

# **PMP22951 Peak Current Mode Controlled (PCMC) Phase Shifted Full-Bridge DC-DC Converter**

---

## **Important Notice**

Texas Instruments and its subsidiaries (TI) reserve the right to make changes to their products or to discontinue any product or service without notice, and advise customers to obtain the latest version of relevant information to verify, before placing orders, that information being relied on is current and complete. All products are sold subject to the terms and conditions of sale supplied at the time of order acknowledgment, including those pertaining to warranty, patent infringement, and limitation of liability.

TI warrants performance of its semiconductor products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

Customers are responsible for their applications using TI components.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards must be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance or customer product design. TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used. TI's publication of information regarding any third party's products or services does not constitute TI's approval, warranty or endorsement thereof.

Copyright © 2019 Texas Instruments Incorporated

---

# 1 Introduction

Phase shifted full bridge (PSFB) DC/DC converters are frequently used to generate isolated +12V, +24V, +48V or +54V DC output from a regulated 400V input in medium- to high-power range in server power, telecom rectifiers, EV/HEV battery charging, and renewable energy applications. A PSFB converter consists of four power switches (MOSFETs or GaNs) forming a full bridge on the primary side of the isolation transformer and a diode rectifier or MOSFET/GaN switches for synchronous rectification (SR) on the secondary side. The secondary switches can be in a full-bridge or a half-bridge current-doubler configuration depending on the power level. This PSFB topology allows for zero voltage switching (ZVS) operation for all switching devices at high output power and results in lower switching losses and higher efficiency. Moreover, ZVS for switches in the right leg (also known as lagging leg) of the full bridge and zero/low voltage or valley switching for switches in the left leg (also known as leading leg) is achieved over a wide load range by changing dead-times for primary side switches based on load conditions. For such an isolated topology, signal rectification is required on the secondary side. For systems with low output voltage and/or high output current ratings, implementing synchronous rectification instead of diode rectification achieves the best possible efficiency by avoiding diode rectification losses. If the power output is low a current doubler sync rectification is a good choice. But for higher power DC/DC with higher output voltages, such as +54V output, a full bridge (FB) synchronous rectification is implemented on the secondary side with different switching schemes or modes to achieve optimum performance under varying load conditions. Figure 1 shows the power stage for such a PSFB DC/DC converter. Q1 and Q2 form the leading leg while Q3 and Q4 form the lagging leg.

A PSFB DC/DC converter system can be controlled in various modes like voltage mode control (VMC), average current mode control (ACMC) or peak current mode control (PCMC). In this design PCMC control schemes (Figure 1) is used to implement control of a PSFB DC/DC with a FB secondary synchronous rectification. PCMC is a highly desired control scheme for power converters because of its inherent voltage feed forward, automatic cycle-by-cycle current limiting, flux balancing and other advantages. For such a PCMC PSFB DC/DC converter, MOSFETs output capacitance  $C_{oss}$  oscillates with  $L_o$  and  $L_r$  causing ringing across the output rectifiers. This requires MOSFETs with higher  $V_{DS}$  rating and, therefore, results in higher  $R_{ds,on}$  and higher power losses. This issue becomes severe in 54V<sub>out</sub> system. Therefore, an active clamping circuit (ACL) across the secondary rectifiers is required to limit the ringing across the output rectifiers allowing the use of lower voltage devices for best possible efficiency. In Figure 1 this ACL circuit is placed across the output rectifiers and consists of a clamp capacitor in series with a MOSFET. This ACL circuit across the output rectifiers, in turn, causes primary peak current distortion and poses additional challenges in peak current mode control implementation. This issue is addressed and discussed in detail in this design.

In this implementation the complete system is controlled using Texas Instruments 32-bit microcontroller TMS320F280049, which is placed on the secondary side of the main isolation transformer. This system implements advanced control strategies to optimally control the power stage under different conditions and also provide system level intelligence to make safe and seamless transitions between operation modes and PWM switching patterns.

$V_{12} = N \cdot V_{34}$   
 $N = N_p / N_s$   
 $V_{12} = V_1 - V_2$   
 $V_{34} = V_3 - V_4$

TMS320C2000™ Systems Applications Collateral

## 1.1 Basic Operation

Figure 2 shows a simplified block diagram of the PCMC PSFB DC/DC control circuit implemented using TMS320F280049 MCU. There are four MOSFET or GaN switches (Q1-Q4) that form a full-bridge (FB) on the high voltage (HV) side of the isolation transformer T1. Four MOSFET or GaN switches (Q5-Q8) on the secondary low voltage (LV) side work as synchronous rectifiers (SR). ACL PWM drives the ACL MOSFET Q9.

This topology allows all the high voltage switching devices to switch with zero voltage switching (ZVS) resulting in lower switching losses and higher efficiency.

Q<sub>1</sub> and Q<sub>2</sub> form the leading leg of the primary bridge. The PWM signals driving these switches are set to 50 % duty with a constant 180-degree phase shift between them. Similarly, the PWM signals driving the lagging leg consisting of Q<sub>3</sub> and Q<sub>4</sub> also maintain 50 % duty and 180 degree phase shift with each other. However, the PWM signals for leg Q<sub>3</sub> – Q<sub>4</sub> of the full-bridge are phase shifted with respect to those for leg Q<sub>1</sub> – Q<sub>2</sub>. This phase shift is set by the desired peak current in the primary side. Thus, the phase shift decides the amount of overlap between diagonal switches, which in turn decides the amount of energy transferred from primary side of the transformer to the secondary side. Four switches Q<sub>5</sub> - Q<sub>8</sub> provide synchronous rectification (SR) on the secondary, while L<sub>o</sub> and C<sub>o</sub> form the output filter.

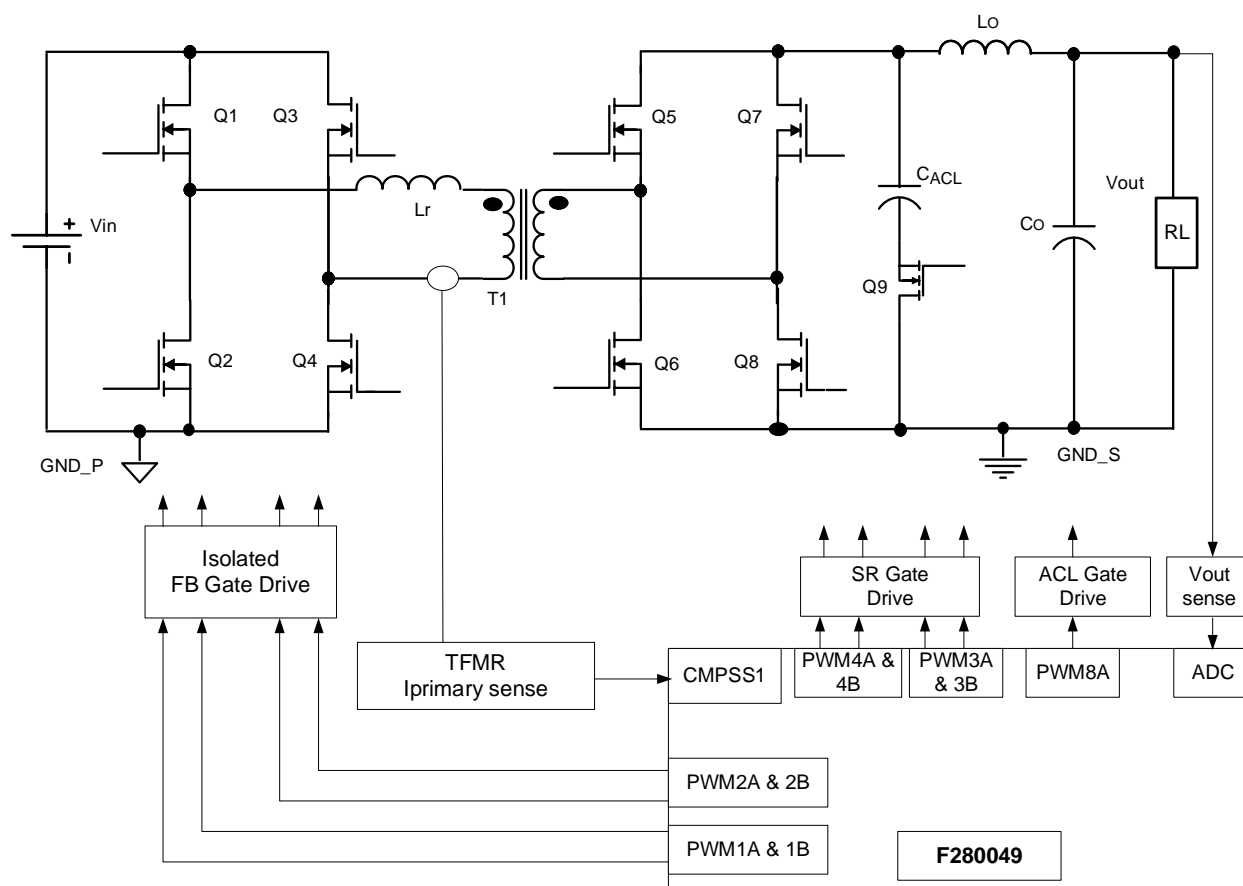
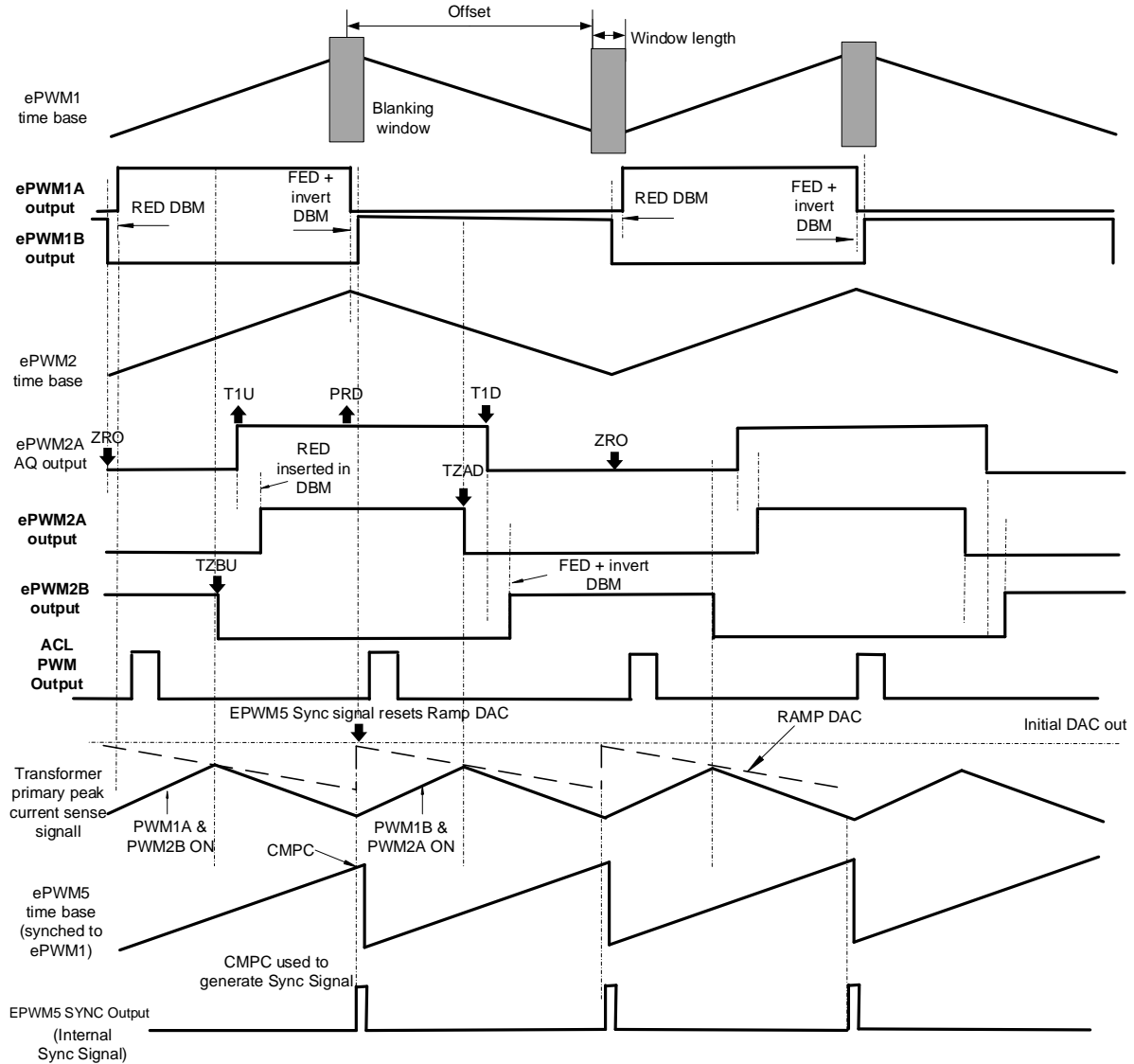
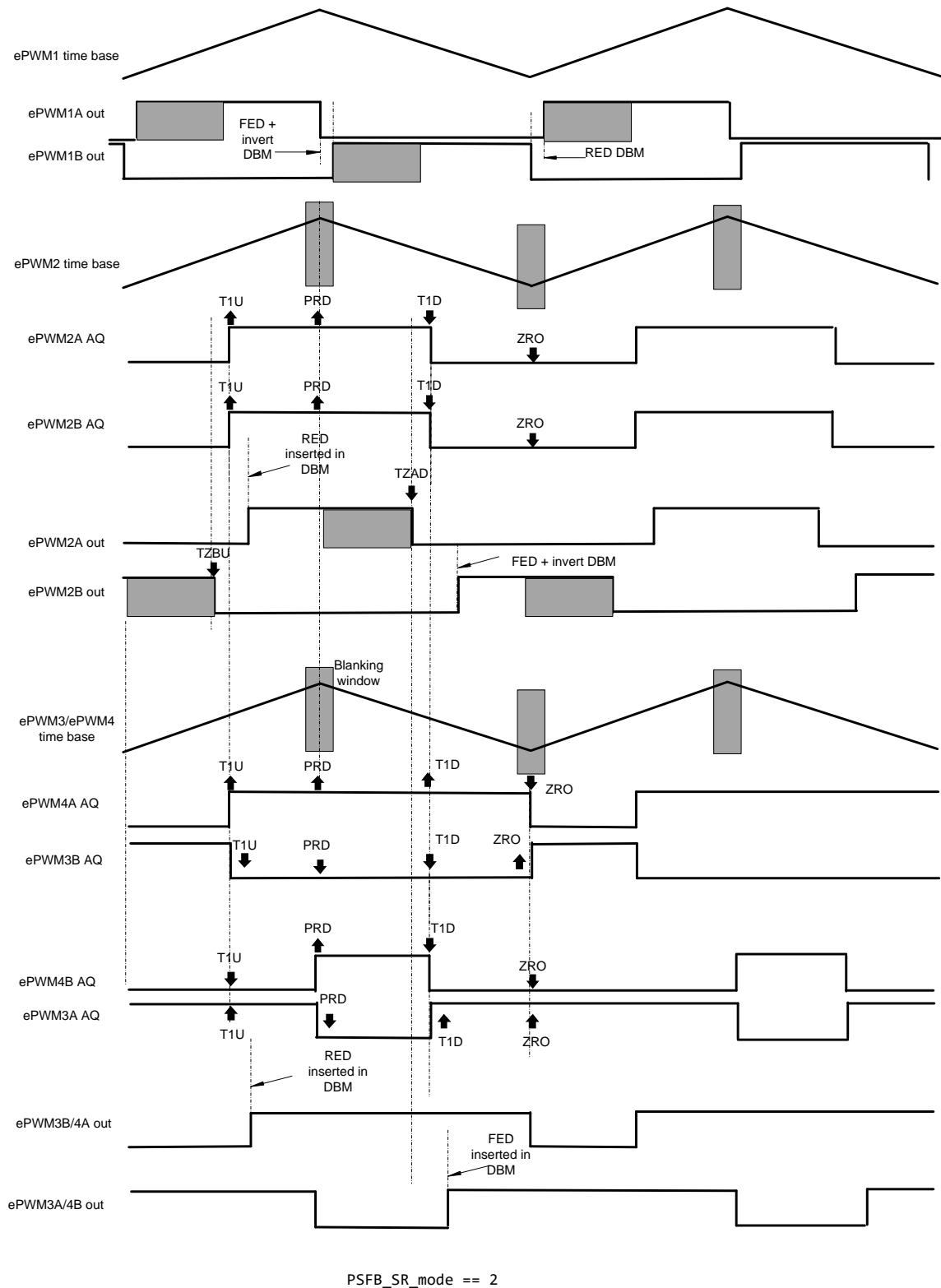


Figure 2 - F280049 Interface to PCMC PSFB DC/DC Power Stage

A small inductor  $L_r$  on the primary side acts as shim inductor and adds to the transformer leakage inductance. This helps maintain ZVS operation of the HV switches (Q1 - Q4) under light load conditions. ACL PWM driving the ACL switch Q9 runs at twice the PWM frequency of the other eight PWM signals. This ACL PWM also maintains a fixed pulse width and a fixed delay with respect to the leading leg PWM signals PWM1A and PWM1B. When the ACL switch Q9 turns on the clamp capacitor limits the ringing across the output rectifiers when they are turned off. Figure 3, 4 and 5 provide all the PWM waveforms and peak primary current waveforms for the system in Figure 2.



**Figure 3 - PCMC PSFB DC/DC primary side PWM, ACL PWM and the primary peak current**



**Figure 4 - PCMC PSFB DC/DC primary and secondary side PWM signals for SR Mode 2**

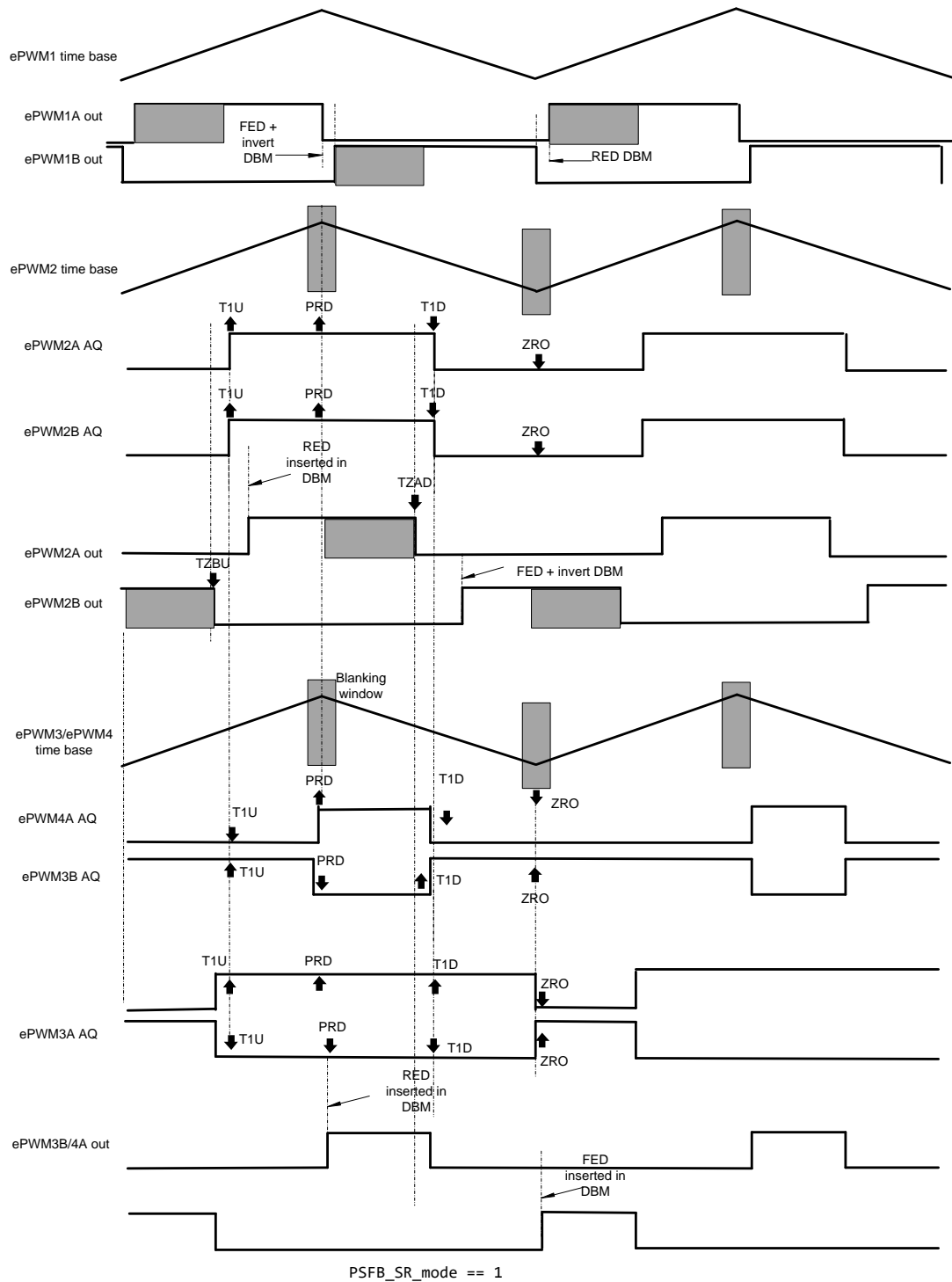


Figure 5 - PCMC PSFB DC/DC primary and secondary side PWM signals for SR Mode 1

## 1.2 Implementation on TIDM-xxxxxx Board

This PCMC PSFB DC/DC power supply is implemented on a TI reference design TIDM-xxxxx board. The control algorithm for this design is implemented using a C2000 microcontroller (MCU) TMS320F280049. The MCU, placed on the secondary side of the isolation transformer, senses the peak primary current, the output voltage and other auxiliary signals. Based on the sensed signals the MCU then implements full digital control and generates all the PWM signals necessary for controlling this PCMC PSFB DC/DC system. Controlling such a DC/DC system optimally for fast load transient response and high efficiency over a wide load current range requires complex PWM signal generation along with fast and efficient control loop calculations. This is made possible on C2000 MCU by advanced on-chip control peripherals like PWM modules, analog comparators with DAC and slope compensation hardware, and 12-bit high speed ADCs coupled with an efficient 32-bit CPU. A detailed description of the software algorithm is provided in the following chapters.

## 1.3 Hardware Specifications

Following lists the key specs for this implementation.

- Input voltage range 350V DC ~ 420V DC
- Output voltage 54V DC. Rated output power 3kW
- Topology: Peak current mode control phase shifted full-bridge with outer voltage loop
- PWM frequency: 140 kHz
- Efficiency: 97.45% peak.
- Multiple Synchronous Rectification (SR) switching schemes for optimum efficiency
- Fault protection: Over-current, over-voltage



## 2 Power Stage and MCU Interface Signals

The key signal connections between the MCU and the power board are listed in the table below.

Power stage Signal	Description	MCU Pin (TMS320F280049)
LEG1_HS	PWM drive for full-bridge switch Q1	GPIO-00/PWM1A
LEG1_LS	PWM drive for full-bridge switch Q2	GPIO-01/PWM1B
LEG2_HS	PWM drive for full-bridge switch Q3	GPIO-02/PWM2A
LEG2_LS	PWM drive for full-bridge switch Q4	GPIO-03/PWM2B
LEG3_HS	PWM drive for sync-rectifier switch Q5	GPIO-04/PWM3A
LEG3_LS	PWM drive for sync-rectifier switch Q6	GPIO-05/PWM3B
LEG4_HS	PWM drive for sync-rectifier switch Q7	GPIO-06/PWM4A
LEG4_LS	PWM drive for sync-rectifier switch Q8	GPIO-07/PWM4B
PWM_ACL	PWM drive for ACL switch Q9	GPIO-14/PWM8A
ORING_EN	Enable signal for ORing switch	GPIO-13/PWM7B
VLV-FB	54V Output sense signal <b>before</b> the ORing FET	A5
VLV-FB1	54V Output sense signal <b>after</b> the ORing FET	A3
ILV-FB	54V DC output current sense signal	B1
IHV_FB	Transformer primary peak current signal	COMP1 (A2/B6/PGA1)

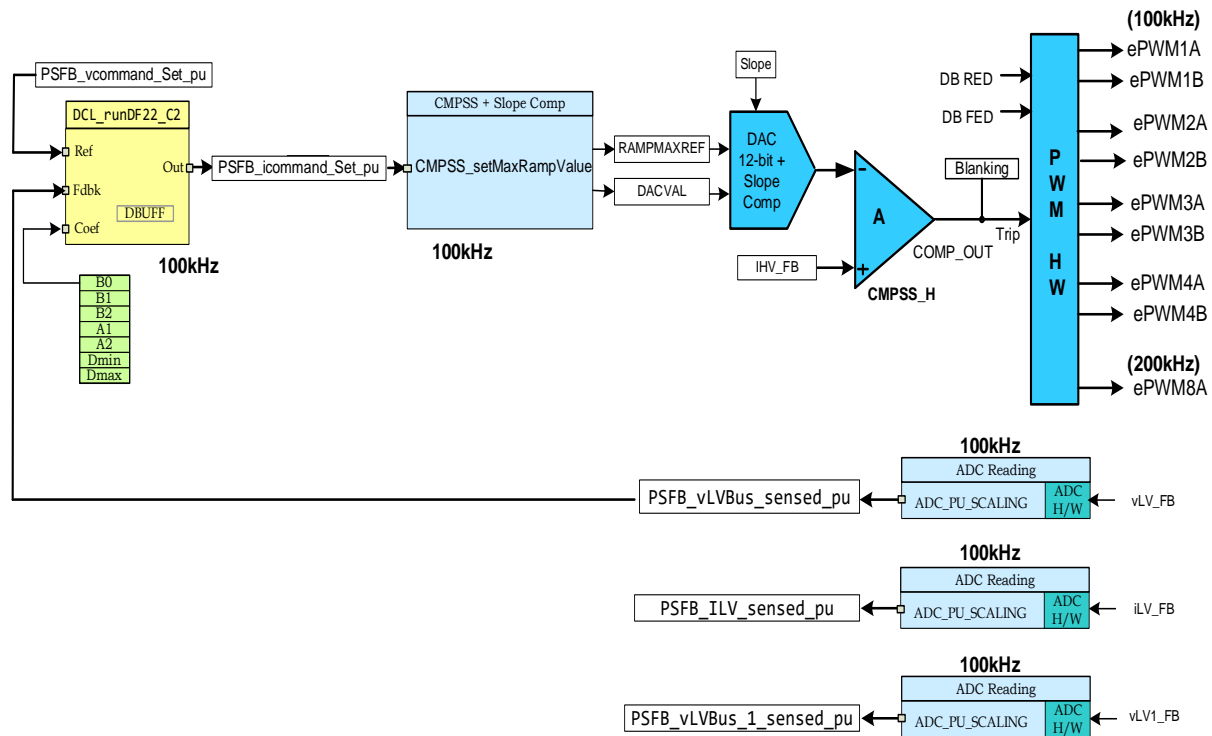
## 3 Functional Description

### 3.1 Peak Current Mode Control (PCMC)

Implementing PCMC for an isolated 48V or 54V output PSFB DC/DC converter requires complex PWM waveform generation with precise timing control. The F28004x family of devices from Texas Instruments feature advanced on-chip control peripherals that make this implementation possible without any external support circuitry. These peripherals include on-chip analog comparators (CMPSS) with blanking window and integrated digital to analog converters (DAC), advanced PWM modules and unique programmable on-chip slope compensation hardware. Figure 6 shows the software control block diagram for such a PCMC PSFB DC/DC converter using F28004x series MCU. The dark blue blocks represent functionality implemented by MCU on-chip hardware. Transformer primary current sense signal (IHV\_FB) is compared with the peak current reference using the on-chip comparator 1. The peak current reference is calculated by the voltage loop controller and applied to the comparator through the DAC. As shown in Figure 3, in every half

PWM period when the transformer primary current reaches the commanded peak reference value, one of the primary PWM signals PWM2A or PWM2B, driving the switches Q3 or Q4 respectively, is 'Reset' immediately ending the power transfer phase. Following this the PWM signals driving the other switch in the same leg is 'Set' after a programmable dead-time (dead-band) window. Appropriate slope compensation is also applied in the DAC that adds a ramp with a programmable negative slope to the peak reference current signal calculated by the voltage loop controller. This is also shown in Figure 3 and indicated as RAMP DAC signal. The 'Resetting' and 'Setting' action of the PWMs in one leg results in a phase shift between PWM signals driving the two legs. The amount of this phase shift, and hence the overlap between diagonal switches (Q1/Q4 pair or Q2/Q3 pair), is dependent on the commanded peak current reference. Higher the peak reference current, longer the overlap between diagonal switches, and higher the energy transferred to the secondary. The voltage loop controller regulates the output voltage by controlling this energy transfer by way of commanding the appropriate peak reference current and, thereby, controlling the actual peak current in the transformer primary.

An important feature of this implementation is that the same peak reference current command is used for both halves of the PWM switching cycle under all operating conditions. This provides optimal flux balancing for the transformer.



**Figure 6 : Software Control Diagram for PSFB PCMC DC/DC using F28004x MCU**

## 3.2 Slope Compensation

The peak current reference command is written to the C2000 MCU register RAMPMAXREFS which is the starting value of the RAMP used for slope compensation. The slope is decided by the value written in the register RAMPDACVALS. The ramp generator produces a falling-ramp input for the high reference 12-bit DAC when selected. Note that in this mode, the reference 12-bit DAC uses the most significant 12 bits of the RAMPSTS countdown register as its input. The low 4 bits of the RAMPSTS countdown register effectively act as a prescaler for the falling-ramp rate configurable with RAMPDECVALA. The slope value is chosen to ensure the fastest inner peak current control loop, and hence a faster outer voltage control loop, while eliminating any possibility of subharmonic oscillations.

The slope value used in such a DC/DC design using C2000 MCU can be explained in the following way.

For the C2000 MCU the internal DAC is 12bit. But the actual DAC input register (RAMPMAXREFA) is 16bit. So the max input value for the DAC is 0xFFFF = 65535. This corresponds to 3.3V at the DAC output. For a CPU clock frequency ( $f_{CPU}$ ) of 100MHz and a slope value of  $m$ , the DAC output voltage will drop at a rate of  $R$ , where  $R$  is given by,

$$R = m \cdot \{(3.3/65535)/(1/f_{CPU})\} = m \cdot \{(3.3/65535)/(1/100 \times 10^6)\} \text{ volts/Sec}$$

If  $m = 20$  ( $m$  can be any value between 0 to 255), then,

$$R = 20 \cdot \{(3.3/65535)/(1/100 \times 10^6)\} = 100710 \text{ volts/Sec} = 0.10071 \text{ volts/uS}.$$

For a specific load power if the ON time (power transfer time) is  $T_d$  (in uS) in every half of the primary side PWM cycle ( $f_{PWM}$ ), then assuming the sampling cycle the same as the PWM cycle, the DAC value will decrease from its initial written value by an amount of  $\Delta V = (R) \cdot (T_d)$

If the measured peak current sense voltage is  $V_{cs}$ , then the commanded DAC value (from the voltage controller output written into the DAC input) represents a voltage of:

$$U = V_{cs} + \Delta V$$

Because of slope compensation this value decreases by  $\Delta V$  during the ON time  $T_d$  and the final DAC output matches the peak feedback current sense voltage  $V_{cs}$ .

## 3.3 Zero Voltage Switching (ZVS)

PSFB DC-DC converters make use of parasitic elements in the circuit to ensure zero voltage across the MOSFET switches before turning them ON, providing soft switching. This considerably reduces the amount of switching losses associated with hard switching.

For the system discussed here, switching transitions for switches in the Q3-Q4 leg end the power transfer interval. Therefore, this leg is also called as 'Active to Passive' leg. When transitions occur for switches in this leg, current in the primary winding is close to its maximum magnitude for that half PWM switching cycle. The reflected load current aids the circulating energy in the

primary circuit during this time, which makes it possible for voltage across switches in this leg to approach zero volts. It is possible to achieve ZVS for switches in this Q3-Q4 leg across the complete load range. It should be noted that as the load decreases the amount of dead-time needs to be increased to achieve/approach ZVS.

Switching transitions for switches in the Q1- Q2 leg start the power transfer interval. Therefore, this leg is also called as 'Passive to Active' leg. During these switching transitions, primary current decreases. It crosses zero current value and changes direction. This results in lower available energy for ZVS. In fact, for operations under low load conditions, voltage across these switches may not go to zero before turning them on. Switching losses can be kept to a minimum by turning these switches ON at a time when the voltage across them is at a minimum. This is called valley switching or low voltage switching (LVS). As the load current changes, the time at which these switches should be turned on to achieve LVS also changes. This requires dead-time adjustment similar to the other Q3-Q4 leg switches.

### 3.4 Synchronous Rectification

Synchronous rectifiers (SR) can work in one of the following three modes at any given time:

- I. SR Mode 0: This is the classical diode rectification mode achieved by keeping the synchronous rectifiers in OFF state. It is enabled under light load condition where synchronous rectifier switching losses are greater than the power savings obtained by synchronous rectification. This load range could be 5 to 15% of rated load.
- II. SR Mode 1: In this mode the synchronous rectifier switches behave like ideal diodes. This mode is enabled for light load to moderately higher load range, say, 15% to 55% of rated load. In this mode, synchronous rectifier MOSFETs are turned ON only when the corresponding diagonal bridge PWM drive signals on the primary side overlap to initiate a power transfer phase.
- III. SR Mode 2: This mode is useful for all other higher load conditions, say 55% to 100% of rated load. In this mode, appropriate synchronous rectifier MOSFETs are turned OFF only when the corresponding opposite diagonal bridge drive signals overlap.

Figure 4 and Figure 5 depict the PWM waveforms generated for driving the synchronous rectifier switches in these modes. To ensure safe operation of the system it is important to implement SR mode transitions seamlessly without any glitches or anomalies on the PWM outputs even during large load transients or sudden phase shift change commands.

## 4 Software overview

### 4.1 Software Control Flow

Once the software project for PCMC PSFB DC/DC is imported in CCS, the Project Explorer will appear inside CCS as shown in Figure 7.

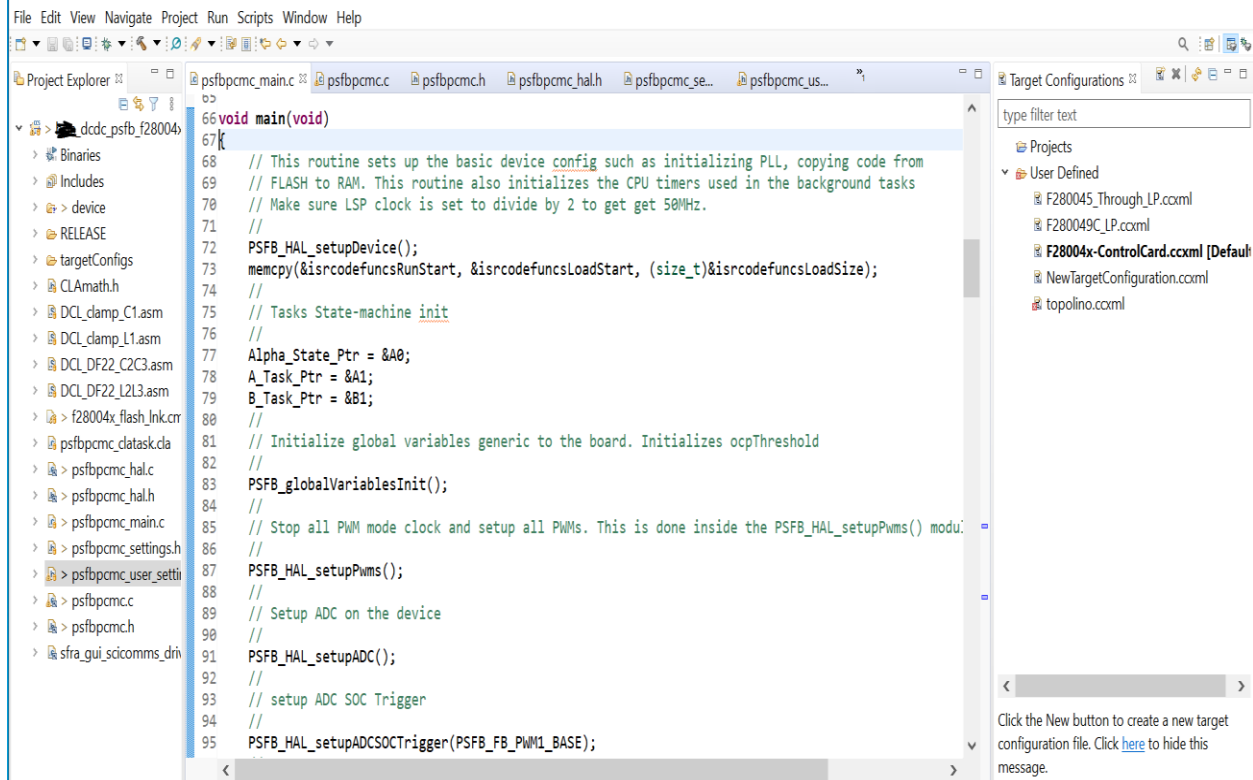


Figure 7 – Project Explorer view in CCS

Solution-specific and device-independent files that consist of the core algorithmic code are in the files *psfbpcmc.c* and *psfbpcmc.h*. Board-specific and device-specific files are in *psfbpcmc\_hal.c* and *psfbpcmc\_hal.h*. These files consist of device-specific drivers (specific to MCU used in this project) to run the solution. If the user wants to use a different modulation scheme (board specific) or a different MCU (device specific), the user is required only to make changes to these files, besides changing the device support files in the project. The *psfbpcmc\_main.c* file consists of the main framework of the project. This file consists of calls to the board and solution files that help in creating the system framework, along with the interrupt service routines (ISRs) and slow background tasks. Two settings files *psfbpcmc\_settings.h* and *psfbpcmc\_user\_settings.h* are used in compiling the project. The file *psfbpcmc\_user\_settings.h* is included by the file *psfbpcmc\_settings.h* and can be used to keep any settings related to ADC/PWM channel mapping, GPIOs etc. The solution name is used as the module name for all the variables and defines used in the solution. Hence, all variables and function calls are prepended by the label PSFB (for example, PSFB\_vLVBus\_sensed\_pu). This naming convention lets the user combine different solutions while avoiding naming conflicts.

This project consists of two interrupt service routines (ISR), ISR1 and ISR2, with ISR2 being the fastest and non-nestable ISR. ISR2 runs at  $fs_2$ , where  $fs_2 = 180\text{kHz}$ . This is reserved for capturing the Timer counter value when the comparator trip events (T1U and T1D) are generated. ISR2 is triggered when a PWM trip zone interrupt caused by DCAEVT2 (Digital Compare B Event 2 interrupt) happens for EPWM2. The source of DCAEVT2 is the comparator events T1U and T1D. ISR1 is triggered by the PSFB\_FB\_PWM1\_BASE→EPWM\_INT\_TBCTR\_ZERO event and runs at  $fs_1$ , where  $fs_1=70\text{kHz}$ . Figure 8 shows the software diagrams and structure of ISR2 and ISR1. ISR1 is responsible for running the voltage loop digital controller, calculating the reference current command for the peak current loop, controlling SR mode switching and controlling frequency folding under light load condition.

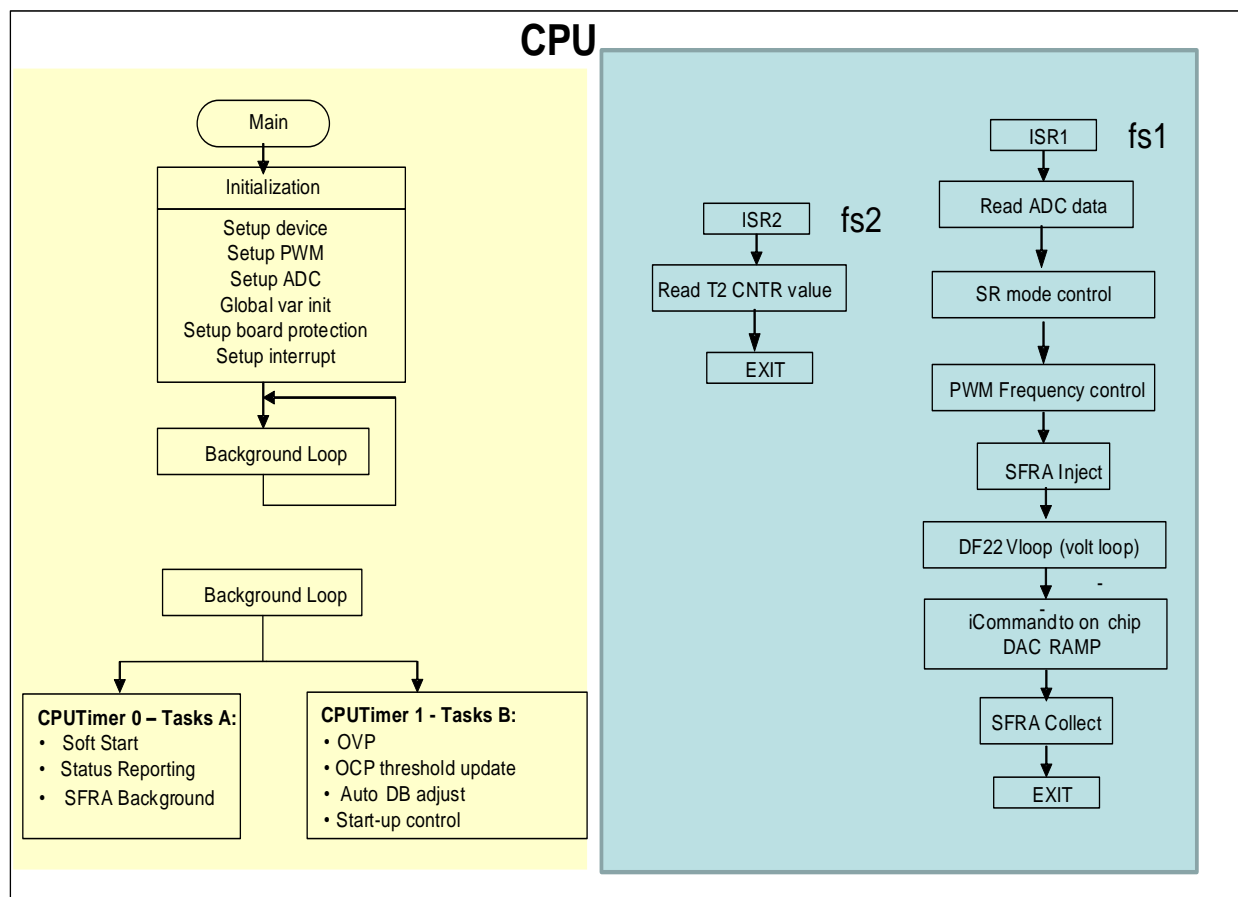


Figure 8 – Software Structure for main, ISRs and background loops

For a closed-loop system under steady-state, the peak current will only have minor changes from one control ISR cycle to the other and hence the ISR2 will be triggered repeatedly.

Additionally, CPU timers are used to trigger slow background tasks. These tasks are not interrupt-driven but polled. Background tasks includes housekeeping functions such as doing soft-start, implementing overvoltage protection, running auto dead-band (DB) adjustment etc.

CPU Timer0 is used for "A" tasks, triggered at TASKA\_FREQ of 2000 Hz. CPU Timer1 is used for task "B" triggered at TASKB\_FREQ of 20 Hz. CPU Timer2 is used for task "C" triggered at TASKC\_FREQ of 1000 Hz. These task frequency parameters are defined in the file *psfbpcmc\_user\_settings.h* file. The function calls corresponding to all of these tasks can be found inside the file *psfbpcmc\_main.c*

## 4.2 Incremental Builds

The software for this reference design is organized in two labs, with incremental builds (INCR\_BUILD) options. These labs allow step-by-step design validation and simplify system bring-up. Lab 1 is used for open voltage loop peak current mode control (PCMC) check of PWM and ADC drivers with protection enabled. For Lab 1 resistive load should be connected across output terminals. Lab 2 is used for closed voltage loop PCMC operation with soft-start, automatic synchronous rectifier (SR) mode switching, automatic dead-band (DB) adjustment and automatic frequency folding at light load.

To select a particular build option, select the corresponding PSFB\_INCR\_BUILD option in *psbpcmc\_settings.h* as shown below and explained in Table 1.

```

psfbpcmc_settings.h
22//*****
23//defines
24//*****
25
26/* Device Related Defines */
27#define PSFB_CPU_SYS_CLOCK_HZ (100*1000000)
28#define PSFB_PWMSYSCLOCK_FREQ_HZ (100*1000000)
29#define PSFB_ECAPSYSCLOCK_FREQ_HZ (100*1000000)
30
31/* Project Options*/
32//=====
33// Incremental Build options for System check-out
34//=====
35//
36// BUILD 1   Open Voltage Loop Check (Peak current loop, in MCU internal h/w, is always closed)
37// BUILD 2   Closed Voltage Loop Check
38//
39#define PSFB_INCR_BUILD 1
40#define PSFB_PCMC 1
41
42// 1 means control runs on C28x , otherwise it will run on CLA
43#define C28x_CORE 1
44#define CLA_CORE 2
45#define PSFB_CONTROL_RUNNING_ON 1
46

```

Incremental Build Options	
PSFB_INCR_BUILD = 1	Check open voltage loop operation with PCMC (check PWM drive circuit, ADC sensing circuit and PCMC operation).
PSFB_INCR_BUILD = 2	Check closed voltage loop operation with PCMC (Full closed loop operation)

Table 1 – PCMC PSFB DC/DC incremental build options

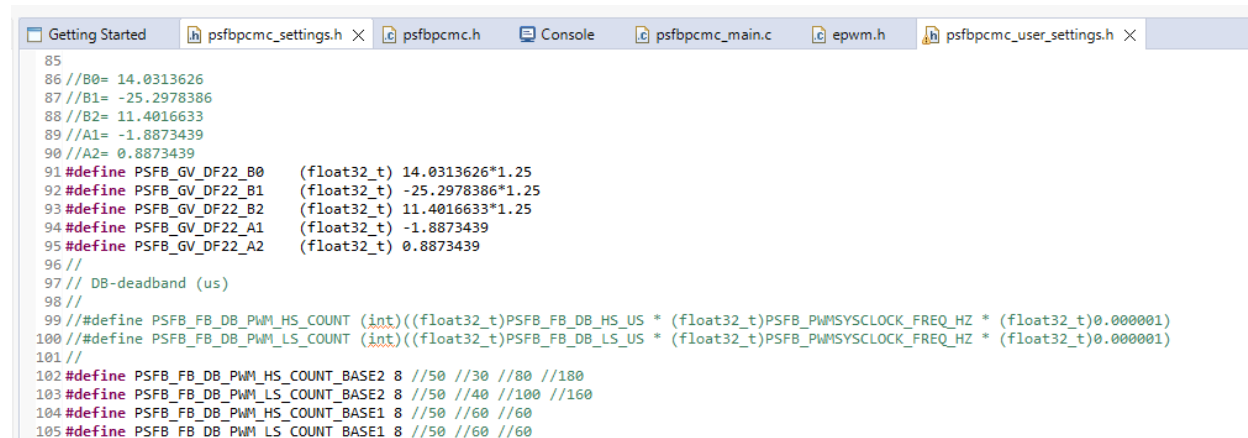
## 5 Procedure for running the incremental builds

### 5.1 Lab 1

The tests under Lab 1 are run using the PCMC PSFB h/w and a F280049 control card. Both the over-voltage protection (OVP) and the over-current protection (OCP) are enabled.

The software in Lab1 has been configured so that the user can quickly evaluate how the on-chip comparator and the on-chip DAC with integrated slope compensation ramp work together to implement PCMC. This lab allows the user adjust the peak current reference command and the slope compensation (two inputs to the DAC) from CCS real-time (RT) watch window and then view various waveforms on an oscilloscope and observe the effect of this change on the output voltage. Additionally, the user can evaluate the ADC software driver module by viewing the ADC sampled data in the watch window.

The on-chip analog comparator (CMPSS) compares the transformer primary current sense voltage signal with the slope compensated peak current reference from the DAC output. Comparator output is connected to the trip zone logic of the ePWM modules. ePWM1 module acts as the master time-base for the system. It operates in up-down count mode. ePWM1A and ePWM1B drive Q1 and Q2 full-bridge switches, while ePWM2A and ePWM2B drive Q3 and Q4 full-bridge switches. ePWM3A, ePWM3B, ePWM4A and ePWM4B drive Q5, Q6, Q7 and Q8 synchronous rectifier (SR) switches respectively. Whenever the comparator output goes high in a PWM half cycle, the ePWM2 module output (ePWM2A or ePWM2B), which was high at that instant, is immediately pulled low (turned OFF) while the other PWM2 module output is pulled high (turned ON) after an appropriate dead-time defined by the dead-band (DB) register values. The initial dead-time values for PWM2 module outputs are defined by the constants `PSFB_FB_DB_PWM_HS_COUNT_BASE2` and `PSFB_FB_DB_PWM_LS_COUNT_BASE2` in the `psfbpcmc_user_settings.h` file. ePWM1A and ePWM1B also maintain some dead-time between them and their initial values are defined by `PSFB_FB_DB_PWM_HS_COUNT_BASE1` and `PSFB_FB_DB_PWM_LS_COUNT_BASE1` in the `psfbpcmc_user_settings.h` file as shown below.



```
85
86 //B0= 14.0313626
87 //B1= -25.2978386
88 //B2= 11.4016633
89 //A1= -1.8873439
90 //A2= 0.8873439
91 #define PSFB_GV_DF22_B0 (float32_t) 14.0313626*1.25
92 #define PSFB_GV_DF22_B1 (float32_t) -25.2978386*1.25
93 #define PSFB_GV_DF22_B2 (float32_t) 11.4016633*1.25
94 #define PSFB_GV_DF22_A1 (float32_t) -1.8873439
95 #define PSFB_GV_DF22_A2 (float32_t) 0.8873439
96 //
97 // DB-deadband (us)
98 //
99 // #define PSFB_FB_DB_PWM_HS_COUNT (int)((float32_t)PSFB_FB_DB_HS_US * (float32_t)PSFB_PWM_SYSCLOCK_FREQ_HZ * (float32_t)0.000001)
100 // #define PSFB_FB_DB_PWM_LS_COUNT (int)((float32_t)PSFB_FB_DB_LS_US * (float32_t)PSFB_PWM_SYSCLOCK_FREQ_HZ * (float32_t)0.000001)
101 //
102 #define PSFB_FB_DB_PWM_HS_COUNT_BASE2 8 //50 //30 //80 //180
103 #define PSFB_FB_DB_PWM_LS_COUNT_BASE2 8 //50 //40 //100 //160
104 #define PSFB_FB_DB_PWM_HS_COUNT_BASE1 8 //50 //60 //60
105 #define PSFB_FB_DB_PWM_LS_COUNT_BASE1 8 //50 //60 //60
```

ePWM3A, ePWM3B, ePWM4A and ePWM4B outputs are also controlled by the analog comparator output and driven in a similar fashion just like the ePWM2 module outputs. The initial dead-time values for these SR PWM outputs are defined by the constants `PSFB_SR_DEADBAND_RED_INITIAL` and `PSFB_SR_DEADBAND_FED_INITIAL` in the



*psfbpcmc\_user\_settings.h* file. The dead-time values for the ePWM1 and ePWM2 modules are adjusted with load to optimize the efficiency while the corresponding values for ePWM3 and ePWM4 module outputs (the SR PWM outputs) remains unchanged during operation. All of these dead-time values are user defined programmable parameters. It should be noted that the slope compensation ramp generation, comparator action and PWM waveform generation are all MCU internal hardware generated without any run-time software (code inside ISR1 and ISR2) involvement as shown in the PCMC Control diagram in Figure 6. However, some register re-configurations are done inside ISR1 in order to implement other functions during run time. These functions are, automatic SR mode switching and automatic frequency folding.

The output voltage sensing circuit is made up of simple voltage dividers. A current sensing circuit consisting of a current transformer, diodes, termination resistor and a sense resistor are used to sense the full-bridge transformer primary current.

To quickly run and verify the functionalities for this Lab1 software (s/w), follow the steps listed below:

1. Verify the hardware (h/w) set-up: Insert the F28004x controlCARD in the 100-pin DIMM connector. Follow the PSFB hardware user-guide and connect the required low voltage bench power supplies to provide all the necessary 12VDC and 5VDC bias power for this h/w. Connect a variable output isolated high voltage DC power source (0 - 600V, 4kW power rating) across the high voltage input connector on the board. Then connect an electronic load (60V, 4kW power rating) across the 54V output connector on the board. Set the electronic load in constant resistor (CR) mode for this Lab1 functionality verification and apply an initial load of 10 ohms. Connect an USB cable between the PC and controlCARD. Do not turn ON any of the power supplies at this time.
2. Open Code Composer Studio.
3. Locate the solution package and import the project “dcdc\_psf\_F28004x” in CCS. In the Project Explorer window on the left, click on the “<” sign to the left of Project. Your project window view will look like the following in Figure 9.

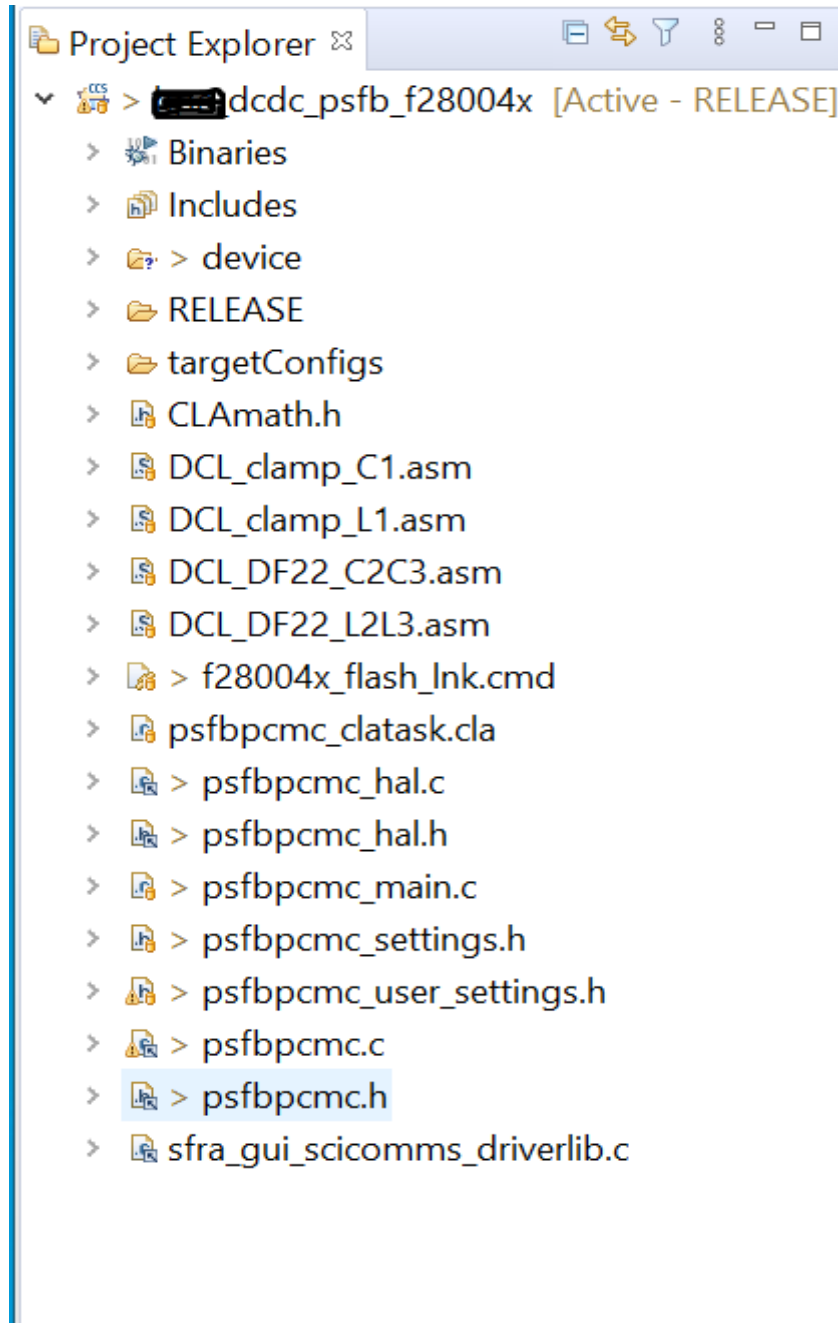


Figure 9 – Project Explorer view

4. Open the file *psfbpcmc\_settings.h*. Set the project for Lab 1 by changing the PSFB\_INCR\_BUILD to 1. Also verify that PSFB\_PCMC is set to 1 in the *psfbpcmc\_user\_settings.h*. For build 1, this will enable the peak current mode control phase shift operation. All the other options in the *psfbpcmc\_user\_settings.h* file can be left at their default settings for now. Click Save.

5. Turn ON the bench power supplies to provide bias power to the board. Click on the Debug button. The Lab 1 code should compile and load to the internal flash memory.
6. Notice the CCS Debug icon in the upper right corner indicating that we are now in the Debug Perspective view. The program should be stopped at the start of *main()*.
7. Populate the expressions window entries by clicking scripting console on the menu bar and then opening the setupdebugenv\_lab1.js file from the project workspace directory (PCMC\_PSFB\_F28004x\setupdebugenv\_lab1.js) using the scripting console Open File() command. Please note that some of the variables have not been initialized at this point in the main code and may contain any un-initialized values. The variables and any other additional variables can also be typed in individually into the watch window. Your watch window view will look like the following in Figure 10.









Expressions  Registers 		
Expression	Type	Value
(x) PSFB_enable	int	0
(x) PSFB_enable_flag	int	0
(x) PSFB_slope	unsigned int	30
(x) PSFB_oring_enable	unsigned int	0
(x) PSFB_gui_icommand_Set_Amps	float	0.0
(x) PSFB_icommand_Set_pu	float	0.0
(x) PSFB_vLVBus_Volts	float	0.0
(x) PSFB_vLVBus_sensed_pu	float	0.0
(x) PSFB_vLVBus_1_sensed_pu	float	0.0
(x) PSFB_guiLVBus_Set_Volts	float	0.0
(x) PSFB_guiLVBus_SlewedSet_Volts	float	0.0
(x) PSFB_start_flag	int	0
(x) PSFB_run_flag	int	0
▼  EPwm2Regs.TBPRD	Register	0x0000
(x) TBPRD	<16-bit unsigned>	0 (Decimal)
▼  EPwm1Regs.TBPRD	Register	0x0000
(x) TBPRD	<16-bit unsigned>	0 (Decimal)
>  EPwm2Regs.DBFED	Register	0 (Decimal)
>  EPwm2Regs.DBRED	Register	0 (Decimal)
>  EPwm1Regs.DBRED	Register	0 (Decimal)
>  EPwm1Regs.DBFED	Register	0 (Decimal)
(x) fbpwm2_db_scaler	float	0.0
(x) fbpwm1_db_scaler	float	0.0
(x) PSFB_SR_mode	int	0
(x) PSFB_SR_enable	int	0

Figure 10 – CCS Watch Window view – Lab 1

8. Enable real-time mode by hovering your mouse on the “Enable real-time mode” button on the horizontal toolbar and clicking on the button. Click on Continuous Refresh button for the watch view.
9. Run the code by using the Run button on the toolbar.
10. Set the electronic load to constant resistor (CR) mode and use a resistor value of 11 ohms.
11. Check the variable PSFB\_enable. This should be set to 1. This enables PWM outputs under peak current mode control (PCMC). The initial slope is set to 55 and the output ORing FET should also be turned on (PSFB\_oring\_enable set to 1). Your watch window view will now look like the following in Figure 11.

<div> <div>(x)= Variables</div> <div>Expressions</div> <div>1010 0101 Registers</div> </div>		
Expression	Type	Value
(x)= PSFB_enable	int	1
(x)= PSFB_enable_flag	int	1
(x)= PSFB_slope	unsigned int	55
(x)= PSFB_oring_enable	unsigned int	1
(x)= PSFB_gui_icommand_Set_Amps	float	0.0
(x)= PSFB_icommand_Set_pu	float	0.0
(x)= PSFB_vLVBus_Volts	float	0.520017087
(x)= PSFB_vLVBus_sensed_pu	float	0.00756835938
(x)= PSFB_vLVBus_1_sensed_pu	float	0.000244140625
(x)= PSFB_guiLVBus_Set_Volts	float	0.0
(x)= PSFB_guiLVBus_SlewedSet_Volts	float	0.0
(x)= PSFB_run_flag	int	1
(x)= PSFB_start_flag	int	0
(x)= PSFB_SR_mode	int	0
(x)= PSFB_SR_switching_enable	int	1
(x)= PSFB_pwm_period_scaler	float	1.0
(x)= PSFB_ovp_Fault	int	0
(x)= PSFB_ocp_Fault	int	0
(x)= fbpwm2_db_scaler	float	9.0
(x)= fbpwm1_db_scaler	float	9.0
> EPwm1Regs.TBPRD	Register	0x0166
> EPwm2Regs.TBPRD	Register	0x0165
> EPwm2Regs.DBFED	Register	0x0048
> EPwm2Regs.DBRED	Register	0x0048
> EPwm1Regs.DBFED	Register	0x0048

Figure 11 – CCS Watch Window view with zero input voltage and code running – Lab 1

12. The variable PSFB\_gui\_icommand\_Set\_Amps is used to set the peak current reference. This current reference is applied to the on-chip comparator input through the internal DAC. Choose an initial low current reference value of 3.5. The RT watch window in Figure 12 below shows that this is set to 3.5. The variable PSFB\_icommand\_Set\_pu represents the per unit value of the peak current reference with respect to the max peak current. In this design the max peak current reference is chosen to be 16A and is initialized by the parameter PSFB\_IHV\_FB\_MAX\_SENSE\_AMPS in *psfbpcmc\_settings.h* file.

(x)= Variables

Expressions

Registers

Expression	Type	Value
PSFB_slope	unsigned int	55
PSFB_oring_enable	unsigned int	1
PSFB_gui_icommand_Set_Amps	float	3.5
PSFB_icommand_Set_pu	float	0.21875
PSFB_vLVBus_Volts	float	14.2616386
PSFB_vLVBus_sensed_pu	float	0.200683594
PSFB_vLVBus_1_sensed_pu	float	0.19921875
PSFB_guiLVBus_Set_Volts	float	0.0
PSFB_guiLVBus_SlewedSet_Volts	float	0.0
PSFB_run_flag	int	1
PSFB_start_flag	int	0
PSFB_SR_mode	int	0
PSFB_SR_switching_enable	int	1
PSFB_pwm_period_scaler	float	1.0
PSFB_ovp_Fault	int	0
PSFB_ocp_Fault	int	0
fbpwm2_db_scaler	float	8.0
fbpwm1_db_scaler	float	8.0
> EPwm1Regs.TBPRD	Register	0x0166
> EPwm2Regs.TBPRD	Register	0x0165
> EPwm2Regs.DBFED	Register	0x0040
> EPwm2Regs.DBRED	Register	0x0040
> EPwm1Regs.DBFED	Register	0x0040
> EPwm1Regs.DBRED	Register	0x0040
+ Add new expression		

Figure 12 – CCS Watch Window view 200Vin, RL=11.1 ohm, Iset = 3.5A – Lab 1

13. Now apply 200 V DC input from the high voltage DC source. The output will rise to around 14V. Observe the watch window variable corresponding to the final output voltage. This is shown as PSFB\_vLVbus\_Volts. This should display a value of about 14. The variable PSFB\_vLVbus\_sensed\_pu represents the per unit value of the output voltage with respect to the max output voltage. In this design the max output voltage is chosen to be 71.29V and is initialized by the parameter PSFB\_VLVBUS\_MAX\_SENSE\_VOLTS in *psfbpcmc\_settings.h* file.

The variable `PSFB_vLVbus_1_sensed_pu` represents the per unit value of the output voltage before the ORing FET with respect to the same max output voltage of 71.29V

Use an oscilloscope to verify the output voltage, the transformer primary and secondary voltage and the transformer primary current. Also check the PWM gate drive signals for the primary bridge and the secondary SR bridge. Appropriate safety precautions should be taken while probing any high voltages and high currents for this isolated DC-DC converter. The scope capture Figure 13 below shows the transformer secondary voltage (Ch 4) and the primary bridge PWM waveforms for the diagonal pair PWM1A (Ch 1) and PWM2B (Ch 2)

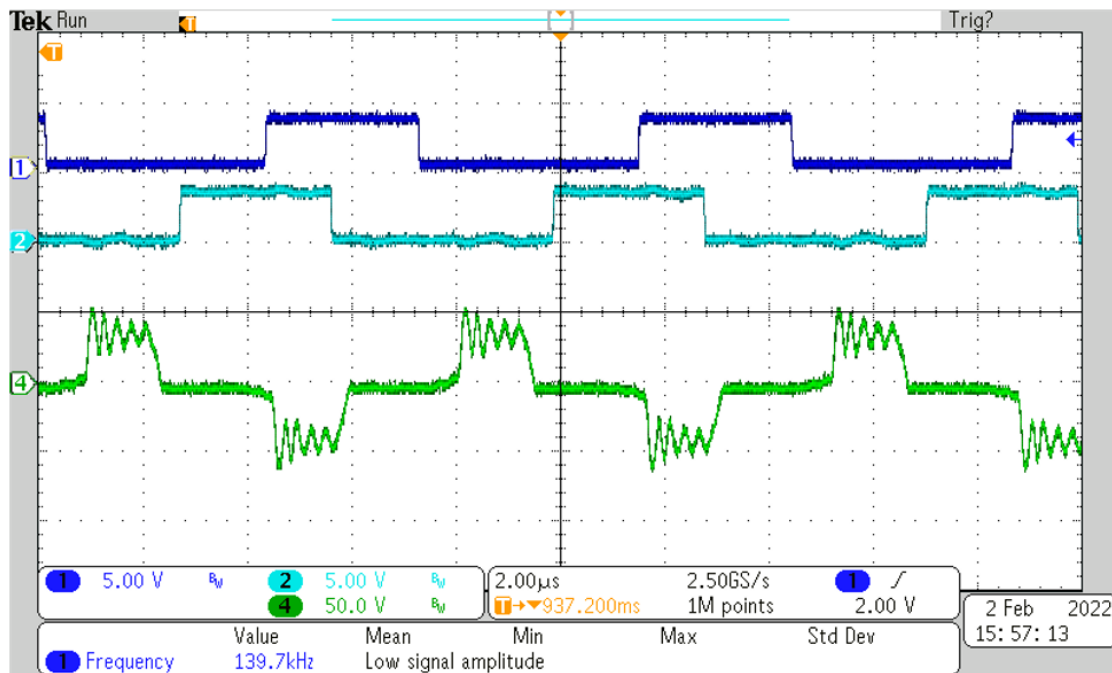


Figure 13 – Test Waveform – Lab 1

14. Check the PWM1 and PWM2 time base period registers on the CCS watch window. These are indicated by `EPwm1Regs.TBPRD` and `EPwm2Regs.TBPRD`. A value of 357 indicates that the PWM frequency is set to 140kHz for this 100MHz MCU. This is the same for PWM3 and PWM4 indicating the same PWM frequency.

15. Check the PWM1 and PWM2 dead-band register values on the CCS watch window. These are indicated by `EPwm1Regs.DBRED`, `EPwm1Regs.DBFED`, `EPwm2Regs.DBRED` and `EPwm2Regs.DBFED`. `EPwm1Regs.DBRED` represents the rising edge delay and `EPwm1Regs.DBFED` represents the falling edge delay for PWM1. A value of 64 indicates that the dead-time is set to 640nS (64 x 10nS) for this 100MHz MCU (10nS CPU clock). These values varied from 80ns to 720nS as the per unit current command `PSFB_icommand_Set_pu` is varied. Higher current command will reduce the dead-time and vice-versa.

This pu current command is proportional to the output load and, therefore, the dead-time gets adjusted based on the load. This is done to maximize and maintain high efficiency over wide load range. To vary the dead-time two scaler variables are used in the code. These scaler variables are shown in the watch window as `fbpwm1_db_scaler` and `fbpwm2_db_scaler`. These scalers change between 1~9 as the load is varied from rated value to light load. The code for this is implemented inside the function `PSFB_UpdateFBPWMsDeadTime()`. The user can find this function inside the file named `psfbpcmc.c`. This function is called from one of the slower

background tasks B1 triggered by CPU Timer 1. Different slower background tasks triggered by CPU Timer 0 and CPU Timer 1 are discussed before towards the end of section 4.1.

A code snippet is shown below. Notice the hysteresis added during transition from one dead-time setting to another. This prevents dead-time values changing continuously when the load range are very close to the dead-time transition thresholds.

```

psfbpcmc.c
475
476 // Update DeadTime function
477 //
478 //
479 void PSFB_UpdateFBPWMsDeadTime(void)
480 {
481     // *****Modified code with 9 values of DT and corresponding load threshold values***** (10/27/21)*****
482     //
483     if(PSFB_icommand_Set_pu <= 0.17f) //Check whether command is less than 0.2pu. Per unit based on 19.74A max
484     {
485         fbpwm1_db_scaler = 9.0f; //If yes, then increase dead time 9 times of the default settings (default = 80ns)
486         fbpwm2_db_scaler = 9.0f;
487     }
488     else if((PSFB_icommand_Set_pu > 0.17f) && (PSFB_icommand_Set_pu <= 0.22f)) //adjust DB for higher load. Always use Scaler of 8 for this range
489     {
490         fbpwm1_db_scaler = 8.0f;
491         fbpwm2_db_scaler = 8.0f;
492     }
493     else if((fbpwm2_db_scaler == 8.0f) && (PSFB_icommand_Set_pu <= 0.16f)) //When using scaler 8 going back to 9
494     {
495         fbpwm1_db_scaler = 9.0f;
496         fbpwm2_db_scaler = 9.0f;
497     }
498     else if((PSFB_icommand_Set_pu > 0.22f) && (PSFB_icommand_Set_pu <= 0.27f)) //Always use Scaler of 7 for this range
499     {
500         fbpwm1_db_scaler = 7.0f;
501         fbpwm2_db_scaler = 7.0f;
502     }
503 }

```

16. Automatic SR mode switching function is enabled in Lab1. The variable *PSFB\_SR\_switching\_enable* is set to 1 indicating that the automatic SR switching is enabled. Based on the value entered for *PSFB\_gui\_icommand\_Set\_Amps*, the SR mode will be set to 0, 1 or 2 (valid three values only) indicating SR mode 0, SR mode 1 and SR mode 2 respectively. Observe the corresponding PWM3 and PWM4 waveforms as the SR modes are changed. Now set the load to 6.5 ohm and *PSFB\_gui\_icommand\_Set\_Amps* to 5. Verify that the SR mode changes to 0 to 1. Verify that the SR PWM waveforms under this condition match with those illustrated in Figure 14 where Ch 1 is SR PWM3B, Ch 2 is SRPWM 4A and Ch 4 is the transformer secondary voltage. The output voltage at this point should be about 26V. The code for run-time SR PWM (PWM3A, PWM3B, PWM4A and PWM4B) signal configuration and generation, under different SR modes, can be found inside the function *PSFB\_pcmcControlCode()*. This function is executed every time the interrupt ISR1 is triggered and the corresponding ISR function *ISR1()* is executed. The function *PSFB\_pcmcControlCode()* resides inside the file named *psfbpcmc.h* and the ISR function *ISR1()* resides inside the file named *psfbpcmc\_main.c*.



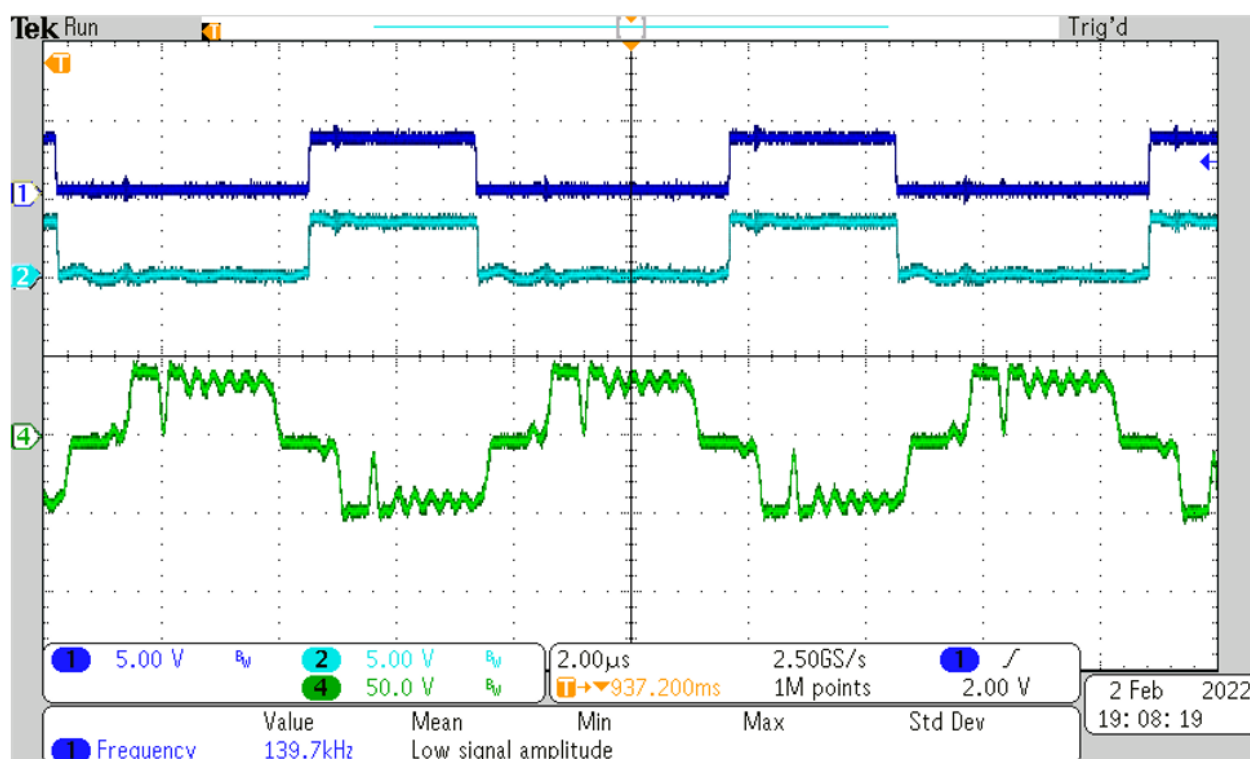


Figure 14 – SR PWMs waveforms under SR mode 1

17. The variable *PSFB\_pwm\_period\_scaler* is set to 1 for Lab1. This indicates that, under Lab 1, the PWM frequency remains constant at 140kHz. In Lab 2 this *PSFB\_pwm\_period\_scaler* is gradually varied between 1 ~ 100 in order to implement frequency folding under light load condition.

18. The variables *PSFB\_ovp\_fault* and *PSFB\_ocp\_fault* is set to 1 in the event of any overvoltage or overcurrent trip condition followed by the activation of the corresponding protection function. The code for over-voltage protection is implemented inside the function named *PSFB\_OVP()*. This OVP function resides inside the file named *psfbpcmc.c* and is called from one of the slower background tasks B1 triggered by CPU Timer 1. Different slower background tasks triggered by CPU Timer 0 and CPU Timer 1 are discussed before towards the end of section 4.1.

19. As discussed before, for this PCMC PSFB DC/DC, transformer peak primary current sense signal is used to implement cycle-by-cycle PCMC. The same current sensed signal is also used to detect any overcurrent condition and the subsequent activation of over current protection function. C2000 MCU has dual comparators for each comparator subsystems (CMPSS). In this implementation cycle-by-cycle PCMC is implemented using the high comparator (CMPSS1H). The corresponding low comparator CMPSS1L in the same comparator subsystems is used to detect the over-current condition. Under any OC condition CMPSS1L generates an internal trip zone (TZ) signal which is used to directly shut off all the necessary PWM signals. Complete OC protection is implemented internally using C2000 MCU on-chip h/w. The configuration code for all these are done during initialization using the protection initialization function called *PSFB\_HAL\_setupBoardProtection()*. Again, this protection initialization function is executed during initialization from the main function called *main()* which resides inside the file *psfdpcmc\_main.c*.



20. When the OC condition is detected by the comparator and the desired protection is triggered, it also sets an internal interrupt flag register bit. One of the slower background tasks A2, triggered by CPU Timer 0, is used to continuously check this flag bit. When this bit is set the code implemented in A2 also sets the OCP fault flag `PSFB_ocp_fault` to 1 indicating an OC fault has occurred.

21. The active clamp (ACL) PWM for this implementation is set to constant 280kHz PWM frequency with an effective pulse of 300nS. The initial pulse width is defined to be 1uSec. And then a rising edge delay (with respect to the PWM1A and PWM1B rising edges) of 700nSec is added. This gives the effective pulse width of 300nS. The code snippet for this initial pulse width setup is shown below. This can be found inside the function `PSFB_HAL_setupACLPWM()`. This setup function resides inside the file named `psfbpcmc_hal.c`.

```
psfbpcmc_hal.c
1504 //
1505 //
1506 EPWM_setPeriodLoadMode(base1, EPWM_PERIOD_DIRECT_LOAD);
1507 //EPWM_setPeriodLoadMode(base1, EPWM_PERIOD_SHADOW_LOAD);
1508 EPWM_setTimeBasePeriod(base1, pwm_period_ticks);
1509 EPWM_setTimeBaseCounter(base1, 0);
1510 EPWM_setPhaseShift(base1, 0);
1511 EPWM_setTimeBaseCounterMode(base1, EPWM_COUNTER_MODE_UP_DOWN);
1512 EPWM_setClockPrescaler(base1, EPWM_CLOCK_DIVIDER_1, EPWM_HSCLOCK_DIVIDER_1);
1513
1514 //
1515 // Counter Compare Submodule Registers
1516 //
1517 //Set the maximum ACL pulse width lower than PWM1A and PWM1B. PWM1A and PWM1B are each 500 ticks.
1518 //Use CMPA and CMPB values to control the max duty of the ACLPWM during +ve and -ve half-cycles of power transfer
1519
1520 EPWM_setCounterCompareValue(base1, EPWM_COUNTER_COMPARE_A, 100); //Initial ACL PWM pulse width of 100*10nS
1521 EPWM_setCounterCompareValue(base1, EPWM_COUNTER_COMPARE_B, pwm_period_ticks - 100); //Init ACL PWM pulse width 100*10nS
1522
```

The code snippet for the ACL dead-time initialization is shown below. This can be found inside the file `psfbpcmc_user_settings.h`. These initial dead-time parameters are used towards the end of the setup function `PSFB_HAL_setupACLPWM()` in order to setup the dead-time using the dead-band (DB) registers.

```
psfbpcmc_user_settings.h
494 //
495 #define PSFB_SR_FLT_L_GPIO 15
496 #define PSFB_SR_FLT_L_GPIO_PIN_CONFIG GPIO_15_EPWM8_B
497
498 #define PSFB_SR_DEADBAND_RED_INITIAL 10
499 #define PSFB_SR_DEADBAND_FED_INITIAL 10
500 //
501 #define PSFB_ACL_DEADBAND_RED_INITIAL 70 //ACL rising edge delay of 700nS
502 #define PSFB_ACL_DEADBAND_FED_INITIAL 1 //ACL falling edge delay of 10nS
503 //
```

21. Now change the load resistor to about 9 ohms. Then slowly increase the input voltage from 200V to 390V. Carefully observe the output voltage represented by the variable `PSFB_vLVbus_Volts`. This should gradually increase to 44V. Also use a voltmeter to cross check the output voltage. The value indicated by `PSFB_vLVbus_Volts` should match closely

with the voltmeter reading. Under this condition the CCS watch window should display the following values.

Expression	Type	Value
PSFB_enable	int	1
PSFB_enable_flag	int	1
PSFB_slope	unsigned int	55
PSFB_oring_enable	unsigned int	1
PSFB_gui_icommand_Set_Amps	float	5.0
PSFB_icommand_Set_pu	float	0.3125
PSFB_vLVBus_Volts	float	43.9470825
PSFB_vLVBus_sensed_pu	float	0.616699219
PSFB_vLVBus_1_sensed_pu	float	0.612304688
PSFB_guiLVBus_Set_Volts	float	0.0
PSFB_guiLVBus_SlewedSet_Volts	float	0.0
PSFB_run_flag	int	1
PSFB_start_flag	int	0
PSFB_SR_mode	int	1
PSFB_SR_switching_enable	int	1
PSFB_pwm_period_scaler	float	1.0
PSFB_ovp_Fault	int	0
PSFB_ocp_Fault	int	0
fbpwm2_db_scaler	float	6.0
fbpwm1_db_scaler	float	6.0
> EPwm1Regs.TBPRD	Register	0x0166
> EPwm2Regs.TBPRD	Register	0x0165
> EPwm2Regs.DBFED	Register	0x0030

The overvoltage protection setting for the output is set to 60V. The code snippet for this initialization is shown below by the parameter PSFB\_V\_TRIP\_LIMIT\_VOLTS and can be found in the file *psfbpcmc\_user\_settings.h*

```

psfbpcmc_settings.h
58
59 #define PSFB_VHVBUS_MAX_SENSE_VOLTS      (float32_t) 673.61
60 #define PSFB_VLVBUS_MAX_SENSE_VOLTS      (float32_t) 71.29
61 #define PSFB_IHV_FB_MAX_SENSE_AMPS        (float32_t) 16 //19.74/2
62 #define PSFB_I_LVMAX_SENSE_AMPS           (float32_t) 71.99
63
64 #define PSFB_V_TRIP_LIMIT_VOLTS           (float32_t) 60
65
66 #define PSFB_I_TRIP_LIMIT_AMPS             (float32_t) 15.98 //18.75/2
67 #define PSFB_VDC_TYPICAL_VOLTS            (float32_t) 400
68 #define PSFB_VDC_Max_VOLTS                (float32_t) 420
69 #define PSFB_VDC_Min_VOLTS                (float32_t) 370
70
71 #define PSFB_VOUT_TYPICAL_VOLTS           (float32_t) 54
72
73 /* Control Loop Gain * /

```

So, if the output voltage is allowed to rise to 60V all the PWMs will trip and the output voltage will reduce to 0V. Use the oscilloscope to again verify the output voltage, the transformer primary and secondary voltage and the transformer primary current. Also check the PWM gate drive signals for the primary bridge, the secondary SR bridge, and the active clamp (ACL) switch. Verify that the ACL PWM has an on-time of 300nS and a rising edge delay of 700nS as explained before. Appropriate safety precautions should be taken while probing any high voltages and high currents for this isolated DC-DC converter.

22. The s/w variable used for slope compensation is *PSFB\_slope*. This is initialized to 55. User can gradually increase or decrease it and observe how phase shift and, therefore, the output voltage, is affected as this parameter is varied.

23. The automatic frequency folding function allows PWM frequency to be reduced under light load condition in order to maintain output voltage regulation. This is needed since the minimum phase shift cannot be reduced to zero to ensure proper operation of the ACL circuit. This automatic frequency folding is disabled in Lab1 since in this lab the voltage loop is open. However, frequency can be reduced manually from the CCS watch window under user control. To do that add the variable *PSFB\_freq\_fold\_enable* to the watch window and verify that it is set to 0. The variable *PSFB\_pwm\_period\_scaler* should also show a value of 1. Now check that the PWM frequency of any of the primary or secondary bridge PWM shows 140kHz. Now from the watch window change *PSFB\_pwm\_period\_scaler* from 1.0 to 1.05. This should reduce the PWM frequency by 5%. Consequently, the output voltage should also drop by a small amount. Test this frequency reduction further by changing the *PSFB\_pwm\_period\_scaler* by a small amount 1 -> 1.05 or 1.05 -> 1.1 and so on, verifying the frequency in every step. Under voltage loop control (Lab 2) this *PSFB\_pwm\_period\_scaler* will be increased only under light load condition and can go up to about 100 indicating minimum PWM frequency of 1.4kHz at zero load. The code for frequency folding, together with its automatic control, can be found in the file named *psfbpcmc.h*. Below is a code snippet from this section of the code.

```

psfbpcmc.h
507
508 if(PSFB_freq_fold_enable == 1)//For closed volt loop mode set this to 1 to check freq reduction operation
509 {
510     if(PSFB_icommand_Set_pu <= 0.13f) //Check whether the duty is saturated to minimum value
511     {
512         if((PSFB_vLVBus_sensed_pu - PSFB_vcommand_Set_pu )>0.01f) //If yes, then check whether Vout is higher
513         {
514             PSFB_pwm_period_scaler = PSFB_pwm_period_scaler + 0.05f; //If yes, then reduce frequency
515             PSFB_pwm_period_scaler = (PSFB_pwm_period_scaler > 99.0f)?
516                 100.0f:PSFB_pwm_period_scaler; //Limit minimum freq 1/100 of the normal
517             PSFB_pwm_period_Set = PSFB_pwm_period_scaler * (PSFB_FB_PWM_PERIOD_TICKS);
518         }
519         else
520         {
521             //Duty is saturated to min. But Vout is within 1% of setpoint. So maintain the freq at its current
522             PSFB_pwm_period_Set = PSFB_pwm_period_scaler * (PSFB_FB_PWM_PERIOD_TICKS);
523         }
524     }
525     else if(PSFB_icommand_Set_pu > 0.25f) //Check whether Duty is out of min saturated value by an additional
526     {
527         PSFB_pwm_period_scaler = PSFB_pwm_period_scaler - 0.05f; //If yes, then increase frequency
528         PSFB_pwm_period_scaler = (PSFB_pwm_period_scaler < 1.0f)?
529             1.0f:PSFB_pwm_period_scaler; //Limit maximum freq to 100kHz
530         PSFB_pwm_period_Set = PSFB_pwm_period_scaler * (PSFB_FB_PWM_PERIOD_TICKS);
531     }
532     else
533     {
534         PSFB_pwm_period_Set = PSFB_pwm_period_scaler * (PSFB_FB_PWM_PERIOD_TICKS); //Hold the freq to its current
535                                             // command has not exceeded
536     }
537 }
538 else
539 {
540 #if PSFB_INCR_BUILD == 1|
541     PSFB_pwm_period_Set = PSFB_pwm_period_scaler * (PSFB_FB_PWM_PERIOD_TICKS);
542 #endif

```

24. This completes most of the functionalities test under Lab 1. For completely halting the MCU when in real-time mode follow the following steps. First reduce the DC input voltage to 0V and wait a few seconds. Then, halt the processor by using the Halt button on the toolbar, or by using Target > Halt. Finally click the button again to take the MCU out of real-time mode and then reset the MCU.

## 5.2 Lab 2

The objective of Lab 2 is to verify the operation of the complete PCMC based PSFB project from the CCS environment. Both the over-voltage protection (OVP) and the over-current protection (OCP) are also enabled in Lab 2. In addition to the functionalities described in Lab 1, Lab2 further implements closed voltage loop control, soft-start function, automatic SR mode switching, automatic frequency folding under light load and automatic reverse current protection.

Figure 15 shows the C2000 software and on-chip h/w blocks used in this Lab 2. A two pole two zero controller is used for the voltage loop. As shown in Figure 15, the voltage loop controller is executed at a frequency of  $f_s$  where,  $f_s = 70$  KHz. The DCL\_runClamp\_C1 function is used to avoid controller wind-up condition. The DCL\_runDF22\_C2 computes the immediate part of the pre-computed DF22 controller. If this value is larger than the max or less than min threshold, a flag will be set with DCL\_runClamp\_C1 function. This stops the further controller saturation. Once the flag is off, DCL\_runDF22\_C3 will compute the partial result of the pre-computed DF22 controller. More information on how the DCL function works could be found inside DCL user's guide in C2000ware.



6. Run the code by using the Run button on the toolbar. Your watch window view will look like the following in Figure 16.

<div> <div>(x)= Variables</div> <div>Expressions</div> <div>Registers</div> </div>		
Expression	Type	Value
PSFB_enable	int	0
PSFB_enable_flag	int	0
PSFB_slope	unsigned int	55
PSFB_oring_enable	unsigned int	0
PSFB_gui_icommand_Set_Amps	float	0.0
PSFB_icommand_Set_pu	float	0.129999995
PSFB_vLVBus_Volts	float	0.520017087
PSFB_vLVBus_sensed_pu	float	0.00708007812
PSFB_vLVBus_1_sensed_pu	float	0.00048828125
PSFB_guiLVBus_Set_Volts	float	0.504738748
PSFB_guiLVBus_SlewedSet_Volts	float	0.0
PSFB_run_flag	int	0
PSFB_start_flag	int	0
PSFB_SR_mode	int	0
PSFB_SR_switching_enable	int	0
PSFB_pwm_period_scaler	float	1.0
PSFB_ovp_Fault	int	0
PSFB_ocp_Fault	int	0
fbpwm2_db_scaler	float	8.0
fbpwm1_db_scaler	float	8.0
> EPwm1Regs.TBPRD	Register	0x0166
> EPwm2Regs.TBPRD	Register	0x0165
> EPwm2Regs.DBFED	Register	0x0008
> EPwm2Regs.DBRED	Register	0x0008
> EPwm1Regs.DBFED	Register	0x0008

Figure 16 – CCS Watch Window view Lab 2

- Set the electronic load to constant current (CC) mode and use a current value of 5A.
- Check the variable PSFB\_enable. This should be set to 0 at this point. Also, the variables PSFB\_run\_flag, PSFB\_SR\_switching enable and PSFB\_oring\_enable should all be set to 0.
- Now slowly apply 390 V DC input from the high voltage DC source. The output voltage reading should be almost zero as the PWM2A and PWM2B are both off at this point. Now from the watch window, set the variable PSFB\_start\_flag to 1. The output will slowly rise to around 54V. Observe the watch window variable PSFB\_vLVbus\_Volts for the final output voltage. This should be about 54V. The initial slope is set to 55. At this point the variables PSFB\_enable\_flag,



PSFB\_run\_flag, PSFB\_SR\_switching enable, PSFB\_freq\_fold\_enable and PSFB\_oring\_enable should all be set to 1. Your watch window view will look like the following in Figure 17.

<div> <div>(x)= Variables</div> <div>Expressions</div> <div>Registers</div> </div>		
Expression	Type	Value
(x)= PSFB_enable	int	0
(x)= PSFB_enable_flag	int	1
(x)= PSFB_slope	unsigned int	55
(x)= PSFB_oring_enable	unsigned int	1
(x)= PSFB_gui_icommand_Set_Amps	float	0.0
(x)= PSFB_icommand_Set_pu	float	0.364282668
(x)= PSFB_vLVBus_Volts	float	54.2396431
(x)= PSFB_vLVBus_sensed_pu	float	0.757568359
(x)= PSFB_vLVBus_1_sensed_pu	float	0.751953125
(x)= PSFB_guiLVBus_Set_Volts	float	54.0
(x)= PSFB_guiLVBus_SlewedSet_Volts	float	54.0
(x)= PSFB_run_flag	int	1
(x)= PSFB_start_flag	int	1
(x)= PSFB_SR_mode	int	1
(x)= PSFB_SR_switching_enable	int	1
(x)= PSFB_pwm_period_scaler	float	1.0
(x)= PSFB_ovp_Fault	int	0
(x)= PSFB_ocp_Fault	int	0
(x)= fbpwm2_db_scaler	float	5.0
(x)= fbpwm1_db_scaler	float	5.0
> EPwm1Regs.TBPRD	Register	0x0166
> EPwm2Regs.TBPRD	Register	0x0165
> EPwm2Regs.DBFED	Register	0x0028
> EPwm2Regs.DBRED	Register	0x0028
> EPwm1Regs.DBFED	Register	0x0028

Figure 17 – CCS Watch Window view Lab 2

10. When PSFB\_start\_flag is set to 1, the soft-start function *PSFB\_softstart()* is initiated and, output voltage slowly ramps to the target value of 54V. The code for *PSFB\_softstart()* resides inside the file *psfbpcmc.c*.

```

psfbpcmc.c
678
679 //
680 // PSFB_SoftStart
681 //
682 void PSFB_SoftStart(void)
683 {
684     if(PSFB_SR_switching_enable == 0)
685     {
686         if(PSFB_run_flag == 0)
687         {
688             if(PSFB_vLVBus_Volts > (PSFB_VOUT_TYPICAL_VOLTS*0.25f))
689             {
690                 fbpwm1_width_cntr = 1.0f;
691
692                 EALLOW;
693                 #if HPE_BD_SW_WORK_AROUND == 0
694                     EPWM_setCounterCompareValue(PSFB_FB_PWM1_BASE, EPWM_COUNTER_COMPARE_A, 0xFFFF);
695                     EPWM_setCounterCompareValue(PSFB_FB_PWM1_BASE, EPWM_COUNTER_COMPARE_B, 0);
696                 #endif
697
698                 #if HPE_BD_SW_WORK_AROUND == 1
699                     EPWM_setCounterCompareValue(EPWM3_BASE, EPWM_COUNTER_COMPARE_A, 0xFFFF);
700                     EPWM_setCounterCompareValue(EPWM3_BASE, EPWM_COUNTER_COMPARE_B, 0);
701                 #endif
702                 EDIS;
703
704                 //Enable frequency folding
705                 PSFB_freq_fold_enable = 1; //Allow for freq reduction during soft-start and a monotonic rampup
706
707                 PSFB_cntlMax = 0.9;
708                 PSFB_guivLVBus_Set_Volts = PSFB_VOUT_TYPICAL_VOLTS;
709             }
710         }
711     }

```

11. Increase the load current to 15A and use an oscilloscope to verify the transformer primary current, transformer secondary voltage and PWM gate drive signals for the primary bridge and the secondary SR bridge. Appropriate safety precautions should be taken while probing any high voltages and high currents for this isolated DC-DC converter. The scope capture below in Figure 18 shows the transformer secondary voltage, primary current and the primary bridge PWM waveforms for the diagonal pair PWM1A and PWM2B. with 390Vin, 54Vout, 15A Load and SR Mode 1. Figure 19 shows the same except that C1 shows the ACL PWM instead of Vsec



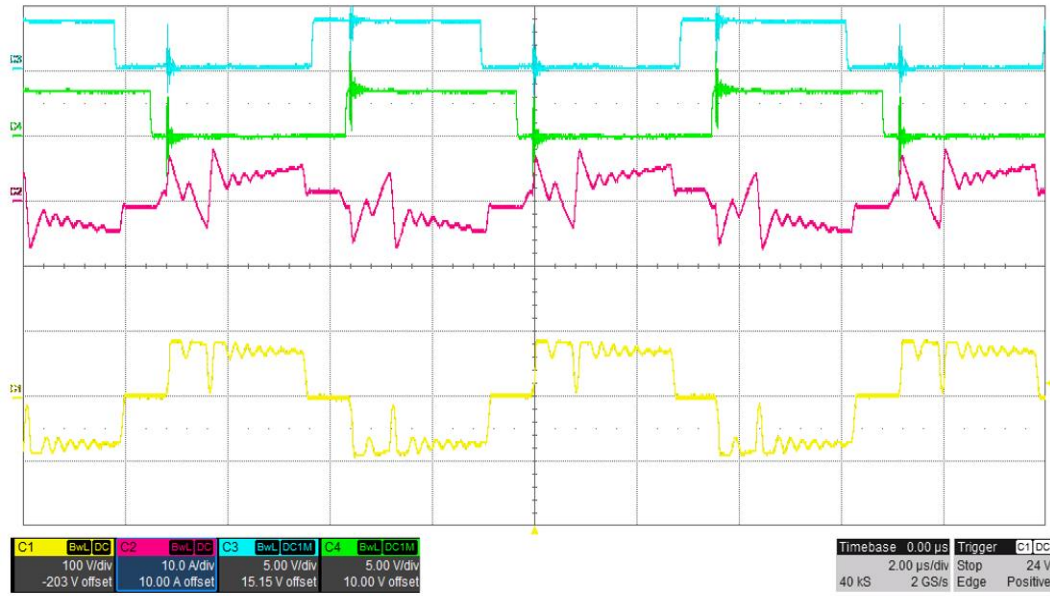


Figure 18 – C1 Vsec, C2 Ipri, C3 leg2\_Is PWM, C4 leg1\_hs PWM, SR Mode 1

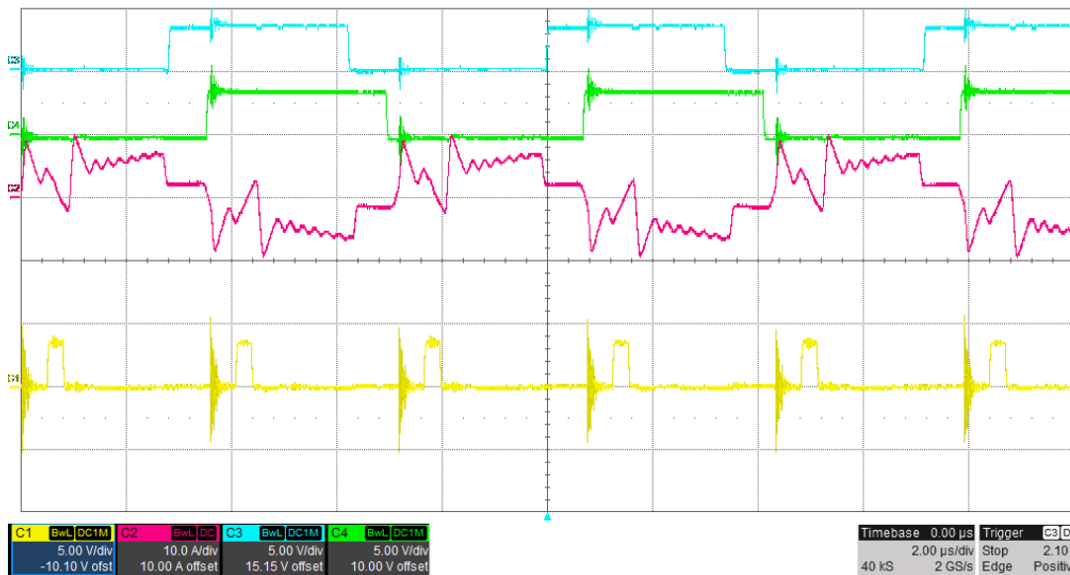


Figure 19 – C1 ACL PWM, C2 Ipri, C3 leg2\_Is PWM, C4 leg1\_hs PWM, SR Mode 1

12. Check the PWM frequency for all primary and secondary PWM are set to 140kHz while the ACL PWM frequency is 280kHz. This is the same for PWM3 and PWM4 indicating the same PWM frequency of 140kHz.

13. Check that the primary PWM dead-time gets adjusted from 720nS to 80nS (9 discrete steps) as the load varies between 2A to full load of 55A (3kW output power). During this load current change the output voltage should be well regulated at 54V. The scaler variables fbpwm1\_db\_scaler and fbpwm2\_db\_scaler will change from 9 to 1 during this load increase.

The code for this is implemented inside the function *PSFB\_UpdateFBPWMsDeadTime()*. The user can find this function inside the file named “*psfbpcmc.c*”. This function is called from one of the slower background tasks B1 triggered by CPU Timer 1. The code for task B1 resides inside *psfbpcmc\_main.c* as shown below. This shows that the automatic dead-time update works only after the end of the soft-start when the variable *PSFB\_SR\_switching\_enable* is also set to 1. During soft-start the dead-time stays constant at 80nS. The code for task B1 also shows that the over-voltage protection function *PSFB\_OVP()* and the start-control function are also called from the same background task B1. Different slower background tasks triggered by CPU Timer 0 and CPU Timer 1 are discussed before towards the end of section 4.1.

```

psfbpcmc_main.c
270
271 void B1(void)
272 {
273     //
274     // Over voltage protection
275     //
276     PSFB_OVP();
277     //
278     //
279     // start_flag set by user to start operation
280     //
281 #if PSFB_INCR_BUILD == 2
282     PSFB_StartupControl();
283 #endif
284
285 //
286 //
287 if(PSFB_SR_switching_enable == 1)
288 {
289     PSFB_UpdateFBPWMsDeadTime();
290 }
291 //
292 //
293 //the next time CpuTimer1 'counter' reaches Period value go to B2
294 //
295 B_Task_Ptr = &B2;
296 }

```

14. Automatic SR mode switching function is enabled in Lab2. Also, the variable *PSFB\_SR\_switching\_enable* is set to 1 only after soft-start is completed and the *PSFB\_run\_flag* is set to 1. This means that the automatic SR switching is disabled during soft-start. After the soft-start is completed and the load current is varied between 2A to 55A (full load of 3kW) the variable *PSFB\_SR\_mode* will change automatically between 0, 1 or 2 only, indicating SR mode 0, SR mode 1 and SR mode 2 respectively. Observe the corresponding PWM3 and PWM4 waveforms as the SR modes varies with load. Verify that the waveforms under different modes match with those illustrated in Figure 20 and Figure 21 at the specified operation conditions.

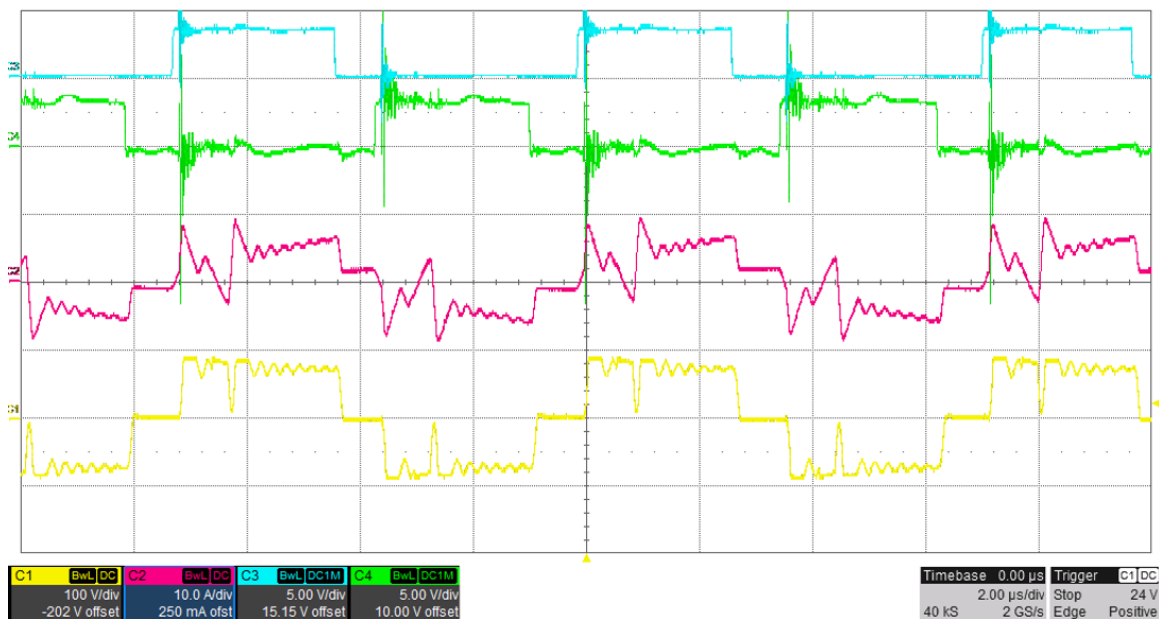


Figure 20 – C1 Vsec, C2 Ipri, C3 leg4\_Is PWM, C4 leg3\_Is PWM, SR Mode 1  
390Vin, 54Vout, 20A Load

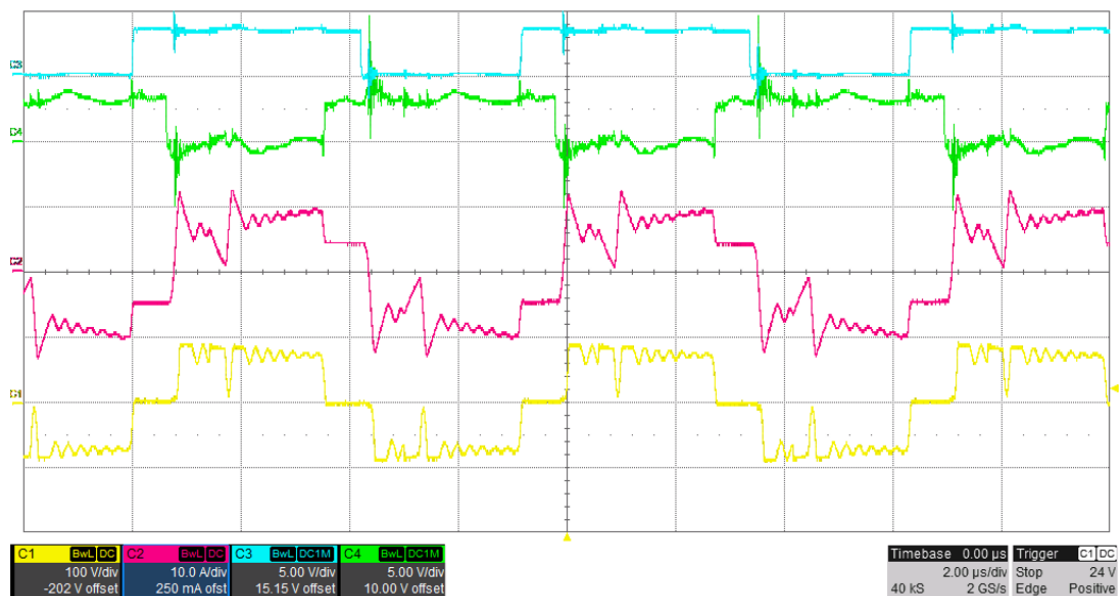


Figure 21 – C1 Vsec, C2 Ipri, C3 leg4\_Is PWM, C4 leg3\_Is PWM, SR Mode 2  
390Vin 54Vout 40A Load

The code for automatic SR mode switching can be found inside the `psfbpcmc.h` file. A code snippet for the automatic SR mode switching is presented below. Notice the hysteresis added between the transition points of different modes. This prevents continuous SR mode switching when the load level is very close to the SR mode transition thresholds.

```

psfbpcmc.h
667 #if PSFB_INCR_BUILD == 2
668
669     if((PSFB_SR_switching_enable == 1) && (PSFB_pwm_period_scaler == 1.0f))//added this new condition on 9/23. This w
670         //after soft-start when PSFB_SR_switching_enable is set to 1. This auto mode switching is also enabled only
671     {
672         //if((PSFB_icommand_Set_pu > 0.20f) && (PSFB_icommand_Set_pu <= 0.65f) && (PSFB_SR_mode != 2)) //Switch from mo
673         if((PSFB_icommand_Set_pu > 0.13f) && (PSFB_icommand_Set_pu <= 0.45f) && (PSFB_SR_mode != 2))
674         {
675             PSFB_SR_mode = 1;
676         }
677         //else if((PSFB_SR_mode == 1)&&(PSFB_icommand_Set_pu <= 0.14f))// Once mode = 1 has been set then wait until co
678         else if((PSFB_SR_mode == 1)&&(PSFB_icommand_Set_pu <= 0.13f))
679         {
680             PSFB_SR_mode = 0;
681         }
682         //else if((PSFB_SR_mode == 1) && (PSFB_icommand_Set_pu > 0.14f) && (PSFB_icommand_Set_pu <= 0.20f))// Once mode
683         else if((PSFB_SR_mode == 1) && (PSFB_icommand_Set_pu > 0.13f) && (PSFB_icommand_Set_pu <= 0.35f))
684         {
685             PSFB_SR_mode = 1;
686         }
687         else if(PSFB_icommand_Set_pu > 0.45f)//Switch from mode 1 to mode 2 under this condition
688         {
689             PSFB_SR_mode = 2;
690         }
691         else if((PSFB_SR_mode == 2) && (PSFB_icommand_Set_pu <= 0.35f))//Once mode = 2 has been set then wait until com
692         {
693             PSFB_SR_mode = 1;
694         }
695         else if((PSFB_SR_mode == 2) && (PSFB_icommand_Set_pu > 0.35f) && (PSFB_icommand_Set_pu <= 0.45f))//Once mode =
696         {
697             PSFB_SR_mode = 2;
698         }
699         //else if((PSFB_SR_mode == 2) && (PSFB_icommand_Set_pu <= 0.20f))//In mode 2 if command drops to 0.2 then switc
700         else if((PSFB_SR_mode == 2) && (PSFB_icommand_Set_pu <= 0.13f))
701     {

```

15. The variable *PSFB\_pwm\_period\_scaler* is set to 1 for Lab2 when the load is higher than about 1A. For loads less than 1A the variable *PSFB\_pwm\_period\_scaler* is gradually varied between 1 ~ 100 in order to implement frequency folding under light load condition. Verify that the minimum frequency is about 4~5kHz at no load and output voltage is well regulated.

16. The variables *PSFB\_ovp\_fault* and *PSFB\_ocp\_fault* is set to 1 in the event of any overvoltage or overcurrent trip condition under Lab 2.

17. As discussed before, for this PCMC PSFB DC/DC, transformer peak primary current sensed signal is used to implement cycle-by-cycle PCMC. The same current sensed signal is also used to detect any overcurrent condition and the subsequent activation of over current protection function. Under any OC condition C2000 internal comparator generates an internal trip zone (TZ) signal which is used to directly shut off all the necessary PWM signals, all implemented internally using C2000 MCU on-chip h/w. The configuration code for all these are done during initialization using the protection function called *PSFB\_HAL\_setupBoardProtection()*. Again, this protection function is executed during initialization from the main function called *main()* which resides inside the file *psfdpcmc\_main.c*.

18. When the OC condition is detected by the comparator it also sets an internal interrupt flag register bit. One of the slower background tasks A2, triggered by CPU Timer 0, is used to continuously check this flag bit. When this bit is set the code implemented in A2 also sets the OCP fault flag `PSFB_ocp_fault` to 1 to indicate an over-current fault condition.

19. The active clamp (ACL) PWM for this Lab 2 is also set to constant 280kHz PWM frequency with an effective pulse of 300nS. This is explained in Lab 1 and a code snippet is also presented in that section. Verify that the ACL PWM has an on-time of 300nS and a rising edge delay of 700nS as explained before. Appropriate safety precautions should be taken while probing any high voltages and high currents for this isolated DC-DC converter. The scope capture presented in Figure 19 before shows the primary bridge PWM waveforms for the diagonal pair PWM1A and PWM2B, transformer primary current and ACL PWM under SR mode 1 with 390Vin, 54Vout, 15A load.

20. The s/w variable used for slope compensation is `PSFB_slope`. This is initialized to 55. User can gradually increase or decrease it and observe how the output voltage stability is affected as this parameter is varied. The slope value can be changed between 0 to 255. A low value of slope compensation will cause sub-harmonic oscillation and a high value can make the output response very damped.

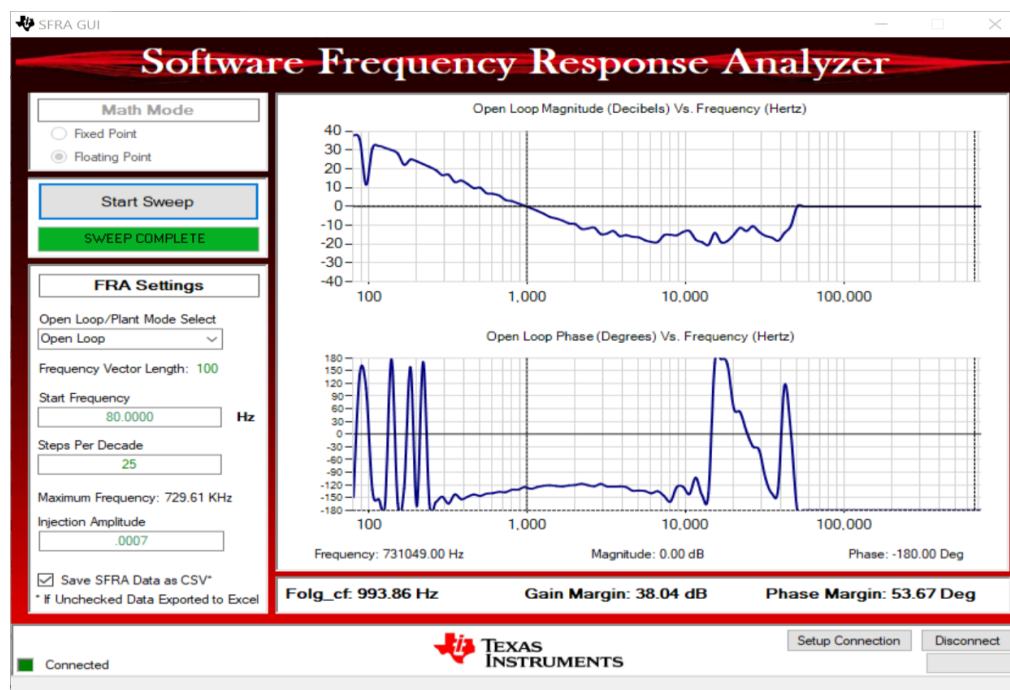
21. The automatic frequency folding function allows PWM frequency to be reduced under light load condition in order to maintain output voltage regulation. This is needed since the minimum phase shift cannot be reduced to zero to ensure proper operation of the ACL circuit. This automatic frequency folding is enabled in Lab2 during soft-start. Add the variable `PSFB_freq_fold_enable` to the watch window and verify that it is set to 1. The variable `PSFB_pwm_period_scaler` should also show a value of 1 when the load current is higher than 1A. Check that the PWM frequency of any of the primary or secondary bridge PWM shows 140kHz. Now reduce the load below 1A and from the watch widow observe how `PSFB_pwm_period_scaler` changes from 1.0 to higher values. Consequently, this will reduce the PWM frequency by a proportional amount in order to maintain output voltage regulated at 54V. Only under light load condition (load < 1A) this `PSFB_pwm_period_scaler` will increase from 1 and can go up to about 100 indicating minimum PWM frequency of 1.4kHz. However at zero load this design operates at about 4-5kHz PWM in order to maintain output voltage regulation. The code for frequency folding, together with its automatic control, can be found inside the function `PSFB_pcmcControlCode()`. This function is executed every time the interrupt ISR1 is triggered and the corresponding ISR function `ISR1()` is executed. The function `PSFB_pcmcControlCode()` resides inside the file named `psfbpcmc.h`. Below is a code snippet from this section of the code.

```

psfbpcmch
507
508 if(PSFB_freq_fold_enable == 1)//For closed volt loop mode set this to 1 to check freq reduction operation
509 {
510     if(PSFB_icommand_Set_pu <= 0.13f) //Check whether the duty is saturated to minimum value
511     {
512         if((PSFB_vLVBus_sensed_pu - PSFB_vcommand_Set_pu )>0.01f) //If yes, then check whether Vout is higher
513         {
514             PSFB_pwm_period_scaler = PSFB_pwm_period_scaler + 0.05f; //If yes, then reduce frequency
515             PSFB_pwm_period_scaler = (PSFB_pwm_period_scaler > 99.0f)?
516                 100.0f:PSFB_pwm_period_scaler; //Limit minimum freq 1/100 of the normal
517             PSFB_pwm_period_Set = PSFB_pwm_period_scaler * (PSFB_FB_PWM_PERIOD_TICKS);
518         }
519         else
520         {
521             //Duty is saturated to min. But Vout is within 1% of setpoint. So maintain the freq at its current
522             PSFB_pwm_period_Set = PSFB_pwm_period_scaler * (PSFB_FB_PWM_PERIOD_TICKS);
523         }
524     }
525     else if(PSFB_icommand_Set_pu > 0.25f) //Check whether Duty is out of min saturated value by an additional
526     {
527         PSFB_pwm_period_scaler = PSFB_pwm_period_scaler - 0.05f; //If yes, then increase frequency
528         PSFB_pwm_period_scaler = (PSFB_pwm_period_scaler < 1.0f)?
529             1.0f:PSFB_pwm_period_scaler; //Limit maximum freq to 100kHz
530         PSFB_pwm_period_Set = PSFB_pwm_period_scaler * (PSFB_FB_PWM_PERIOD_TICKS);
531     }
532     else
533     {
534         PSFB_pwm_period_Set = PSFB_pwm_period_scaler * (PSFB_FB_PWM_PERIOD_TICKS); //Hold the freq to its current
535                                             // command has not exceeded
536     }
537 }
538 else
539 {
540     #if PSFB_INCR_BUILD == 1
541         PSFB_pwm_period_Set = PSFB_pwm_period_scaler * (PSFB_FB_PWM_PERIOD_TICKS);
542     #endif

```

22. To check the voltage control loop Bode-plot set the load current to about 15A. Now run the SFRA\_GUI.exe from the sfra gui folder. The voltage loop bode plot should be as shown below. This plot shows a bandwidth of about 1kHz with 38dB gain margin and 54 deg phase margin





23. The reverse current protection function shuts off the SR PWMs when the phase shift is very low. This is indicated by a very low pulse width of the transformer primary or secondary voltage for both +ve and -ve cycle (of transformer voltage). Under this condition C2000 MCU captures the PWM timer counter value when the internal PCMC comparator trips in each PWM cycle. The timer counter value is a measure of the phase shift in that PWM cycle and this is compared with a specified minimum phase shift. If this is less than or equal to the minimum phase shift then the SR PWMs are turned off to prevent power back flow from the secondary side to the primary side. To capture the timer counter value a second interrupt ISR2 is generated in every half cycle of the primary PWM cycle. In every half cycle when the internal PCMC comparator trips the PWM it also generates an interrupt. Therefore, the frequency of ISR2 is 280kHz. This interrupt is given highest priority and it interrupts the other 70kHz ISR1. Inside ISR2, the value of the timer counter is quickly read and compared against the minimum phase shift. Then SR mode is set to 0 (turning OFF all SR PWM) for the next PWM cycle if the timer counter value is less than the minimum phase shift. Otherwise, the SR mode remains unchanged to what is set by the load demand as explained before in automatic SR mode switching section. This new ISR2 code can be found inside psfbpcmc\_main.c as shown below.

```
psfbpcmc_main.c
155
156 //New ISR for capturing TCNTR at T1 Events
157 //
158
159 #if PSFB_CONTROL_RUNNING_ON == C28x_CORE
160 interrupt void ISR2(void) //Resonant freq
161 {
162     PSFB_TCNTR_CaptureCode();
163
164     PSFB_HAL_clearInterrupt(PSFB_C28x_ISR2_INTERRUPT_PIE_GROUP_NO);
165
166 }
167 #endif
168
169 //*****end of new ISR here
```

Inside this ISR2, the function PSFB\_TCNTR\_CaptureCode() is executed which captures the counter value, compares it against the minimum phase shift and sets the SR mode. The code for this function resides inside psfbpcmc.h file and is shown below.

```

*psfbpcmc.h
261 //*****New ISR for TCNTR value capture at T1 Event
262 //
263 //
264 #pragma FUNC_ALWAYS_INLINE(PSFB_TCNTR_CaptureCode)
265
266 //
267 //
268 inline void PSFB_TCNTR_CaptureCode(void)
269 {
270     PSFB_epwm2counter = HWREGH(PSFB_ADC_OVERSAMPLE_PWM_BASE + EPWM_O_TBCTR);
271     //
272     //
273     #if PSFB_INCR_BUILD == 2
274
275     if(PSFB_run_flag == 1)
276     {
277
278         if (PSFB_epwm2counter < PSFB_min_phase_shift)
279         {
280             PSFB_SR_switching_enable = 0;
281         }
282         else
283         {
284             PSFB_SR_switching_enable = 1;
285         }
286
287     }
288     #endif
289     //
290     EPWM_clearTripZoneFlag(PSFB_C28x_ISR2_INTERRUPT_TRIG_PWM_BASE, EPWM_TZ_FLAG_DCAEVT2);
291

```

24. This completes the test of all functionalities under Lab 2. For completely halting the MCU when in real-time mode follow the following steps. First shut off the DC input voltage and wait a few seconds. Then, halt the processor by using the Halt button on the toolbar, or by using Target > Halt. Finally click the button again to take the MCU out of real-time mode and then reset the MCU.