectual property right of TI covering or relating to any machine, process, or combination in which such TI products or services might be or are used.

Mailing Address:

Texas Instruments

Post Office Box 655303

FCC Warning

This evaluation board/kit is intended for use for **ENGINEERING DEVELOPMENT, DEMONSTRATION, OR EVALUATION PURPOSES ONLY** and is not considered by TI to be a finished end-product fit for general consumer use. It generates, uses, and can radiate radio frequency energy and has not been tested for compliance with the limits of computing devices pursuant to part 15 of FCC rules, which are designed to provide reasonable protection against radio frequency interference. Operation of this equipment in other environments may cause interference with radio communications, in which case the user at his own expense will be required to take whatever measures may be required to correct this interference.