

Implementing a Digitally Controlled Renewable Energy System with C2000 Micro-Controllers

Oct 22 2009

Abstract

This document presents the implementation details of a digitally controlled renewable energy system using a C2000 controller. This system has been developed to show how a C2000 device interfaces and controls various power stages in such a system, and to provide a means to learn and experiment digital control with C2000 controllers. A renewable energy system, like the one implemented here, is suitable for residential and/or industrial applications. Such a system typically provides a regulated AC output voltage that may also track the input mains utility voltage in phase and amplitude at hundreds to thousands of Watts.

The system developed here is only one way of implementation. Various power stage topologies and different implementations are possible in such a system. The renewable energy system described here is designed to demonstrate the functionality of a larger system at a safe input voltage and power level. As such the input voltage is restricted to a maximum of 20V DC input and around 24V AC output at up to 40 Watts. It is not intended to demonstrate any extremes in electrical design (smallest size, highest efficiency, etc). The emphasis is on using this board as a learning platform that includes a conservative board design and many test points, allowing the user to easily probe the most significant waveforms.

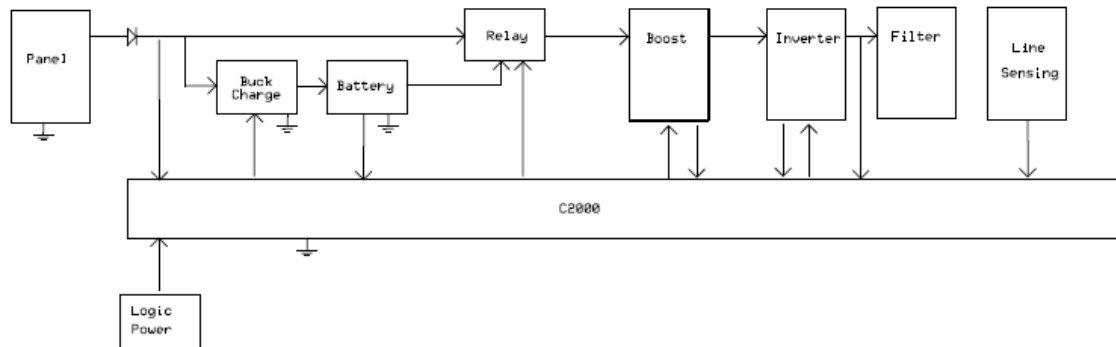


Figure 1. System Block Diagram

Fig 1 gives a simplified block diagram of the system here. The system consists of an inverter section, a front-end DC-DC boost section, a line sensing/synchronizing section and a synchronous buck battery charging section. This board offers all the voltage and current measurement hooks so that one can create and test new topologies and techniques.

The transition to digital power control means that functions that are previously implemented in hardware are now implemented in software. In addition to the flexibility this adds to the system, this simplifies the system considerably. In the current release software supports the F2808, F28027 and F28035 device control cards.

1 Introduction

The ever increasing demand and cost of conventional energy sources coupled with their limited availability and increasing awareness towards environmental factors has prompted researchers to investigate alternate modes of energy. Solar energy systems are at the forefront of this effort. Similarly fuel cells and other sources of energy are also being investigated as possible renewable sources for certain applications. A solar panel or a fuel cell produces direct current (DC) power. Therefore, a power conditioning system is essential to produce commercial alternating current (AC) power (120/220 V, 60/50 Hz). A simple generic block diagram of such a system is shown below:

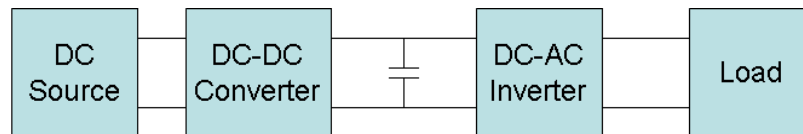


Figure 2. Block Diagram of a Typical Renewable Energy System

The DC-DC converter stage could be a *boost* or a *buck* implementation depending on the voltage available at the source and the application/load requirement. This stage also provides voltage regulation on the intermediate DC bus that is not available at the source terminals. For residential applications the inverter is usually a single phase DC-AC implementation. However, 3-phase implementations are also possible. For some applications it is beneficial to tie this output to the grid. In this case the inverter output has to be synchronized with the line voltage. This necessitates a line synchronization stage implementation (shown in Fig 1). There could be an energy storage mechanism in the system in the form of big DC bus capacitors and/or batteries. This could help provide for sudden load transients or serve as a back-up in case of power outages or lack of enough energy generation from the DC source (shown in Fig 1).

While making these sources as energy efficient as possible, is the key element of research, it is also important to get the most out of the power conditioning system. Some of the key considerations at the system level are the power stage efficiencies and power density, design complexity and scalability, design flexibility, system level integration and low system cost to make these renewable energy systems viable.

Today's high performance 32-bit microcontrollers, like Texas Instruments C2000 family of controllers, combine advanced control peripherals with a DSP like processing power providing reliability, high intelligence and flexibility to the system in addition to simplifying the system design considerably. This provides a means to efficiently control multiple power stages and multiple sections in a system using a single controller. Furthermore, implementing advanced control algorithms and features like advanced maximum power point tracking, temperature compensation, anti-islanding, and digital PLL implementation for line synchronization is relatively easy to achieve using this type of digital controllers. These controllers are also known to support power line communication (PLC) that allows cities to pinpoint power outages and enable e-metering applications. These controllers support different standard communication ports for simple interface with other components or host control and are available in different memory and peripheral configurations to suit the application requirements.

This document presents the implementation details of such a digitally controlled renewable energy system using a C2000 controller. It presents the hardware design and interface and the corresponding software implementation.

1.1 Implementation on the Renewable Energy Board

Fig 2 shows the simplified block diagram of the complete circuit implemented here. This system can be divided into four main sections; a DC-DC boost front end, a single phase or a three phase Inverter section, a DC-DC buck battery charging section, and an AC line synchronization section. There is also a Relay to switch between the panel/DC source and the battery depending on whether the renewable source is available to supply the load requirements or not.

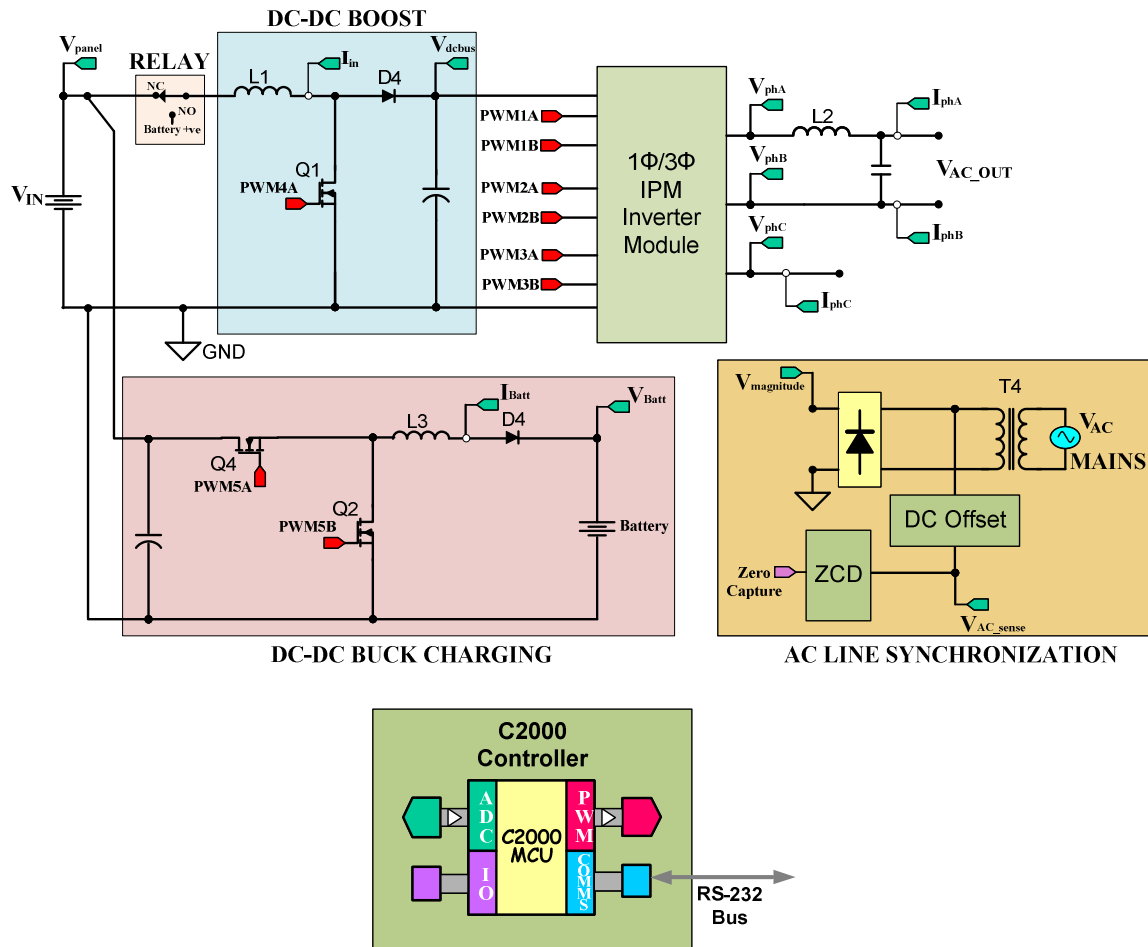


Figure 2. Renewable Energy System

The Relay decides which source is connected to the DC-DC boost stage. Inductor $L1$, MOSFET switch $Q1$ and diode $D4$, together, form the boost circuit. The boost output voltage is fed-back to the ADC to close the voltage loop and regulate this output DC bus voltage. The boost circuit operates at 25 KHz. An Integrated Power Module (IPM) inverter is used to generate either a single phase or a three phase AC output.

Note that although it is possible to connect output filters for a three phase output on this board, the renewable energy board comes with an on-board LC output filter for single phase inverter output only. Six PWM controller outputs drive this inverter at 25 KHz. All the output phase voltages and currents are sensed via the ADC as shown in Fig2.

MOSFET switches $Q4$ and $Q2$ form the DC-DC synchronous buck battery charging circuit. Both the charging current and battery voltage are sensed to allow constant current and/or constant voltage charging. The PWM switching frequency is 100 KHz, while the control loop is executed at 25 KHz.

Finally, the line synchronization stage helps sense the AC line voltage waveform, the zero crossing of the line voltage for phase synchronization and the voltage magnitude.

The control algorithm is implemented on a C2000 micro-controller (MCU). The MCU interacts with the hardware by way of the feedback signals and the PWM drive signals. Thus the key control peripherals on C2000 devices for this application are: the on-chip ADC to sense all feedback signals mentioned above, PWM modules to drive the MOSFET switches, different communication peripherals and different general purpose IOs, as shown in Fig 2. TZ or the trip zone pins are a part of the PWM modules and perform the function of protection.

The duty cycle of the PWM output driving the Q1 MOSFET switch determines the amount of boost imparted by the boost converter. This is the controlled parameter for the boost stage.

For the Inverter stage, the duty cycles of the four/six PWM signals driving the two/three inverter legs are modulated in a sinusoidal fashion. This modulation is also synchronized in phase and frequency to the AC mains input voltage to provide an appropriate AC output for the load. Thus the controlled parameters here are the phase, frequency and amplitude of this sinusoidal modulation, which in effect control the duty cycle of the PWM signals.

The duty cycle of the PWM output driving the Q4 and Q2 MOSFET switches determines the amount of charging energy imparted to the battery. This duty is the controlled parameter for the battery charging stage.

All this control is made possible by the high frequency/high resolution PWM modules, an equally high control loop frequency and a 12-bit on chip ADC available on the high performance 32-bit C2000 controllers. A detailed description of the software algorithm is provided in the following chapters.

CAUTION

This board is a stand alone board that is not enclosed. It is intended for development environment only. In case the 110/220V line voltage is connected for synchronization with the grid the user is advised to exercise extreme caution.

1.2 Electrical Specifications

Following lists the key highlights of the renewable energy board.

- Panel/DC Input Voltage: 9V to 20V DC.
- Battery Voltage: 12 V DC.
- Single phase or three phase Inverter operation.
- Maximum inverter output voltage 15Vrms (with appropriate VdcBus voltage). Normal operation below 12Vrms.
- Maximum inverter output current 1A (with appropriate VdcBus voltage).
- Maximum VdcBus voltage (Inverter input) of 30Vdc.

- 25 kHz switching frequency for the Inverter.
- 25 kHz switching frequency for the DC-DC boost stage.
- 100 kHz switching frequency for battery charging stage.

1.3 Identifying Key components on the board

Now let's identify the key components in the block diagram on the actual hardware.

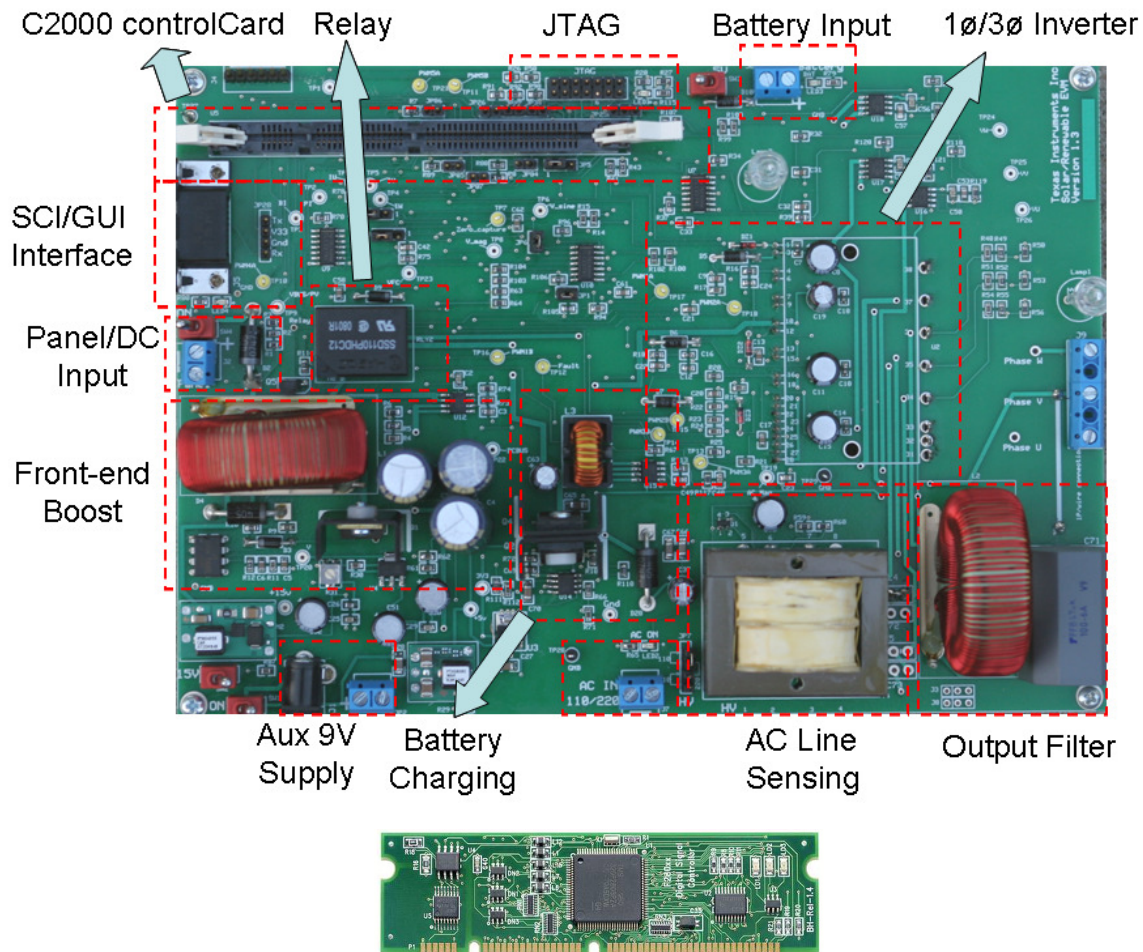


Figure 3. a) Renewable Energy Board, b) ControlCard

2 Software overview

2.1 Software Control Flow

The Renewable project makes use of the “C-ISR/background” framework. It 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 C code also contains the ISR (Interrupt Service Routine), which runs all the critical control code and typically this includes ADC reading, control calculations, and PWM updates. Fig 4 depicts the general software flow for this project.

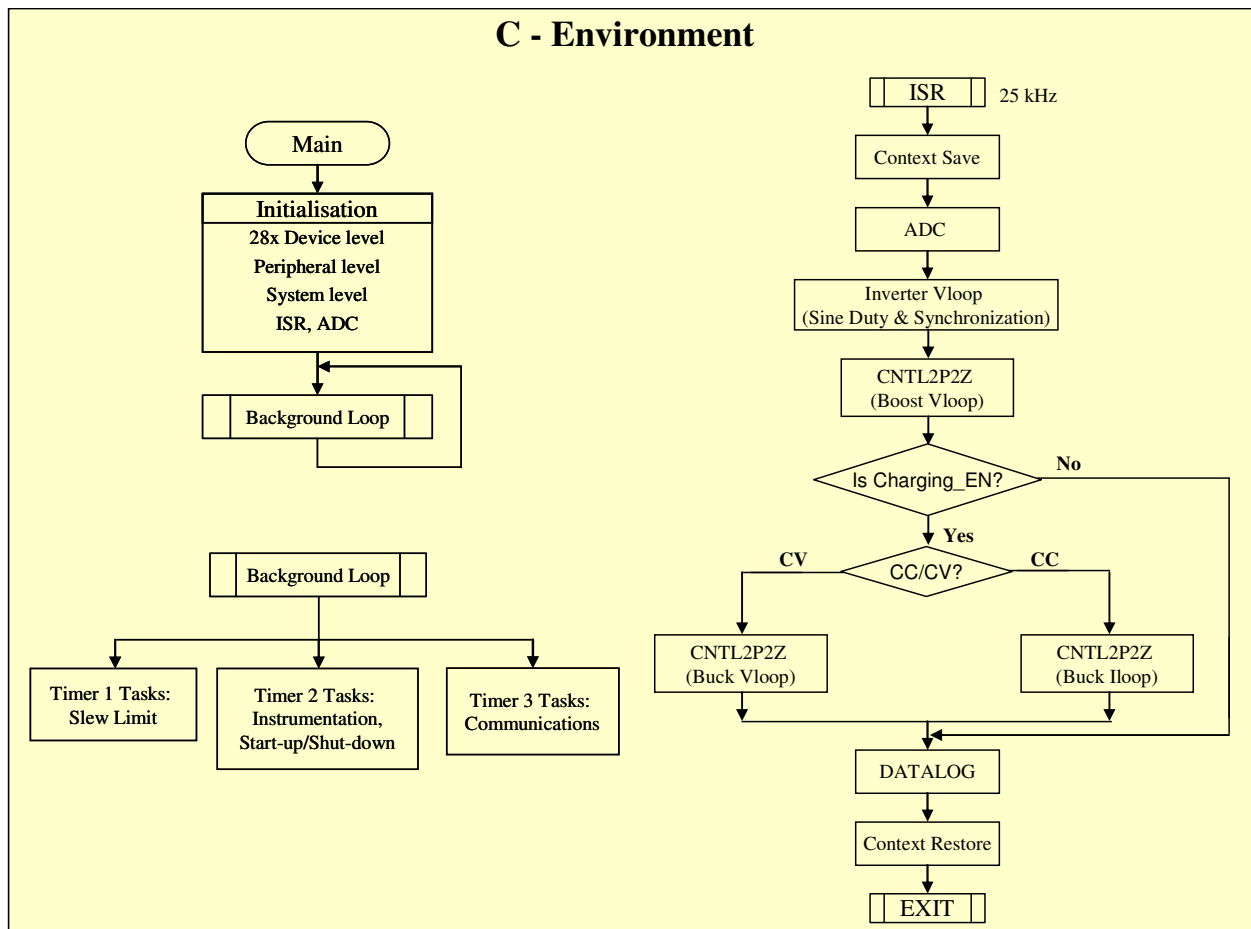


Figure 4. Software flow

The project has configuration built in for F2808, F28027(Piccolo-A) and F28035(Piccolo-B) devices. The user can select any one of these depending on the control card being used. (Note: Jumper settings on the board might vary depending on the device being used). The key framework C files used in this project are:

Renewable-Main-F280x/ Renewable-Main-F2802x/ Renewable-Main-F2803x.c – this file is used to initialize, run, and manage the application. This is the “brains” behind the application. It also contains all time critical “control type” code.

Renewable-DevInit-F280x.c/ Renewable-DevInit-F2802x/ Renewable-Main-F2803x.c – this file is responsible for a one time initialization and configuration of the F280x device, and includes functions such as setting up the clocks, PLL, GPIO, etc.

Sgen.c – this file is responsible for generating the sinusoidal reference commands for the Inverter duty cycle. The ‘gain’ and ‘frequency’ of this sine function can be adjusted.

As shown in Fig 4, the main code first initializes the controller at the device level, the peripheral level and the system level. This is where all the PWMs, ADCs, GPIOs and ISRs are initialized. The ISR is triggered at 25 KHz using ePWM1 event. This is where the Inverter output voltage loop is closed and the line synchronization takes place. A 2-pole 2-zero (2P2Z) algorithm is used to close the DC-DC boost voltage loop. When enabled, the battery charging algorithm may be operating in a constant voltage (CV) loop or a constant current (CC) loop. The software allows the user to decide which loop to operate in. Alternatively, this charging algorithm may be automated to switch the control from a CC charging mode to a CV charging mode. Such an example is shown in the file “Renewable-Main_batt_algo.c”.

The ISR code is executed at a rate of 25 KHz i.e every 40 us. However, the ISR code doesn't take this long to execute and only runs for a part of this 40 us time interval. The slower background loop is executed for the remaining time in this interval. This is where slower system tasks like instrumentation, soft start/shutdown, communications, slew limit codes, etc are executed.

There is a lot of flexibility on C2000 devices to trigger various events and to perform the control in a very deterministic manner.

2.2 Inverter and Line Synchronization

Inverters transform the direct current (DC) energy produced by renewable sources like solar PV modules into the alternating current (AC) energy. Grid-tied inverters synchronize the output they produce with the grid's utility grade AC line, allowing the system to feed this energy to the utility grid. Therefore, the inverter control algorithm should be synchronizing its AC output to the exact AC voltage and frequency of the grid.

The inverter stage is implemented using an intelligent power module (IPM) This module integrates all the power switching MOSFETs of a three phase voltage source inverter with the necessary MOSFET pre-drivers in one single package. It needs six PWM signals to control six switches that are provided by the C2000 controller.

The IPM module doesn't offer any dead-band generation capability and it is the responsibility of the user to provide sufficient dead-band with the applied PWM signals. C2000 devices offer accurate and reliable dead-band generation capability suitable for these applications.

On the renewable energy board, J7 is connected to the utility to monitor the zero crossings of the line voltage to synchronize the inverter output and the mains power system. Note that, one can easily implement 3-phase synchronization using the zero crossing detection circuit and the 3-phase inverter on the board assuming the phase B and phase C are following phase A with 120° phase lag. After scaling down the utility voltage by the transformer (T4), and the necessary signal conditionings on the board, the zero crossing pulses (zero_capture) generated by an on-board comparator are sent to GPIO 24 to set the external interrupt flag. (**Note for F2802x:** GPIO 19 is configured as the ZCD pin, this is configured by setting the SW3 pin on the F2802x control card. This switch routes the pin 19 of the device to GPIO24 pin on the DIMM100). Each time the zero crossing pulse is received by MCU; the code in Renewable -Sgen1.c checks the phase angle of reference signal and keeps it minimum simply by adjusting the frequency of reference signal. This procedure provides good synchronization of utility voltage and inverter output reference signal. One may also employ simple PLL (phase locked loop) type algorithms or any other frequency tracking functions, if the power system is not stabilized and exposed to fast transients. Similar to the frequency adjustment, the amplitude of the inverter output is tuned in Renewable -Main.c based on the mean value of rectified single phase output and updated after receiving zero crossing pulse. Both the frequency and amplitude correction incremental steps are increase or decreased depending on the amount of error between feedback and the reference values.

2.3 DC-DC Boost Converter

The Renewable Energy board offers a front-end boost converter to boost the input voltage or the battery voltage to a suitable level for the inverter. The boost converter is a traditional single phase converter with a single switching MOSFET. This DC-DC boost converter operation is controlled by way of ePWM4A output and the *VdcBus* feedback on ADC channel B2. In open loop operation (*closeLoop_boost = 0*) the amount of boost can be directly controlled through the software/watch window by changing the ePWM4A duty (i.e. by changing *EPwm4Regs.CMPA.half.CMPA*). ePWM4 is operated in up-down count mode at 25 KHz switching frequency.

A 2-pole 2-zero controller is used to close the DC-DC boost voltage loop and regulate the DC bus voltage. The duty value of the PWM output is calculated from the 2P2Z output using the following equation:

$$\text{boost_cmd} = \text{EPWM4_PRD} - (\text{cntl2p2z_boost.Out} * \text{EPWM4_PRD})$$

where,

$\text{cntl2p2z_boost.Out}$ = Normalized output of the 2P2Z controller (For a Q15 representation of the 2P2Z output this is $= \text{cntl2p2z_boost.Out}(Q15)/(2^{15})$, and

boost_cmd = calculated duty value for the up-down count mode.

The control flow for the boost stage is shown in Fig 5. 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 'orange' 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. Such a modular representation makes it convenient to visualize and understand the complete system control flow. It also allows for easy use and additions/deletions of various functionalities.

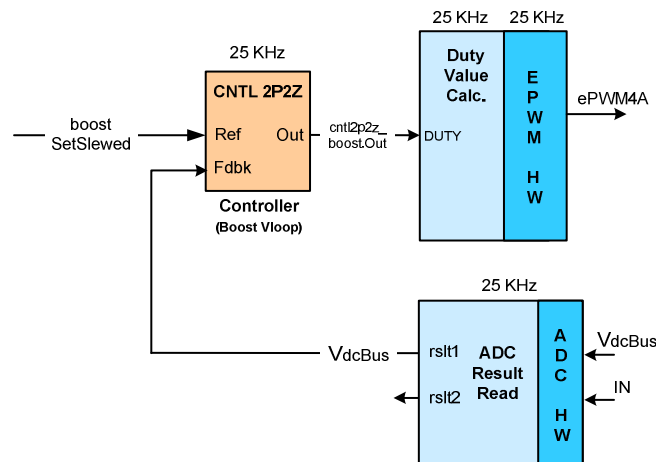


Figure 5. Boost Control flow

Although not used in the control loop, there is a provision made in the hardware to also sense the DC-DC boost inductor current on ADC channel B0.

2.4 Battery Charging

The renewable energy board has a synchronous DC/DC buck stage which can be utilized to implement a battery charging algorithm. The input DC voltage to the board (connected at J2) is the input to the DC/DC converter as well. This DC-DC buck converter operation is controlled by way of ePWM5A/5B outputs, and the V_{batt} and I_{batt} feedback on ADC channels A1 and B1 respectively. (**Note for F2802x**, ePWM3A and 3B are used for battery charging, For this pins 2 and 3 of JP3 & JP4 are connected together. Now the ePWM3A and ePWM3B do not go to the IPM, however this does not effect the Inverter operation being observed in this project as we observe only U & V phases). In open loop operation (*closeloop_buck* = 0) the output voltage can be directly controlled through the software/watch window by changing the ePWM5 duty (i.e. by changing *EPwm5Regs.CMPA.half.CMPA*). The PWM is operated in the up-down count mode and the switching frequency is selected as 100 KHz. The control loop runs at 25 KHz and resides inside the ISR.

The DC/DC stage can be utilized as a battery charger or as a stand alone DC/DC buck. The included light bulb can be utilized as the DC/DC converter load.

The battery charging algorithm may be operating in a constant voltage (CV) loop or a constant current (CC) loop. 2P2Z controllers are used to close the constant current or the constant voltage loops. The duty value of the PWM output in CV mode is calculated from the 2P2Z output using the following equation:

$$buck_cmd = EPWM5_PRD - (chrging_V.Out * EPWM5_PRD)$$

where,

chrging_V.Out = Normalized output of the 2P2Z voltage controller (For a Q15 representation of the 2P2Z output this is = *chrging_V.Out* (Q15)/(2¹⁵), and

buck_cmd = calculated duty value for the up-down count mode.

When operating in CC mode the duty value is calculated using a similar equation except that the term *chrging_V.Out* is replaced by *chrging_I.Out*, which is the output of the 2P2Z current controller.

The control flow for the DC-DC buck battery charging stage in CC (constant current) mode of operation is shown in Fig 6. Notice that while the control loop is executed at 25 KHz, the output PWM frequency is 100 KHz.

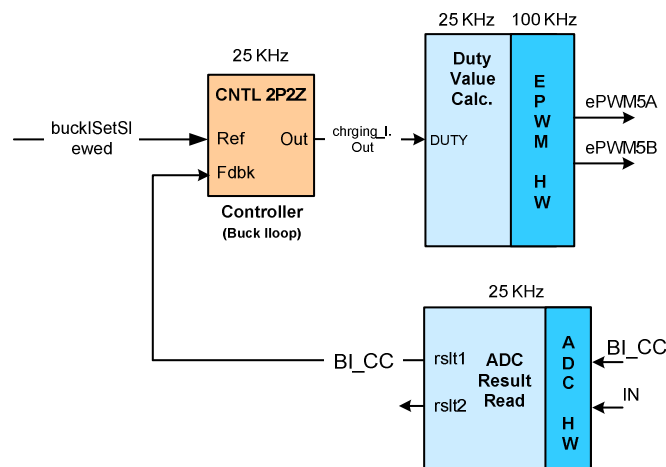


Figure 6. Battery Charging CC Mode Control flow

The control flow in the CV mode of operation is similar to Fig 6, except that the reference and feedback signals now point to the constant voltage reference command and the battery voltage.

2.5 2-Pole 2-Zero Controller

A two pole two zero controller block is used for the front-end boost voltage loop. Similarly one such block is used for the battery charging voltage loop, and another one for the battery charging current loop. Depending on the application's control loop requirements some other controller block like a PI, a 3-pole 3-zero, etc may also be used. CNTL2P2Z is a 2nd order compensator realized from an IIR filter structure. The 5 coefficients needed for this function are declared in the structure by the name CNTL2P2Z. This compensator is defined as a Macro CNTL2P2Z_MACRO.

The CNTL2P2Z_MACRO block can be instantiated multiple times if the system needs multiple loops, as is the case here. Each instance can have separate set of coefficients. The CNTL2P2Z_MACRO instance for the boost voltage loop uses the coefficients stored in the structure *cntl2p2z_boost*. Similarly CNTL2P2Z_MACRO instance for the battery charging voltage loop (CV mode) uses the coefficients stored in the structure *chrging_V*, while the battery charging current loop (CC mode) uses the coefficients stored in the structure *chrging_I*.

Directly manipulating the five coefficients independently by trial and error is almost impossible, and requires mathematical analysis and/or assistance from tools such as matlab, mathcad, etc. These tools offer bode plot, root-locus and other features for determining phase margin, gain margin, etc.

To keep loop tuning simple and without the need for complex mathematics or analysis tools, the coefficient selection problem has been reduced from five degrees of freedom to three, by conveniently mapping the more intuitive coefficient gains of P, I and D to B0, B1, B2, A1, and A2, as shown below. This allows P, I and D to be adjusted independently and gradually.

For the boost stage voltage loop these P, I and D coefficients are: Pgain, Igain and Dgain respectively. For the battery charging voltage loop these P, I and D coefficients are: Pgain_Vbatt, Igain_Vbatt and Dgain_Vbatt respectively. Similarly, for the battery charging current loop these coefficients are: Pgain_Ibatt, Igain_Ibatt and Dgain_Ibatt respectively.

The compensator block (CNTL2P2Z) has 2 poles and 2 zeros and is based on the general IIR filter structure. It has a reference input and a feedback input coming from the ADC. The transfer function is given by:

$$\frac{U(z)}{E(z)} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}}$$

The recursive form of the PID controller is given by the difference equation:

$$u(k) = u(k-1) + b_0 e(k) + b_1 e(k-1) + b_2 e(k-2)$$

where:

$$\begin{aligned} b_0 &= K_p' + K_i' + K_d' \\ b_1 &= -K_p' + K_i' - 2K_d' \\ b_2 &= K_d' \end{aligned}$$

And the z-domain transfer function form of this is:

$$\frac{U(z)}{E(z)} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 - z^{-1}} = \frac{b_0 z^2 + b_1 z + b_2}{z^2 - z}$$

Comparing this with the general form above, we can see that PID is nothing but a special case of CNTL2P2Z control where:

$$a_1 = -1 \quad \text{and} \quad a_2 = 0$$

2.6 Background loop

As mentioned before the background loop executes a slow state machine that goes through various tasks. These are usually the slower system level tasks that include fault management, slew limits, serial communication, instrumentation/display (dashboard measurements) and implementing the start-up and shut-down routine, if desired.

A task state-machine has been implemented as part of the background code. Tasks are arranged in groups (A1, A2, A3..., B1, B2, B3..., C1, C2, C3...). Each group is executed according to 3 CPU timers which are configured with periods of 1 ms, 50 ms, and 100 ms respectively. Within each group (e.g. “B”) each task is run in a “round-robin” manner. For example, group B executes every 50 ms, and there are 2 tasks in group B. Therefore, B1 and B2 execute once every 100 ms. System dashboard measurements are conveniently done by group 3 tasks (i.e. B1 – voltage measurement, B2 – current measurement).

In this project, other than the dashboard measurements the background loop also handles the serial communications with the host GUI, enabling/disabling different power stages, changing PID gains for the control loop, and providing slew rates for various control loop reference commands. The approach to implement the slew rates is only one way of implementation. Users may wish to implement their own routines.

3 Getting Familiar with the Hardware

3.1 The Control Card Interface

The C2000 microcontroller based control cards are the computing engine for this board. The appropriate control card must be installed on the board by plugging the control card into 100pin DIMM socket.

The control card shown below must be installed in the DIMM socket (**U5/DIM100**). The socket is keyed and the card can be installed only in one way. The renewable energy board provides the required 5Vdc to the control card. For F2802x, SW3 of the control card must be set appropriately to route the GPIO19 of the device to pin GPIO24 off the DIM100.

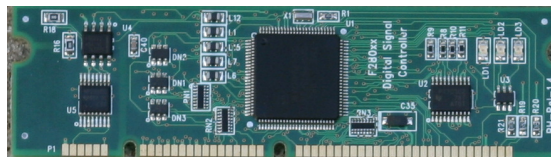


Figure 7. TMS320F2808/ TMSF2802x/ TMSF2803x based control card

3.2 JTAG Port

The board has a 14-pin JTAG header to connect external emulator for debugging and code development purpose. The JTAG pins are pulled externally low or high to improve the noise immunity of the board. The board offers the option of providing 5V to the power pin (5) of the **JTAG** connector. The 5V option offers backward compatibility with earlier C2000 emulators. In most of the cases the supply voltage will be at 3.3V set by the jumper **JP25**. The following table shows JP25 jumper options –

Jumper position	JTAG Pin 5
2-3 (Default)*	3.3V
1-2*	5.0V

Table 1. JP25 Jumper options

**pin1 of JP25 is the nearest to the silkscreen name JP25*

The following table shows the pin mapping of the JTAG header and the corresponding pull-up/down configurations –

Pin Name	Pin	Pull up/dwn	Pin Name	Pin	Pull up/dwn
TMS	1	up	nTRST	2	dwn
TDI	3	up	GND	4	-
VCC	5	3.3/5.0V	NC (Key)	6	-
TDO	7	up	GND	8	-
TCK	9	-	GND	10	-
TCK	11	-	GND	12	-
EMU0	13	-	EMU1	14	-

Table 2. JTAG Header pin mapping

3.3 The Boot Options

This board offers all the boot options available in 280xx and 2833x family of devices. The appropriate jumpers can be populated for the desired boot option. The following table shows the jumper and GPIO mapping for 280xx boot options –

Jumper	GPIO	Default Jumper Position
JP34	GPIO34	Populated (Serial Boot)
JP26	GPIO29	Not populated
JP27	GPIO18	Not populated

Table 3. Boot options for 280xx devices

The following table shows the jumper and GPIO mapping for 2833x boot options –

Jumper	GPIO	Default Jumper Position
JP84	GPIO87	Not populated
JP85	GPIO86	Not populated
JP86	GPIO85	Not populated
JP87	GPIO84	Not populated

Table 4. Boot options for 2833x devices

3.4 Powering the Renewable Energy Board

This board generates all the required DC logic voltages to power the logic circuitry, control card, and the IPM (Intelligent Power Module). It takes a 9V supply at **J1**. The power pack is included with the kit. The user can also use the terminal block **JP2** to provide 9V using a separate DC power supply. The toggle switch **SW1** turns ON/OFF the supply to the board.

The board internally generates the following voltages –

- 15 VDC for Intelligent Power Module (IPM) – test point +15V.
- 5 VDC for control card – test point +5V.
- 3.3 VDC for logic circuitry – test point 3V3
- Vsensor – a variable DC output for reference voltages, if needed (not used here).

3.5 The Input Voltages

The following inputs can be connected to the renewable energy board:

- **Renewable Energy Source/DC Input** – The connector labeled **Panel (J2)** is to connect the external source or any other DC supply as the main power source for the board. The switch **SW4** is to turn ON/OFF the supply to the board. The **LED4** will indicate the ON/OFF status of the supply.

- **110/220AC Input** – The connector labeled **AC IN (J7)** is to connect the line voltage (110 or 220) to synchronize the inverter output. It is extremely important to configure the sensing transformer setup. The **JP7** jumpers need to be set for the correct AC line voltage.
 - **110V AC** – two jumpers covering all four positions.
 - **220V AC** – one jumper covering middle two positions leaving outer positions empty. The silk screen clearly shows the connections. The **LED2** labeled **AC ON** indicates the presence of AC voltage.

CAUTION

This board is a stand alone board that is not enclosed. It is intended for development environment only. In case the 110/220V line voltage is connected for synchronization with the grid the user is advised to exercise extreme caution.

Position	Function
1-2, 3-4	110VAC
2-3	220VAC

Table 5. JP7 - 110/220 Selections

- **Battery Input** – The connector labeled **Battery (J6)** is to connect the back-up battery to the board. This should be a 12V battery. This battery can be used to supply the inverter in the absence of sufficient energy from the renewable source. This battery can be also charged using the DC/DC converter included with the system. **SW3 (5V)** can take the battery in and out of the system. This switch should remain in **OFF** position. **LED3** indicates the status of battery. In case the battery is not connected and DC/DC converter is used to generate a lower voltage the LED3 will indicate the output of the DC/DC converter.

3.6 The Inverter Stage

The inverter stage is implemented using an intelligent power module (IPM) This module integrates all the power switching MOSFETs of a three phase voltage source inverter with the necessary MOSFET pre-drivers in one single package. The module needs a 15VDC supply which is provided by the board. It needs six PWM signals to control six switches. These PWM signals are provided by the C2000 controller and the signal mapping is shown in the table below –

No.	PWM Output	IPM Connection	Test Point
1.	ePWM1A	UP	TP17
2.	ePWM1B	UN	TP16
3.	ePWM2A	VP	TP18
4.	ePWM2B	VN	TP15
5.	ePWM3A	WP	TP13
6.	ePWM3B	WN	TP14

Table 6. PWM output mapping for the inverter power stage

In case of any fault the IPM module shuts-off and generates a fault signal. This fault signal labeled **Fault (TP 12)** is fed back to C2000 controller through a jumper **JP5**. In case of a TMS320F2808 controller card JP5 setting is 2-1 (default) and the fault signal goes to **GPIO28**. In case of a Piccolo controller card JP5 setting is 3-1 and the fault signal goes to **GPIO32**.

Output Phase Voltage Measurements

The inverter output voltages are measured using voltage divider circuits. The voltage divided IPM phase output voltages are fed directly to the ADC inputs. The following table shows the utilized ADC channels as well as the test point allocations –

No.	Analog Signal	IPM output	Test Point	Allocated ADC Channel
1.	PhVU	U	TP26	ADC-A3
2.	PhVV	V	TP25	ADC-A4
3.	PhVW	W	TP24	ADC-A5

Table 7. ADC channel allocation for Inverter Output Voltage measurement

Output Phase Current Measurements

Inverter output phase currents are also sensed and fed to the ADC. The signal from current sense circuit gives a voltage between 0 to 5V DC with zero current corresponding to 2.5V. The negative currents are between 0 to 2.5V and the positive currents are between 2.5V to 5V. A voltage divider circuit is needed to make sure the input to the ADC input of C2000 devices does not exceed 3.3V. A voltage divider circuit with a division of 0.83 is added to lower the output voltage. One needs to take every caution not to exceed the inverter output current so that the ADC input voltage stays below 3.3V. The following table shows the utilized ADC channels as well as the test point allocations (not used in the software) –

No.	Analog Signal	IPM output	Test Point	Allocated ADC Channel
1.	PhIU	Phase U	TP30	ADC-B3
2.	PhIV	Phase V	TP5	ADC-B4
3.	PhIW	Phase W	TP4	ADC-B5

Table 8. ADC channel allocation for Inverter Output Current measurement

Inverter Output Configuration

The hardware is capable of generating three-phase as well as single-phase AC output voltage. The board comes with a single phase L-C output filter. Connector **J9** brings out the three phase inverter output. The following table shows the connection mapping of **J9**

Connection	Phase
1	U
2	V
3	W

Table 9. Connector J9 Mapping

Connector **J3** terminates the negative end of the L-C filter capacitor. The following table shows the connection mapping of **J3**.

Connection	Capacitor/Phase
1	C71/U
2	Non-pop/V
3	Non-pop/W

Table 10. Connector J3 Mapping

Connector **J8** is shorted with-in. This connector can be utilized to create a wye connected three-phase output. The solar board is shipped with a single phase L-C filter installed on phase-U. The L-C filters for the remaining two phases are not installed but all the hardware hooks are provided for later installation.

Zero-crossing Detection of the Line Voltage

The zero crossing circuitry compares a voltage divided, and DC shifted AC line voltage with an appropriate DC voltage level. This generates a square pulse with a rising edge at every positive going zero crossing of the input AC voltage and a falling edge at the

negative going zero crossing. The test point **TP7** can be used to observe the square pulse using an oscilloscope. The jumper **JP6** needs to be populated to input the square pulse to the controller.

Note that the controller and the appropriate software will decide which edge of the square pulse to detect. C2000 controllers can detect rising edge, falling edge or both edges.

The zero crossing detection circuitry provides additional information about the connected line voltage as well. The additional features are described below –

A DC voltage proportionate to the line peak voltage is generated using a full bridge rectifier. This voltage labeled **V_mag_BH**, can be observed at test point **TP8**. This voltage is fed to an ADC channel of the C2000 controller.

In addition to the magnitude, a true AC waveform is also provided. The AC signal is the replica of the connected line voltage but is DC shifted above zero voltage so that the signal can be measured using one of the ADC channels of the controller. The line voltage proportionate analog signal is labeled **V_sine_BH** and can be observed at test point **TP6**.

For F2802x, SW3 of the control card must be set appropriately to route the GPIO19 of the device to pin GPIO24 off the DIM100. All channel allocations and GPIO assignments are provided in a table below:

Function	Signal Name	Test Point	Digital/Analog	GPIO/ ADC Channel	Jumper setting (pop)
Zero crossing detection	zero_capture_BH	TP7	Digital	GPIO-24 (For F2803x & F280x)/ GPIO-19 (For F2802x)	JP6
Line magnitude	V_mag_BH	TP8	Analog	ADC-B6	-
Line voltage	V_sine_BH	TP6	Analog	ADC-A6	JP1*

Table 11. Zero crossing detection related signals and channel assignments

**This adds the necessary offset to the sine voltage so that the voltage becomes positive (uni-polar) and can be safely interfaced with the assigned ADC channel.*

3.7 The Front-end Boost Converter

This board offers a front-end boost converter to boost the input voltage or the battery voltage to a suitable level for the inverter input. The boost converter is a traditional single phase converter with a single switching MOSFET. A single PWM output is used to control the boost converter.

The boost stage input current (inductor current) is sensed and fed to the ADC. The signal from current sense circuit gives a voltage between 0 to 5V DC with zero current corresponding to 2.5V. The negative currents are between 0 to 2.5V and the positive currents are between 2.5V to 5V. A voltage divider circuit is needed to make sure the input to the ADC input of C2000 devices does not exceed 3.3V. A voltage divider circuit with a division of 0.83 is added to lower the output voltage. One needs to take every caution not to exceed the inverter output current so that the ADC input voltage stays below 3.3V.

The renewable/DC source input voltage is measured using a voltage divider circuit. This voltage divided signal is fed directly to the ADC. The input voltage is measured from the cathode of the blocking diode, D2.

Similarly, the boost converter output voltage is also measured using a voltage divider circuit. The boosted DC bus voltage is measured from the cathode of the boost circuit blocking diode, D4. Please refer to the “*Renewable-Calculations.xls*” file that gives detailed calculations for some of these sensed signals.

The following table shows the utilized PWM channel, ADC channels as well as the test point allocations for front-end boost converter stage –

No.	Function	Signal Name	Test Point	ADC Channel	PWM Channel
1.	Boost	ePWM4A_IPM	TP10	-	PWM4A
2.	Input Current	PI	TP2	ADC-B0	-
3.	Input Voltage	VBUS	TP9	ADC-A0/A2	-
4.	Boost Voltage	VPFC	TP23	ADC-B2	-

Table 12. Front-end Boost Converter Signal mapping

3.8 Battery Charging Stage

The renewable energy board has a synchronous DC/DC buck stage which can be utilized to implement a battery charging algorithm. The renewable/DC source input voltage to the board (source connected to J2) also serves as the input to the DC/DC converter. The required PWM signals come from C2000 controller.

The battery charging current is sensed and fed to the ADC. The signal from current sense circuit gives a voltage between 0 to 5V DC with zero current corresponding to 2.5V. The negative currents are between 0 to 2.5V and the positive currents are between 2.5V to 5V. A voltage divider circuit is needed to make sure the input to the ADC input of C2000 devices does not exceed 3.3V. A voltage divider circuit with a division of 0.83 is added to lower the output voltage. One needs to take every caution not to exceed the inverter output current so that the ADC input voltage stays below 3.3V

The DC/DC stage can be utilized as a battery charger or can be utilized as a stand alone DC/DC buck. If the DC/DC converter is utilized as a standalone buck converter then **JP8** should be populated and **JP9** should be left unpopulated. The included light bulb can be

utilized as the DC/DC converter load. If the DC/DC converter is utilized as a battery charger then **JP8** should be unpopulated and **JP9** should be populated. The following table shows the jumper options for DC/DC operations –

Function	Jumper JP8	Jumper JP9
DC/DC Buck	Populated	Not populated
Charger	Not populated	Populated

Table 13. Jumper J8/J9 settings

Similar to other voltage measurements on this board, the DC/DC converter output voltage/Battery voltage is measured using a voltage divider circuit. The output voltage is measured from the cathode of the blocking diode, D28.

This board is compatible to TMS320F2808 control card and Piccolo control card. An appropriate jumper selection is needed depending on installed control card. The following table shows the jumper positions for **JP3** and **JP4** –

Device	JP3 position	JP4 position
TMS320F2808/ F2803x	2-3	2-3
TMSF2802x	1-2	1-2

Table 14. JP3/JP4 Selection

The following table shows the utilized PWM channels, ADC channels as well as the test point allocations for DC/DC buck converter –

Function	Signal Name	Test Point	PWM Channel	ADC Channel
High Side MOSFET Switching	Buck_BH	TP21* TP13**	ePWM5A* ePWM3A**	-
Low Side MOSFET Switching	Buck_BL	TP11* TP14**	ePWM5B* ePWM3B**	-
Charging Current	BI_CC	TP3	-	ADC-B1
Battery/DCDC Voltage	BV	TP1	-	ADC-A1

Table 15. DC/DC Converter Signal Mapping

*TMS320F2808/F2803x Control Card

** F2802x Control Card

3.9 Input Relay Operation

The input to the front-end boost converter can be either the DC voltage connected to the terminal J2 or the battery voltage connected to terminal J6. In a typical scenario the renewable source will be connected to J2 and a battery will be connected to J6. From system operation point of view, either of the renewable/DC source or the battery will provide the energy and the relay will make sure that the correct source is connected to the front-end boost converter.

By default, the relay keeps the renewable/DC source connected. This is a normally connected (NC) relay and connects to the DC voltage during board power-up. The relay can be opened (NO) to connect the battery voltage to the input of the boost converter. The relay operation is controlled using a General Purpose Input/Output (GPIO) pin of C2000 controller. The board utilizes GPIO-17 to control the relay. The following table shows the GPIO mapping and states for the relay operation –

GPIO-17 State	NC	NO	Voltage to the Input of the Boost Converter	TP
0	√		Connector J2 Voltage	Relay
1		√	Connector J6 Voltage	

Table 16. Relay operation using GPIO-17

3.10 Serial Communication Port

This board comes with a standard serial port interface. A standard DB9 (**J5**) connector is provided for serial interface. The board also provides a four pin connector, **JP28**, appropriate for Texas Instruments Inc. serial interface cable included with this kit. The following table shows the pin mapping of **JP28** connector.

Function	Pin name	Pin No
Transmit	TX	1
3.3V (Isolated)	V33-ISO	2
Ground (Isolated)	GND-ISO	3
Receive*	RX	4

Table 17. JP28 Pin mapping

* Align with the Red wire of the serial cable.

The following table shows the pin mapping of the connector, J5

Function	Pin Name	Pin No
Receive	RX	3
Transmit	TX	2
Ground	GND	5
NC	NC	1
NC	NC	4
NC	NC	6
NC	NC	7
NC	NC	8
NC	NC	9

Table 18. J5 Pin mapping

3.11 General Purpose Input/Output Port

This board provides four general purpose input/output pins. These pins are brought out using connector **J4**. The following table shows the GPIO mapping of the connector J4.

Function	GPIO	Pin
Voltage (3.3V)	-	1
Input/Output	GPIO01	2
Input/Output	GPIO02	3
Input/Output	GPIO03	4
Input/Output	GPIO04	5
Ground	-	6

Table 19. Connector J4 pin mapping

4 Procedure for running the Renewable Energy System

➤ Objective

The objective of the Renewable Energy project is to evaluate a complete Renewable Energy system comprising of a front end DC-DC boost stage followed by a line synchronized single phase or a three phase Inverter stage and being controlled by a single C2000 controller. The board also demonstrates a battery charging stage implementation that is also being controlled by the same controller.

➤ Overview

The key signal connections between the C2000 Controller and the Renewable Energy board and important jumper settings are summarized in the tables below.

Signal mapping and resource allocation table

Signal Name	C2000 Signal	Function	Test Point
ePWM1A_IPM	ePWM1A	PWM for upper switch of phase U	TP17
ePWM1B_IPM	ePWM1B	PWM for lower switch of phase U	TP16
ePWM2A_IPM	ePWM2A	PWM for upper switch of phase V	TP18
ePWM2B_IPM	ePWM2B	PWM for lower switch of phase V	TP15
ePWM3A_IPM	ePWM3A	PWM for upper switch of phase W	TP13
ePWM3B_IPM	ePWM3B	PWM for lower switch of phase W	TP14
PhVU	ADC-A3	Phase U Voltage	TP26
PhVV	ADC-A4	Phase V Voltage	TP25
PhVW	ADC-A5	Phase W Voltage	TP24
PhIU	ADC-B3	Phase U Current	TP30
PhIV	ADC-B4	Phase V Current	TP5
PhIW	ADC-B5	Phase W Current	TP4

Table 20. Resource allocations for Inverter operation

Signal Name	C2000 Signal	Function	Test Point
ePWM4A_IPM	ePWM4A	PWM for the boost switch	TP10
PI	ADC-B0	Input current to the boost converter	TP2
VBUS	ADC-A0/A2	Input voltage to the boost converter	TP9
VPFC	ADC-B2	Output voltage of the boost converter	TP23

Table 21. Resource allocations for Front-end Boost Converter

Signal Name	C2000 Signal	Function	Test Point
Buck_BH	ePWM5A* ePWM3A**	PWM for the high side MOSFET of the synchronous DC/DC converter	TP21* TP13**
Buck_BL	ePWM5B* ePWM3B**	PWM for the low side MOSFET of the synchronous DC/DC converter	TP11* TP14**
BI_CC	ADC-B1	DC/DC output current/charging current	TP3
BV	ADC-A1	DC/DC output voltage/Battery voltage	TP1

Table 22. Resource allocations for DC/DC synchronous converter/charger

* TMS320F2808/ F2803x Control card.

** F2802x Control card.

Also ensure proper jumper selection.

Signal Name	C2000 Signal	Function	Test Point
Zero_capture_BH	GPIO-24	The square wave after zero crossing detection.	JP6
V_mag_BH	ADC-B6	Analog voltage proportionate to the magnitude of AC line voltage	TP8
V_sine_BH	ADC-A6	Analog voltage with an off-set proportionate to AC line voltage	TP6

Table 23. Resource allocations for Zero-crossing detection

Connector	Function
J9	Inver output voltage.
J7	AC line input for zero-crossing detection.
JP2	Input 9V
J1	9V input jack.
J2	Main DC/renewable source input
J5	UART/Serial port
U5	100-DIMM for control card
J6	Battery input/DC/DC output.

Table 24. Board connector descriptions.

Switch	Function	Default
SW1	Main supply	OFF
SW2	To feed boost converter input with 15V test voltage (For demo purposes only)	OFF
SW3	To feed battery input with 5V test voltage (Not used)	OFF
SW4	Main Renewable/DC source input to the board	OFF

Table 25. Mechanical switch functionalities

Jumper	Function	Default
JP8	Connects output capacitors to the output of synchronous DC/DC converter/charger. JP9 should be taken out when JP8 is populated.	Open
JP9	Connects capacitors across battery voltage. This should be populated only during charging operations. In case of charging JP8 should be taken out.	Open
JP1	Connects the offset to the conditioned AC line voltage.	Close
JP6	Connects the zero crossing square wave to C2000 controller	Close
JP25	Selects voltage for JTAG pin 5	3-2
JP3	Selects PWM for charging depending on C2000 device	3-2
JP4	Selects PWM for charging depending on C2000 device	3-2
JP5	Selects GPIO for fault signal based on C2000 device	1-2
JP34	280xx boot option pin	Close
JP26		Open
JP27		Open
JP84	2833x boot option pin	Open
JP85		Open
JP86		Open
JP87		Open

Table 26. Jumper settings

LED	Name	Function
LED4	-	Indicates the presence of renewable/DC source
LED2	AC ON	Indicates the presence of AC line voltage
LED3	BAT	Indicates the presence of battery voltage
LED1		Indicates the presence of 3.3V

Table 27. LED descriptions

➤ Procedure

The following steps are for F2803x RAM configuration, differences for other configurations are highlighted for other configurations.

Hardware Set-Up

- Make sure the Switches SW1, SW2, SW3 and SW4 are in “OFF” Position.
- Make sure the following jumpers are plugged in: JP34, JP6, pins 2 and 3 of JP4, pins 2 and 3 of JP3, pins 1 and 2 of JP5, pins 2 and 3 of JP25 (pin1 of JP25 is the nearest to the silkscreen name JP25), Jumper JP1, and appropriate jumper setting at JP7 as shown in Table 5. (Note for F2802x, jumpers are plugged in pins 1 and 2 of JP4, pins 1 and 2 of JP3, and SW3 of the control card to route GPIO19 to GPIO24 DIM100 is set accordingly)
- Connect a suitable C2000 emulator between the JTAG port and the host PC.
- Connect one bulb (lamp load) at J9 between pins 2 and 3 (i.e. between Phase V and Phase U). Connect the second bulb at J6.
- Now make sure that switch SW4 is in the OFF position and connect a 9-12V DC bench power supply at J2. The positive terminal must be connected to the positive pin of J2.
- Connect a suitable power cable at J7 for the AC mains input. **Do not plug in the other end of this cable until asked to do so in the procedure.**

Start CCS and Open a Project

To quickly execute this project using the pre-configured work environment, follow the following steps:

1. Once the hardware is set-up correctly, connect a 9V supply at J1. Turn ON SW1.
2. Double click on the Code Composer Studio icon on the desktop. Maximize Code Composer Studio to fill your screen. Code Composer Studio has a *Connect/Disconnect* feature which allows the target to be dynamically connected and disconnected. This will reset the JTAG link and also enable “hot swapping” a target board. Connect to the target.

Click: Debug → Connect

The menu bar (at the top) lists File ... Help. Note the horizontal tool bar below the menu bar and the vertical tool bar on the left-hand side. The window on the left is the project window and the large right hand window is your workspace.

3. A workspace file has been created for this project. Load the workspace file ***Renewable_F2803x.wks (or Renewable_F2802x or Renewable_F280x.wks)*** by clicking:

File → Workspace → Load Workspace...

and look in ***..\TI_F28xxx_SysSW\Renewable***. Loading the workspace file will automatically open up the project file (*.pjt) for the corresponding project (***Renewable.pjt***). A *project* contains all the files and build options needed to

develop an executable output file (.out) which can be run on the MCU hardware. If for some reason the workspace doesn't open up correctly and a message appears that says "Do you want to try to load as much as possible from the workspace?" Click "Yes". If the corresponding project file doesn't open, on the menu bar click: Project → Open... and navigate to ..\TI_F28xxx_SysSW\Renewable directory and open the corresponding project file.

This project (.pjt file) will invoke all the necessary tools (compiler, assembler, linker) to build the project. It will also create a folder that will hold immediate output files. Ensure the correct configuration is selected by Right clicking Renewable.pjt in the Project Window and select "Configuration..." and select F2803xRAM as active configuration (other configurations available are F2803x_RAM, F2802x_FLASH and F280x_RAM) depending on the control card being used.

4. In the project window on the left, click the plus sign (+) to the left of Project. Now, click on the plus sign next to Renewable.pjt. Click on the plus sign next to Source to see the current source file list.





Device Initialization, Main, and ISR Files

Note: *DO NOT* make any changes to the source files – *ONLY INSPECT*

5. Open and inspect Renewable-DevInit_F2803x.c by double clicking on the filename in the project window. Notice that system clock, peripheral clock prescale, and peripheral clock enables have been setup. Next, notice that the shared GPIO pins have been configured.
6. Open and inspect Renewable-Main-F2803x.c (If it's not already open). Notice the call made to DeviceInit() function and other variable initialization. Also notice code for the ISR, PWM and ADC initialization and the background for(;;) loop.
7. Inspect the *epwm1_isr* interrupt service routine. This is where the control codes for running various control loops (Inverter and line synchronization, boost stage and battery charging) reside. Optionally, you can close the inspected files.

Build and Load the Project

8. The top four buttons on the horizontal toolbar control code generation. Hover your mouse over each button as you read the following descriptions:

Button	Name	Description
	Compile File	Compile, assemble the current open file
	Incremental Build	Compile, assemble only changed files, then link
	Rebuild All	Compile, assemble all files, then link
	Stop Build	Stop code generation

9. Code Composer Studio can automatically load the output file after a successful build. On the menu bar click: Option → Customize... and select the "Program/Project/CIO" tab, check "Load Program After Build".

Also, Code Composer Studio can automatically connect to the target when started. Select the "Debug Properties" tab, check "Connect to the target at startup", then click OK.

10. Select the Incremental build option as 1 in the **ProjectSettings.h** file found under the Include tab in the project window. Only the incremental build option 1 is valid with this software.

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

11. Click the “Rebuild All” button and watch the tools run in the build window. The output file should automatically load.
12. Under Debug on the menu bar click “Reset CPU”, “Restart”, and then “Go Main”. You should now be at the start of Main().

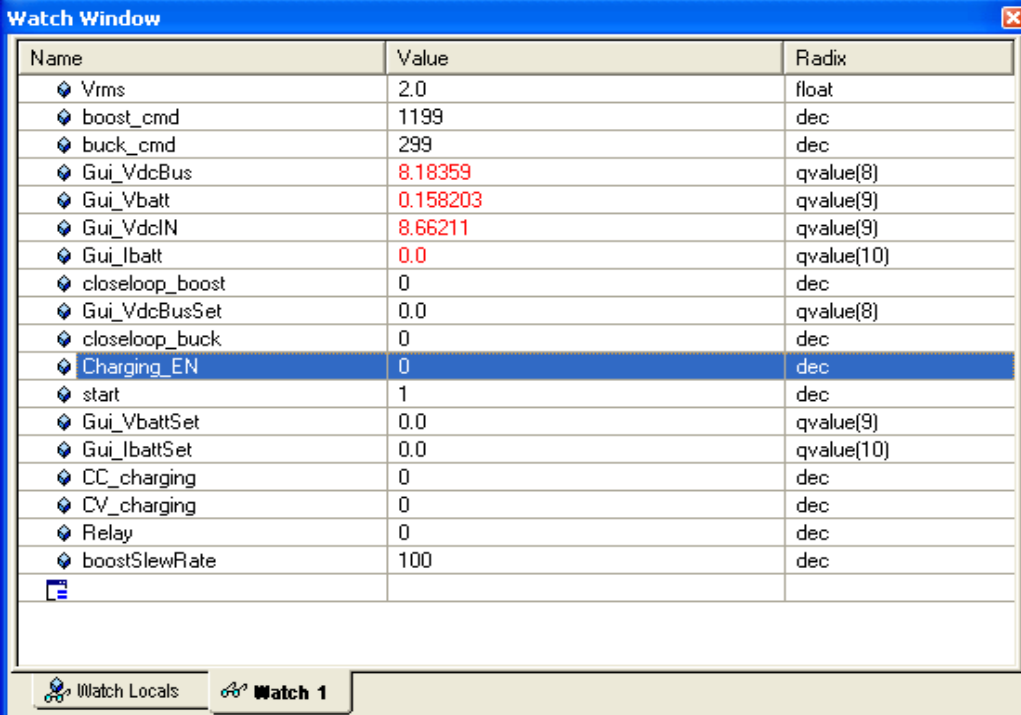
Debug Environment Windows

It is standard debug practice to watch local and global variables while debugging code. There are various methods for doing this in Code Composer Studio, such as memory windows and watch windows. Additionally, Code Composer Studio has the ability to make time (and frequency) domain plots. This allows us to view waveforms using graph windows. We will use two of them here: watch windows and graph windows.

13. Loading the workspace file in step 3 will also opens a pre-configured watch window for this project. If a watch window did not open correctly in step 3, open a new *watch window* and add various parameters to it by following the procedure given below. In this case make sure to save the workspace before closing CCS. This is discussed in the next step.

Click: View → Watch Window on the menu bar.

You may add any variables to the watch window. In the empty box in the "Name" column, type the symbol name of the variable you want to watch and press enter on keyboard. The watch window should look something like:

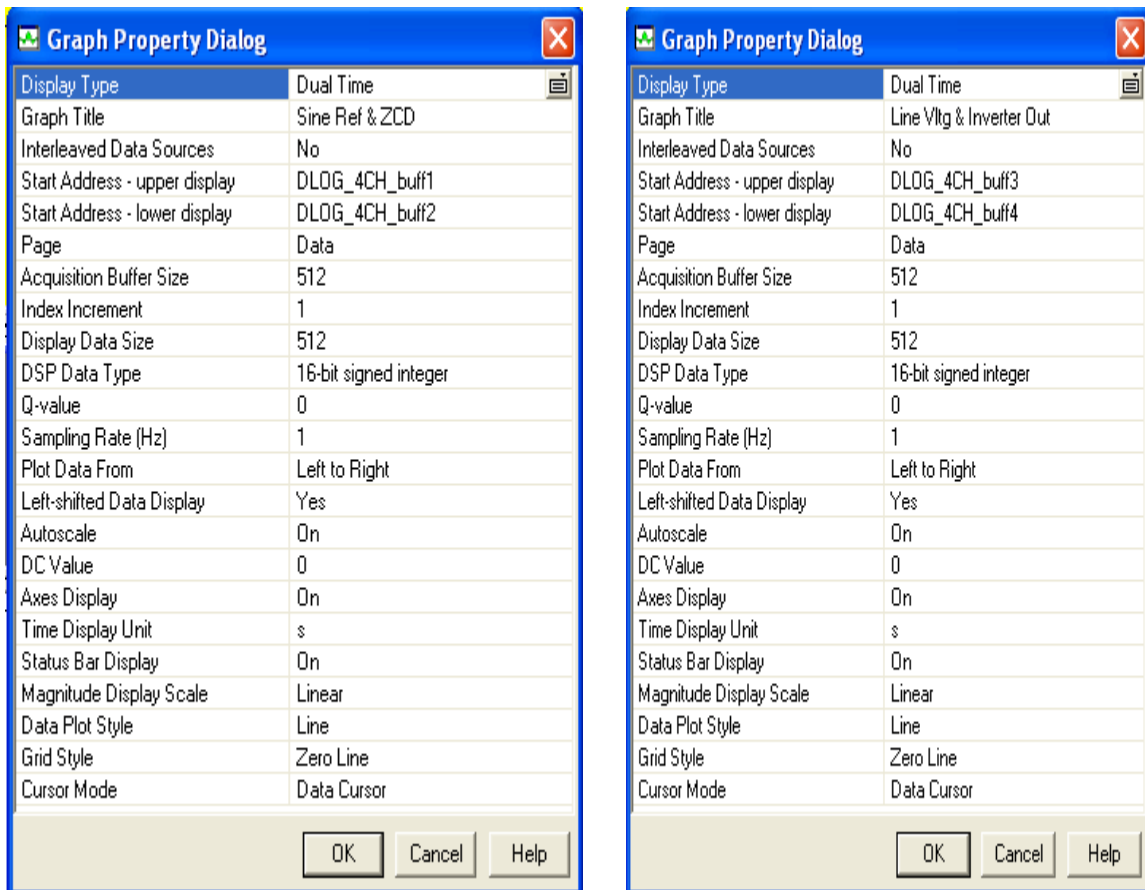


The screenshot shows the 'Watch Window' in Code Composer Studio. It contains a table with three columns: 'Name', 'Value', and 'Radix'. The 'Name' column lists various variables, some with a small icon to the left. The 'Value' column shows the current values, with some in red. The 'Radix' column shows the data format. At the bottom, there are buttons for 'Watch Locals' and 'Watch 1'.

Name	Value	Radix
Vrms	2.0	float
boost_cmd	1199	dec
buck_cmd	299	dec
Gui_VdcBus	8.18359	qvalue(8)
Gui_Vbatt	0.158203	qvalue(9)
Gui_VdclN	8.66211	qvalue(9)
Gui_lbatt	0.0	qvalue(10)
closeloop_boost	0	dec
Gui_VdcBusSet	0.0	qvalue(8)
closeloop_buck	0	dec
Charging_EN	0	dec
start	1	dec
Gui_VbattSet	0.0	qvalue(9)
Gui_lbattSet	0.0	qvalue(10)
CC_charging	0	dec
CV_charging	0	dec
Relay	0	dec
boostSlewRate	100	dec

Figure 8. CCS watch window for Build 1

14. Two graph windows have been configured and saved as a part of the workspace. These are dual time graph windows to plot the four data log buffers DLOG_4CH_buff1, DLOG_4CH_buff2, DLOG_4CH_buff3 and DLOG_4CH_buff4. If the workspace didn't open correctly in step 3, you may set-up the two graph windows as described below. For your reference the procedure to setup a graph window is Click: *View* → *Graph* → *Time/Frequency...* and set the graph properties for the two as below:



Saving the Workspace Environment

The workspace contains all of the elements that make up the current Code Composer Studio working environment. These elements include the project, project settings, configuration settings, and windows such as watch window and graphs. A workspace can be saved in a workspace file (*.wks) and reloaded at a later time. This is very useful for a subsequent Code Composer Studio session, or if a problem occurs and the tools need to be reset.

15. You can save the current workspace by naming it appropriately after clicking:

File → Workspace → Save Workspace As...

and saving in ..\TI_F28xxx_SysSW\Renewable

When needed, a workspace can be loaded by clicking:

File → Workspace → Load Workspace...

and looking in the saved location.

Using Real-time Emulation

Real-time emulation is a special emulation feature that allows the windows within Code Composer Studio to be updated at up to a 10 Hz rate *while the controller is running*. This not only allows graphs and watch windows to update, but also allows the user to change values in watch or memory windows, and have those changes affect the MCU behavior. This is very useful when tuning control law parameters on-the-fly, for example.

16. Enable real-time mode by selecting:

Debug → Real-time Mode

17. 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.

18. In real-time mode, we would like to have our window continuously refresh. Click:

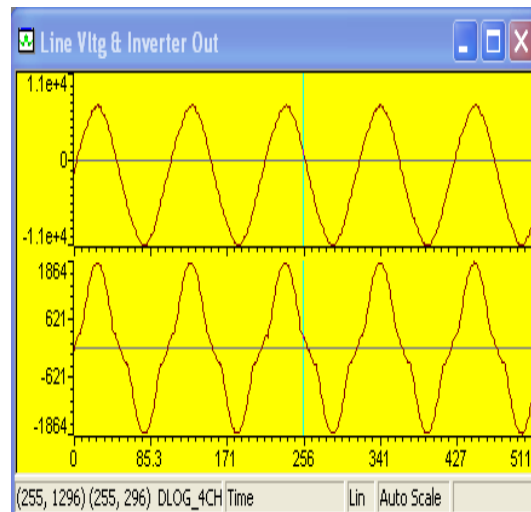
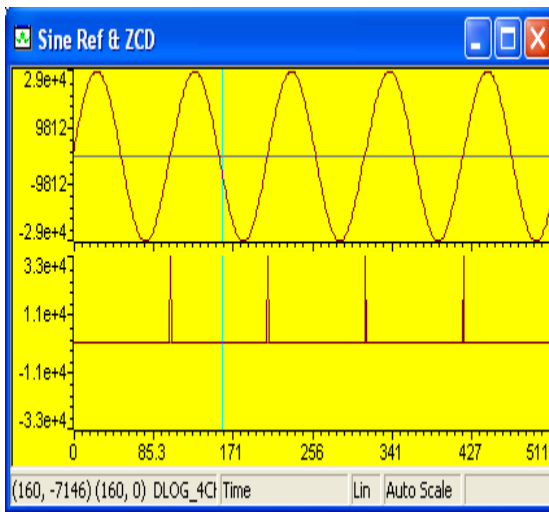
View → Real-time Refresh Options...

and check “Global Continuous Refresh”. Use the default refresh rate of 100 ms and select OK. Alternately, we could have right clicked on each window individually and selected “Continuous Refresh”.

Note: “Global Continuous Refresh” causes all open windows to refresh at the refresh rate. This can be problematic when a large number of windows are open, as bandwidth over the emulation link is limited. Updating too many windows can cause the refresh frequency to bog down. In that case, either close some windows, or disable global refresh and selectively enable “Continuous Refresh” for individual windows of interest instead.

Run the Code

19. Plug-in the other end of the AC line input at J7 to the AC input. The AC ON (LED2) should light up. Use extreme caution when this voltage is connected in the circuit.
20. Now, run the code by using the <F5> key, or using the Run button on the vertical toolbar, or using Debug → Run on the menu bar.
21. In the watch window, the variable *start* should be set to 0. This variable enables the PWM drive to the Inverter and the boost stage.
22. Now turn ON switch SW4 (with the bench DC power supply connected at J2).
23. With *Vrms* set to ‘2.0’ in the watch window, set *start* = 1. The lamp load at the inverter output (J9) should now light up. The two graph windows should now look something like:



Boost Stage

24. Notice that the *boost_cmd* variable is set to 1199(1999 for F280x) by default. This directly controls the compare value for the PWM driving the boost stage and thus controls the duty cycle and the amount of boost imparted. A value of 1199 implies zero duty cycle while a value of 0 implies 100% duty cycle.
25. With the *closeloop_boost* set to 0, we will be operating in open loop and can directly control the amount of boost using *boost_cmd*. The software clamps the maximum boost possible by clamping minimum allowable *boost_cmd* to a value of 840 (1400 for F280x).
26. Observe the change in the amount of boost by observing *Gui_VdcBus* in the watch window for different *boost_cmd* values. Change the *boost_cmd* back to 1199 before going to the next step.
27. The *VdcBus* voltage can be regulated by closing the boost stage voltage loop. As discussed in the previous chapter, this software uses a two pole two zero controller as the control loop with its 5 coefficients mapped to P, I and D coefficients for intuitive tuning. Very low P, I and D gains are used in this project. These coefficients may be changed by the user if better control loop performance is desired.
28. Close the boost stage voltage loop by setting *closeloop_boost* variable to 1 in the watch window. You may now set the desired DC bus voltage directly by using *Gui_VdcBusSet* variable in the watch window. An important thing to note here is that there is slew limit implemented in the software on this *VdcBus* reference command. Once the loop is closed and the first time the *Gui_VdcBusSet* value is changed from 0 to a value greater than *Gui_VdcBus*, the reference command to the controller will slowly ramp up from 0 to this value. Therefore it will take a good amount of time for the *VdcBus* to reach your set reference command. Optionally, you may change the *boostSetSlewed* and the *boostSlewRate* variables from the watch window to make this process faster. Do not make the *Gui_VdcBusSet* value greater than 12V.
29. At this stage you may observe the effect of varying the input voltage at J2. There should be virtually no effect on the *Gui_VdcBus* for modest variations in the input voltage.

30. At this stage you may try to increase the inverter output voltage to 5V using the command *Vrms* in the watch window. As you increase the inverter output voltage command you may have to increase the amount of boost so as to make a higher DC bus voltage available at the inverter input. With the lamp load connected at the inverter output, limit the inverter output voltage below 5V. For other loads make sure the output voltage and current levels do not exceed the rated values. As you increase the inverter output voltage the output voltage waveform on the graph should start looking better.

Battery Charging Stage

31. Notice that the *buck_cmd* variable is set to 299 (499 for F280x) by default. This directly controls the compare value for the PWM driving the battery charging stage and thus controls the duty cycle and the amount of buck imparted. A value of 499 implies zero duty cycle while a value of 0 implies 100% duty cycle.
32. In the watch window, the variable *Charging_EN* should be set to 0. This variable enables the PWM drive to battery charging stage. With *buck_cmd* set to 299 in the watch window, set *Charging_EN* = 1.
33. With the *closeloop_buck* set to 0, we will be operating in open loop and can directly control the amount of buck using *buck_cmd*. The software clamps the maximum duty possible by clamping minimum allowable *buck_cmd* to a value of 120.
34. Observe the change at the voltage at J6 by observing *Gui_Vbatt* in the watch window for different the *buck_cmd* values. Below a certain *buck_cmd* value the lamp load at J6 will light up. Also notice that *Gui_Ibatt* also changes with different *buck_cmd* values. Change the *buck_cmd* back to 299 before going to the next step.
35. The battery charging stage could operate in one of two modes: constant voltage (CV) charging, and constant current (CC) charging. According to the mode selected either of the voltage loop or the current loop is closed and controls the output. As discussed in the previous chapter, this software uses two pole two zero controller as the control loop with its 5 coefficients mapped to P, I and D coefficients for intuitive tuning. Very low P, I and D gains are used in this project. These coefficients may be changed by the user if better control loop performance is desired.
36. Close the battery charging voltage loop by setting *CV_Charging* variable to 1 in the watch window. You may now set the desired output voltage directly by using *Gui_VbattSet* variable in the watch window. An important thing to note here is that there is slew limit implemented in the software on this Vbatt reference command. Once the loop is closed and the first time the *Gui_VbattSet* value is changed to a non-zero value, the reference command to the controller will slowly ramp up from 0 to this value. Therefore it will take a good amount of time for the Vbatt to reach your set reference command. Optionally, you may change the *buckVSetSlewed* and the *boostVSlewRate* variables from the watch window to make this process faster. With the lamp load connected at J6, do not make the *Gui_VbattSet* value greater than 6V.
37. At this stage you may observe the effect of varying the input voltage at J2. There should be virtually no effect on the *Gui_Vbatt* value for modest variations in the input voltage.

38. Change the *Gui_VbattSet* to 0 and wait for *Gui_Vbatt* to go close to 0 before going to the next step.
39. Take the battery charging out of the voltage loop by changing *CV_Charging* variable to 0 in the watch window. Now close the battery charging current loop by setting *CC_Charging* variable to 1 in the watch window. You may now set the desired output charging current directly by using *Gui_IbattSet* variable in the watch window. An important thing to note here is that there is slew limit implemented in the software on this Ibatt reference command. Once the loop is closed and the first time the *Gui_IbattSet* value is changed to a non-zero value, the reference command to the controller will slowly ramp up from 0 to this value. Therefore it will take a good amount of time for the Ibatt to reach your set reference command. Optionally, you may change the *buckISlewRate* and the *boostISlewRate* variables from the watch window to make this process faster. With the lamp load connected at J6, do not make the *Gui_IbattSet* value greater than 0.4A. For Ibatt values close to 0.8/1A, the sensed Ibatt value is closer to the actual value.
40. Change the *Gui_IbattSet* to 0 and wait for *Gui_Vbatt* to go close to 0 before going to the next step.
41. Change *start* and *Charging_EN* to 0 in the watch window. Both lamp loads must now be turned off. Turn Off SW4.
42. Fully halting the controller when in real-time mode is a two-step process. With the DC supply at J2 shut off wait a few seconds. First, halt the processor by using Shift <F5>, or using the Halt button on the vertical toolbar, or by using Debug → Halt. Then click Debug → Real-time Mode and uncheck the “Real-time mode” to take the MCU out of real-time mode and then reset the MCU.
43. You may choose to leave Code Composer Studio running or optionally close CCS.
44. This software allowed the user to decide which loop to operate in. Alternatively, an example charging algorithm may be automated to switch the control from a CC charging mode to a CV charging mode. Such an example for charging a 12V battery is shown in the file “Renewable-Main_batt_algo.c”. This file can be used to replace the “Renewable-Main.C” file in this software project to run the complete battery charging algorithm.

Note: By default JP8 has not been populated and doesn't need to be for the above procedure. However, if a better transient response is desired and to support higher loads at the DC-DC buck converter output, it may be required to populate this jumper.

End of Exercise

References

For more information please refer to the following guides:

- **Renewable** – provides detailed information on the Renewable energy kit.
C:\TI_F28xxx_SysSW\ Renewable \~Docs\ Renewable.pdf
- **Renewable-Calculations, Renewable-Piccolo-Calculations.xls** – a spreadsheet showing some of the key calculations made within the *Renewable* project.
C:\TI_F28xxx_SysSW\ Renewable \ Renewable -Calculations.xls
C:\TI_F28xxx_SysSW\ Renewable \ Renewable -Calculations-Piccolo.xls
- **QSG-Renewable-Piccolo-GUI** – gives an overview on how to quickly demo the Renewable Energy project using an intuitive GUI interface.
C:\TI_28xxx_SysSW\ Renewable\~Docs\QSG-Renewable-GUI.pdf
- **Renewable-HWdevPkg** – a folder containing various files related to the hardware on the Renewable energy Kit board (schematics, bill of materials, Gerber files, PCB layout, etc).
C:\TI_F28xxx_SysHW\Renewable-HWdevPkg
- **System Framework Overview Guide** – presents more information on the system framework found in F28xxx EVM projects.
C:\TI_F28xxx_SysSW\~Docs\SystemFrameworkOverview.pdf
- **F28xxx User's Guides**
<http://www.ti.com/f28xuserguides>