PMP23216 Peak Current Mode Controlled (PCMC) Phase Shifted Full-Bridge DC-DC Converter

– September 2022

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. A current doubler sync 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. 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 Coss oscillates with Lo and Lr causing ringing across the output rectifiers. This requires MOSFETs with higher VDS rating and, therefore, results in higher Rds,on and higher power losses. Therefore, active clamping circuits (ACLs) across the secondary rectifiers are required to limit the ringing across the output rectifiers allowing the use of lower voltage devices for best possible efficiency. In Figure 1 each 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.

Implementing PCMC for such a PSFB system requires complicated pulse width modulated (PWM) waveform generation with precise timing control. A C2000 microcontroller (MCU) is used to generate all the necessary PWM waveforms without requiring any additional support circuitry. The MCU provides programmable on-chip slope compensation hardware that is used for appropriate slope compensation to guarantees open loop stability and to eliminate any sub-harmonic oscillations.

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.

This document presents the details of this microcontroller-based implementation of peak current mode controlled (PCMC) phase shifted full-bridge (PSFB) DC-DC converter.

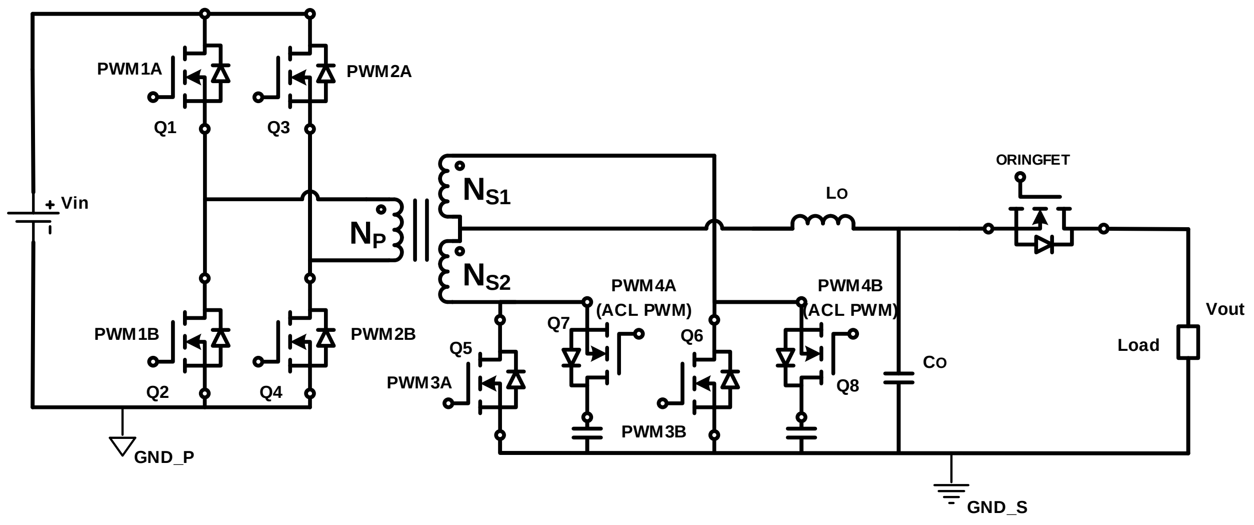


Figure 1. PSFB DC/DC Power Stage with Active Clamp Output

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. Two MOSFET or GaN switches (Q5 and Q6) on the secondary low voltage (LV) side work as synchronous rectifiers (SR). ACL PWM drives the ACL MOSFET Q7 and Q8. This topology allows all the high voltage switching devices to switch with zero voltage switching (ZVS) resulting in lower switching losses and higher efficiency.

Q1 and Q2 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 lag consisting of Q3 and Q4 also maintain 50 % duty and 180 degree phase shift with each other. The PWM signals for leg Q3 – Q4 of the full-bridge are phase shifted with respect to those for leg Q1 – Q2. 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. Two switches Q5 and Q6 provide synchronous rectification (SR) on the secondary, while Lo and Co form the output filter. ACL PWMs driving the ACL switch Q7 and Q8 runs at the PWM frequency. The ACL PWMs PWM4A and PWM4B maintain a fixed pulse width and a fixed delay with respect to the leading leg PWM signals PWM1A and PWM1B respectively. The ACL switches turn on/off the clamp capacitor which helps limit the ringing across the output rectifers. Figure 3, 4 and 5 provide all the PWM waveforms and peak primary current waveforms for the system in Figure 2.

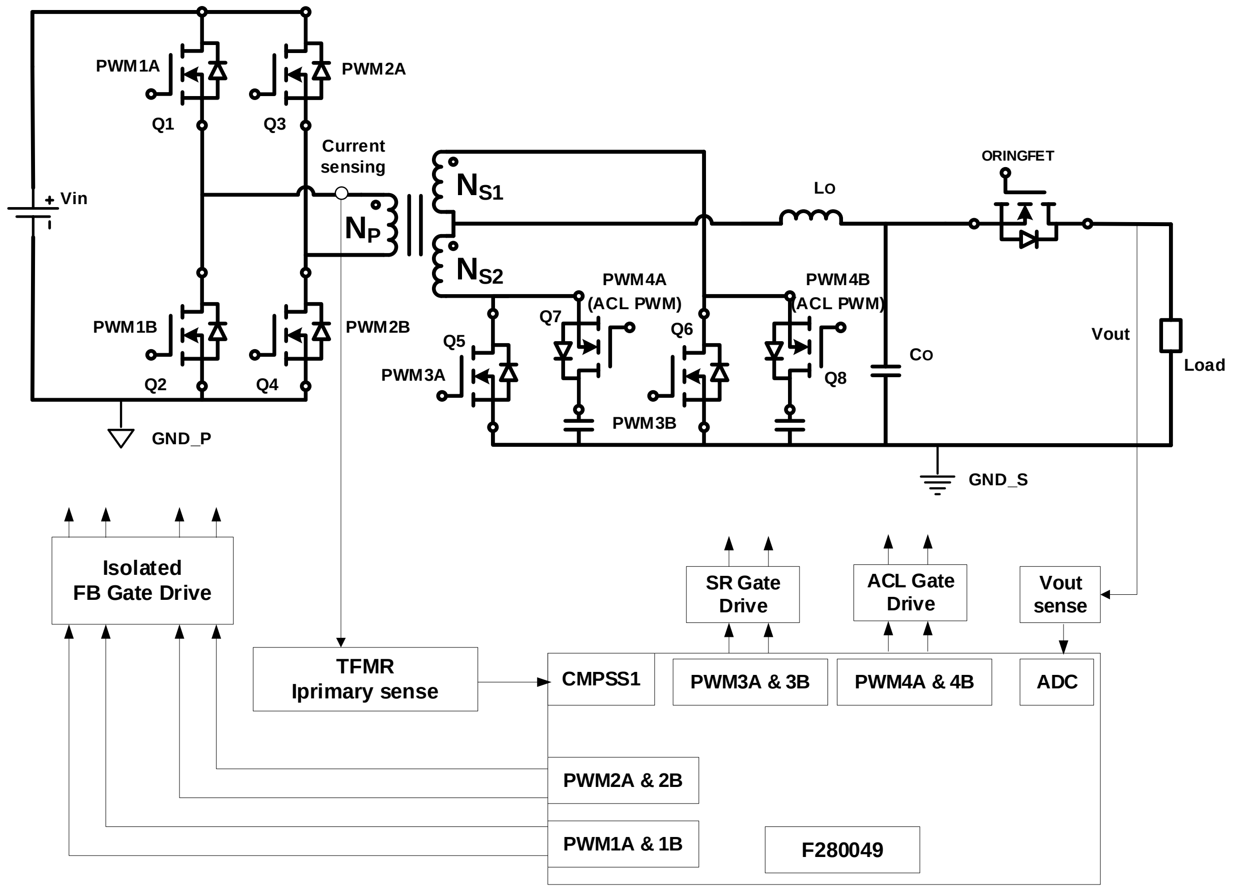


Figure 2 - F280049 Interface to PCMC PSFB DC/DC Power Stage

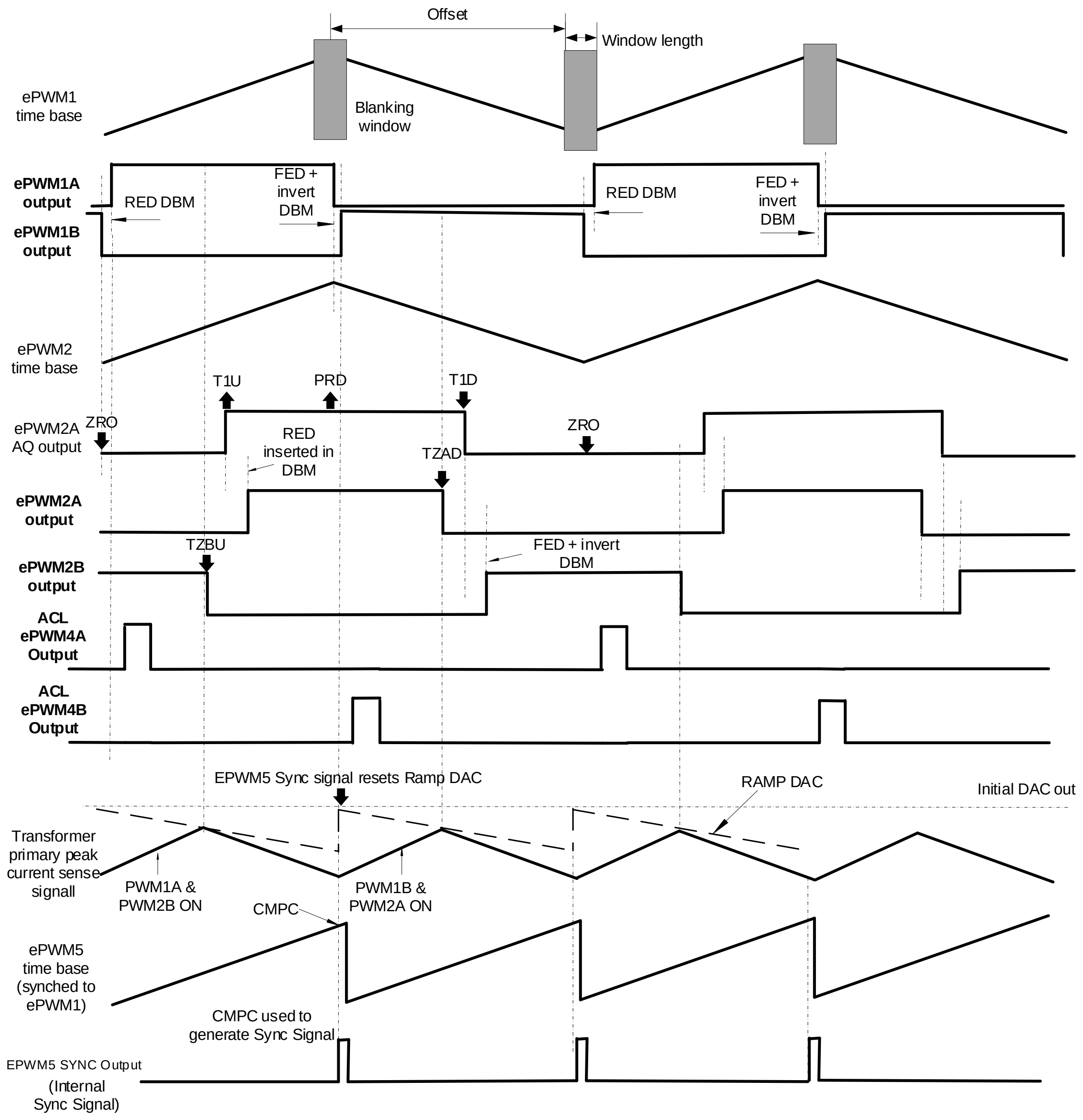


Figure 3 - PCMC PSFB DC/DC Primary Side PWM, ACL PWM and Transformer Current

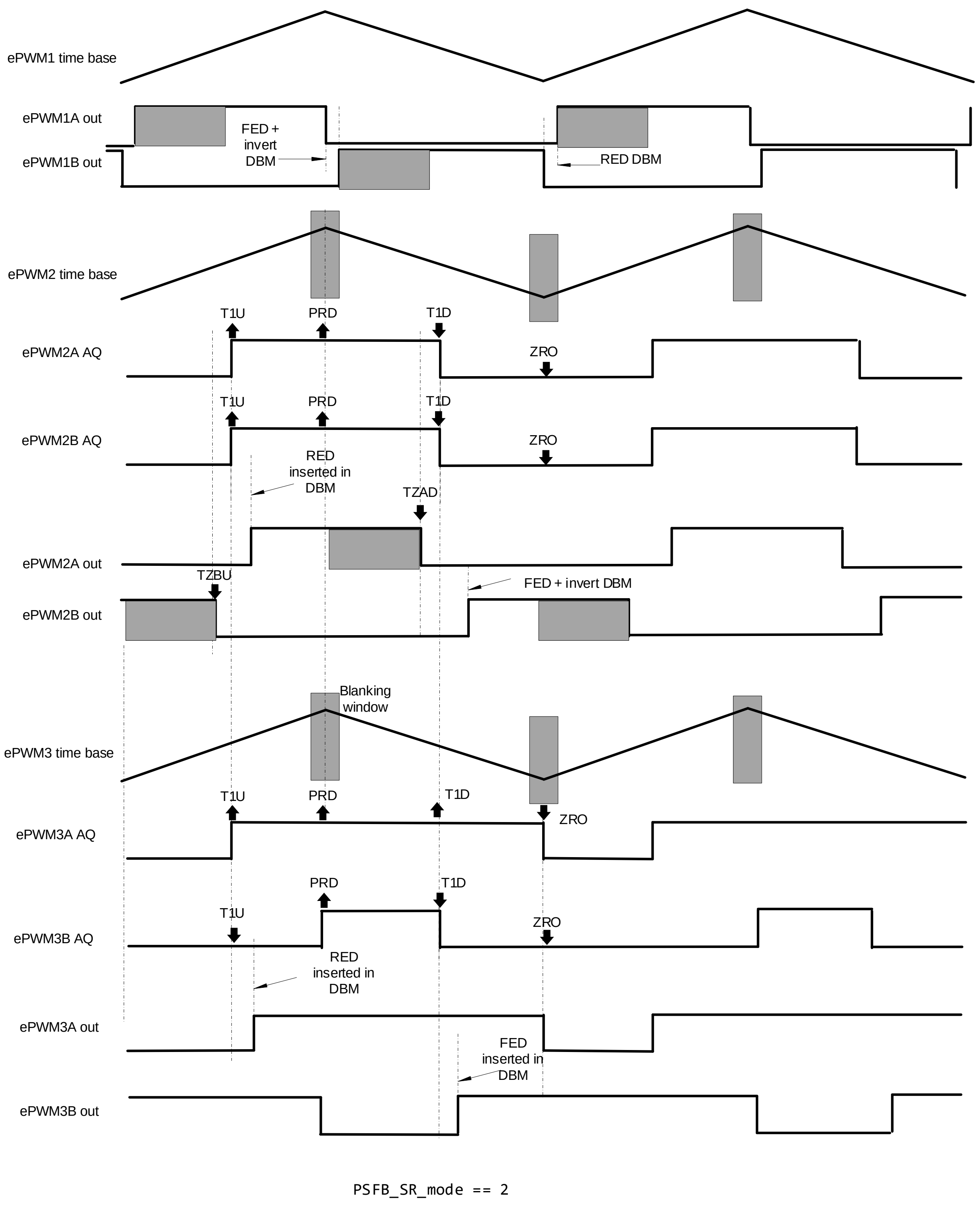


Figure 4 - PCMC PSFB DC/DC Primary and Secondary Side PWM Signals for SR Mode 2

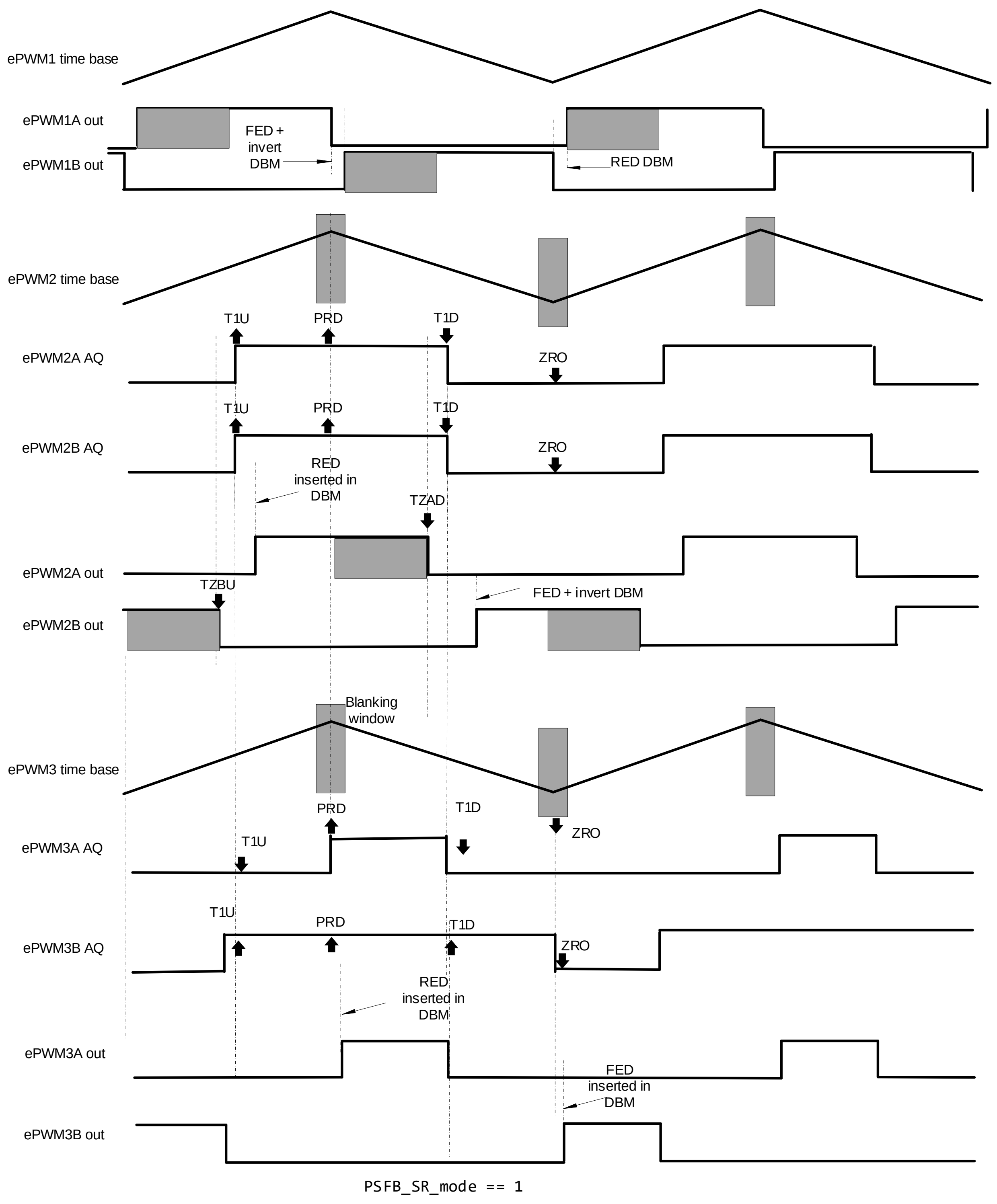


Figure 5 - PCMC PSFB DC/DC Primary and Secondary Side PWM Signals for SR Mode 1

1.2 Implementation on PMP23126 Board

This PCMC PSFB DC/DC power supply is implemented on a TI reference design PMP23126 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 12V DC. Rated output power 3kW
- Topology: Peak current mode control phase shifted full-bridge with outer voltage loop
- PWM frequency: 100 kHz
- Efficiency: 97.74% peak 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 ACL switch Q7	GPIO-06/PWM4A
LEG4_LS	PWM drive for ACL switch Q8	GPIO-07/PWM4B
ORING_EN	Enable signal for ORing switch	GPIO-13
VLV-FB	12V Output sense signal **before** the ORing FET	A5
VLV-FB1	12V Output sense signal **after** the ORing FET	A3
ILV-FB	12V 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 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.

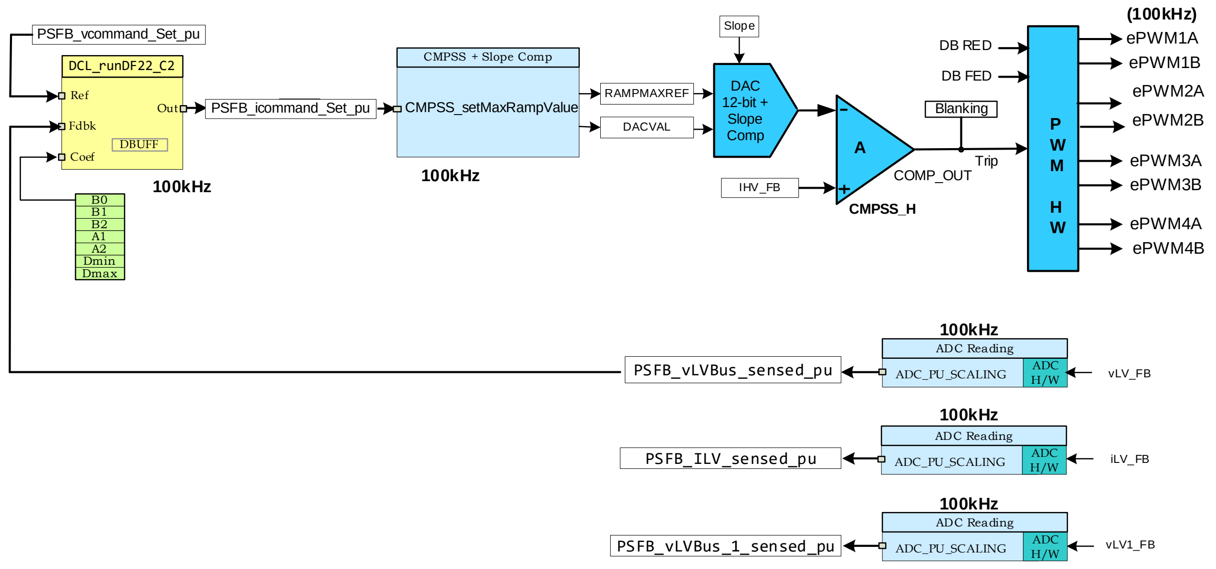


Figure 6 - Software Control Diagram for PSFB PCMC DC/DC using F28004x MCU

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. The actual DAC input register (RAMPMAXREFA) is 16bit. 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,

$R = m*\{(3.3/65535)/(1/f_{CPU})\} = m*\{(3.3/65535)/(1/100x10^6)\}$ volts/Sec

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

$R = 20*\{(3.3/65535)/(1/100x10^6)\} = 100710$ volts/Sec = 0.10071 volts/uS.

For a specific load power if the ON time (power transfer time) is Td (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 $\Delta V = (R)*(Td)$

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

$U = Vcs + \Delta V$

Because of slope compensation this value decreases by ΔV during the ON time Td and the final DAC output matches the peak feedback current sense voltage Vcs.

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.

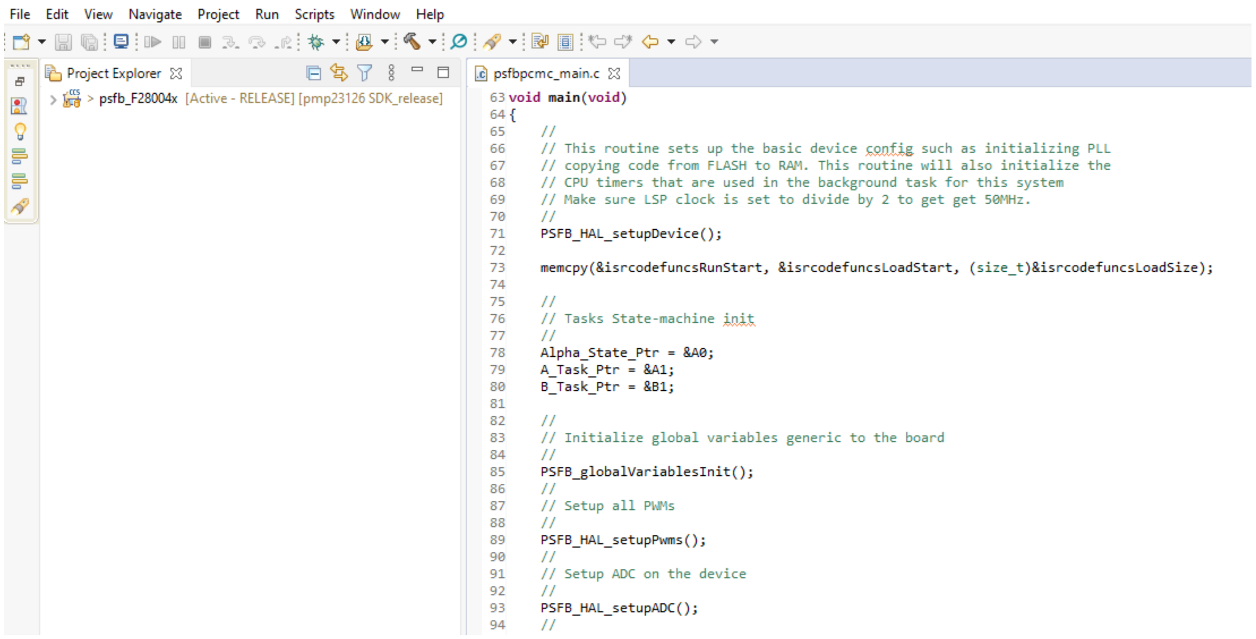
III. SR Mode 2: This mode is useful for all other higher load conditions, say 55% to 100% of rated load. In this mode, appropriate synchronous rectifier MOSFETs are turned OFF only when the corresponding opposite diagonal bridge drive signals overlap.

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.



```
void main(void)
{
    //
    // This routine sets up the basic device config such as initializing PLL
    // copying code from FLASH to RAM. This routine will also initialize the
    // CPU timers that are used in the background task for this system
    // Make sure LSP clock is set to divide by 2 to get get 50MHz.
    //
    PSFB_HAL_setupDevice();

    memcpy(&isrcodefuncsRunStart, &isrcodefuncsLoadStart, (size_t)&isrcodefuncsLoadSize);

    //
    // Tasks State-machine init
    //
    Alpha_State_Ptr = &A0;
    A_Task_Ptr = &A1;
    B_Task_Ptr = &B1;

    //
    // Initialize global variables generic to the board
    //
    PSFB_globalVariablesInit();
    //
    // Setup all PWMs
    //
    PSFB_HAL_setupPwms();
    //
    // Setup ADC on the device
    //
    PSFB_HAL_setupADC();
    //
```

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 and can be used to keep any settings related to ADC/PWM channel mapping, GPIOs etc. The file psfbpcmc_settings.h 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 fs2, where fs2 = 200kHz. 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 fs1, where fs1=100kHz. 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, performing startup sequence, controlling SR mode switching, dead-band adjustment, and controlling frequency folding under light load condition.

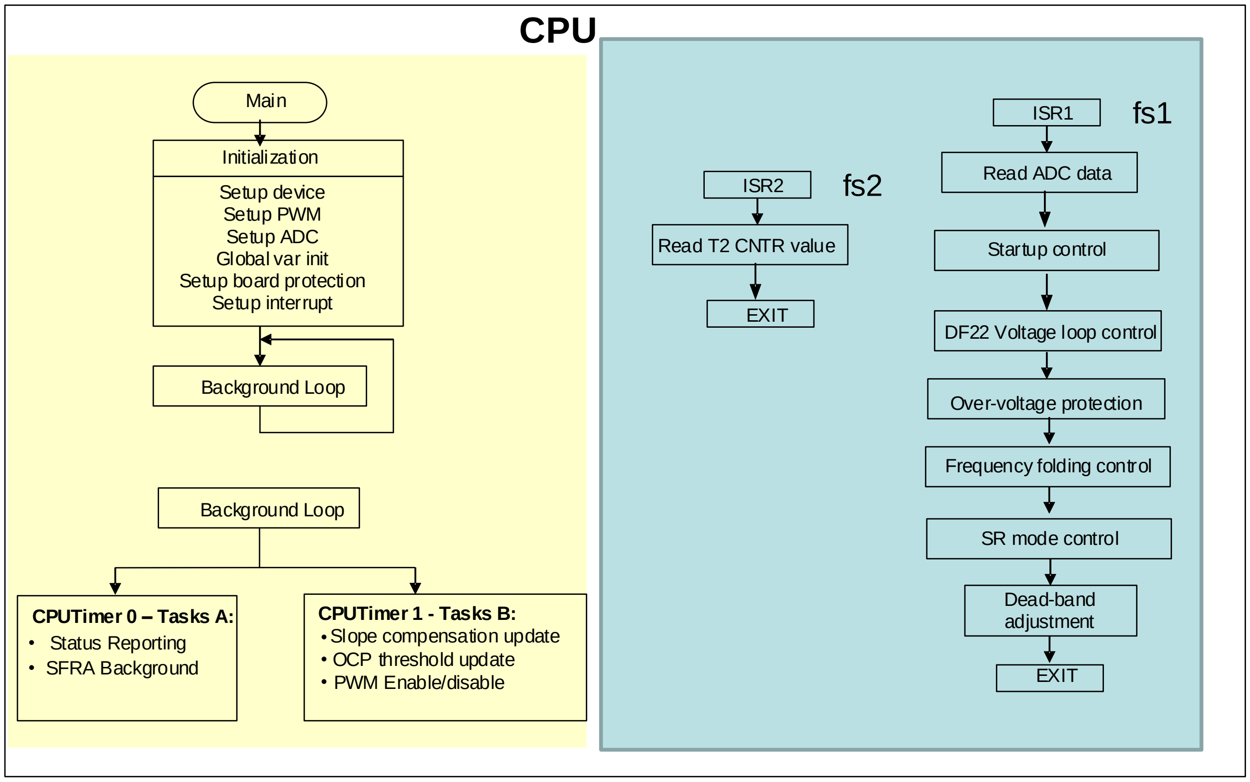


Figure 8 - Software structure for main, ISRs and background loops

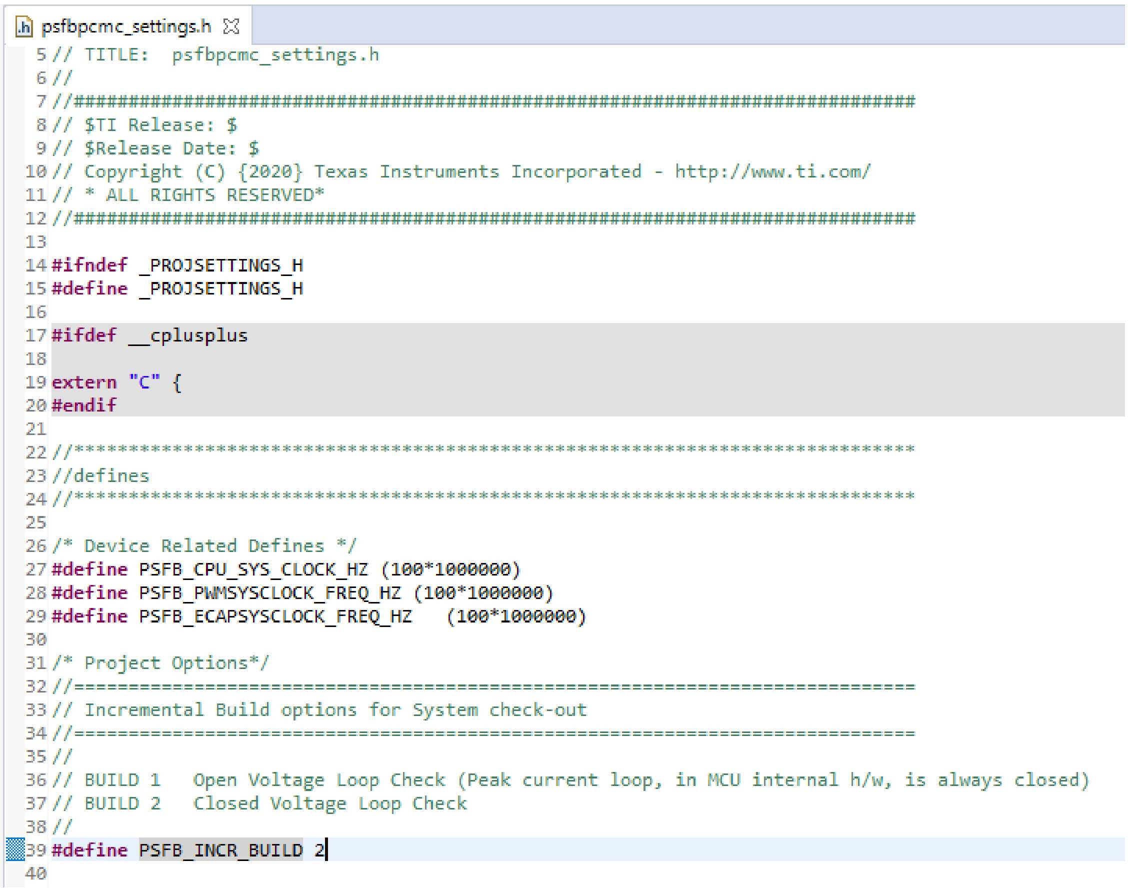
For a closed-loop system under steady-state, the peak current will only have minor changes from one control ISR cycle to the other and hence the ISR2 will be triggered repeatedly.

Additionally, CPU timers are used to trigger slow background tasks. These tasks are not interrupt-driven but polled. Background tasks includes other functions such as doing SFRA, updating board status, adjusting protection threshold and slope compensation 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 2000 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 in Figure 9. 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. 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.



psfbpcmc_settings.h

```
// TITLE:  psfbpcmc_settings.h
//
//###########################################################################
// $TI Release: $
// $Release Date: $
// Copyright (C) {2020} Texas Instruments Incorporated - http://www.ti.com/
// * ALL RIGHTS RESERVED*
//###########################################################################

#ifndef _PROJSETTINGS_H
#define _PROJSETTINGS_H

#ifdef __cplusplus

extern "C" {
#endif

//***************************************************************************
//defines
//***************************************************************************

/* Device Related Defines */
#define PSFB_CPU_SYS_CLOCK_HZ (100*1000000)
#define PSFB_PWMSYSCLOCK_FREQ_HZ (100*1000000)
#define PSFB_ECAPSYSCLOCK_FREQ_HZ   (100*1000000)

/* Project Options*/
//===========================================================================
// Incremental Build options for System check-out
//===========================================================================
//
// BUILD 1   Open Voltage Loop Check (Peak current loop, in MCU internal h/w, is always closed)
// BUILD 2   Closed Voltage Loop Check
//
#define PSFB_INCR_BUILD 2

```

Figure 9 – INCR_BUILD Setting

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.

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

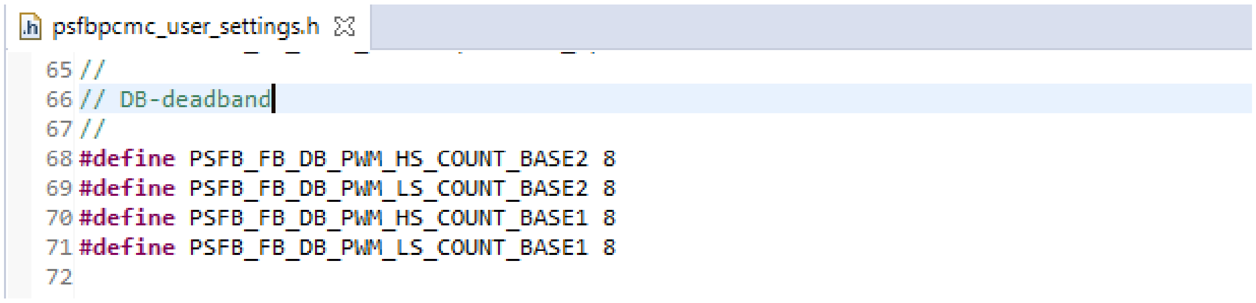
5 Procedure for running the incremental builds

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 CMPSS 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 CMPSS 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 and ePWM3B drive Q5 and Q6 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.


psfbpcmc_user_settings.h

```
//
// DB-deadband
//
#define PSFB_FB_DB_PWM_HS_COUNT_BASE2 8
#define PSFB_FB_DB_PWM_LS_COUNT_BASE2 8
#define PSFB_FB_DB_PWM_HS_COUNT_BASE1 8
#define PSFB_FB_DB_PWM_LS_COUNT_BASE1 8

```

Figure 10 – Dead-band Initial Setting

ePWM3A and ePWM3B 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 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, dead-band adjustment and automatic frequency folding.

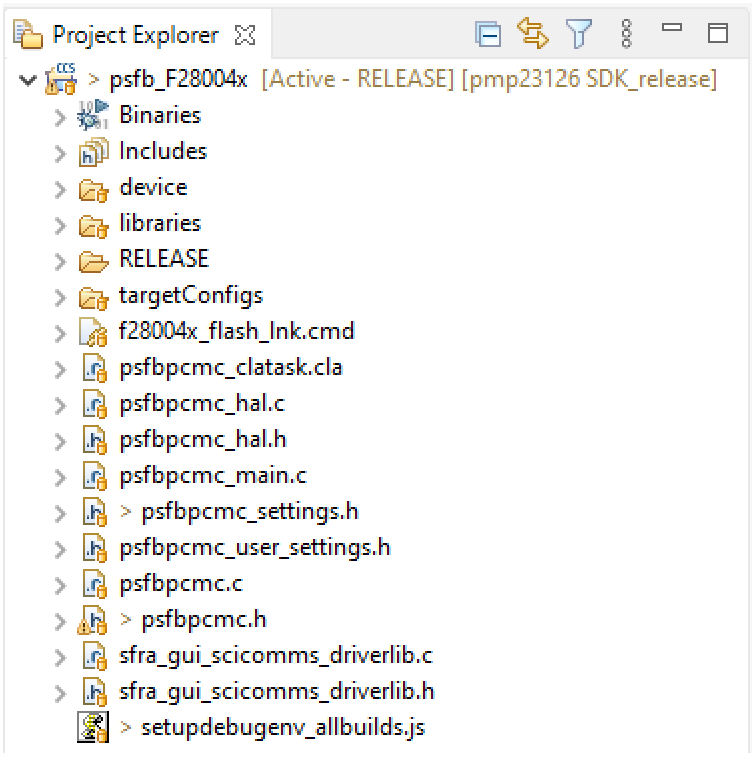
The output voltage sensing circuit is made up of simple voltage dividers. A current sensing circuit consisting of a current transformer, diodes, termination resistor and a sense resistor are used to sense the full-bridge transformer primary current.

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 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 1.84 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_psfb_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 11.



```
psfb_F28004x [Active - RELEASE] [pmp23126 SDK_release]
  Binaries
  Includes
  device
  libraries
  RELEASE
  targetConfigs
  f28004x_flash_lnk.cmd
  psfbpcmc_clatask.cla
  psfbpcmc_hal.c
  psfbpcmc_hal.h
  psfbpcmc_main.c
  > psfbpcmc_settings.h
  psfbpcmc_user_settings.h
  psfbpcmc.c
  > psfbpcmc.h
  sfra_gui_scicomms_driverlib.c
  sfra_gui_scicomms_driverlib.h
  > setupdebugenv_allbuilds.js
```

Figure 11 – Project Explorer View

4. Open the file *psfbpcmc_settings.h*. Set the project for Lab 1 by changing the PSFB_INCR_BUILD to 1. 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().*

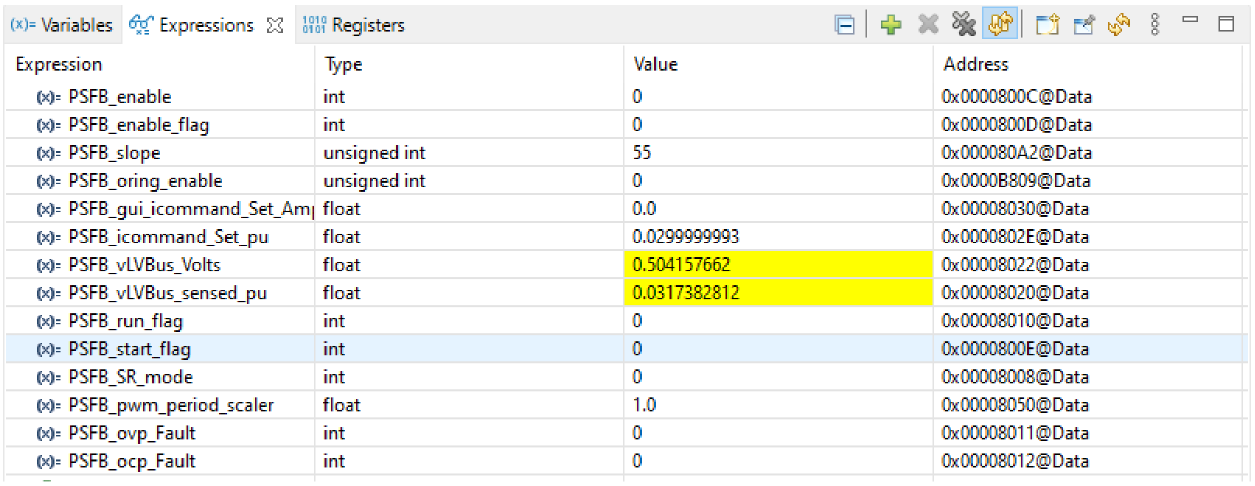
7. Populate the expressions window entries by clicking scripting console on the menu bar and then opening the setupdebugenv_allbuilds.js file from the project workspace directory (PCMC_PSFB_F28004x\ setupdebugenv_allbuilds.js) using the scripting console Open File() command. Please note that some of 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.

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 1.84 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 12.

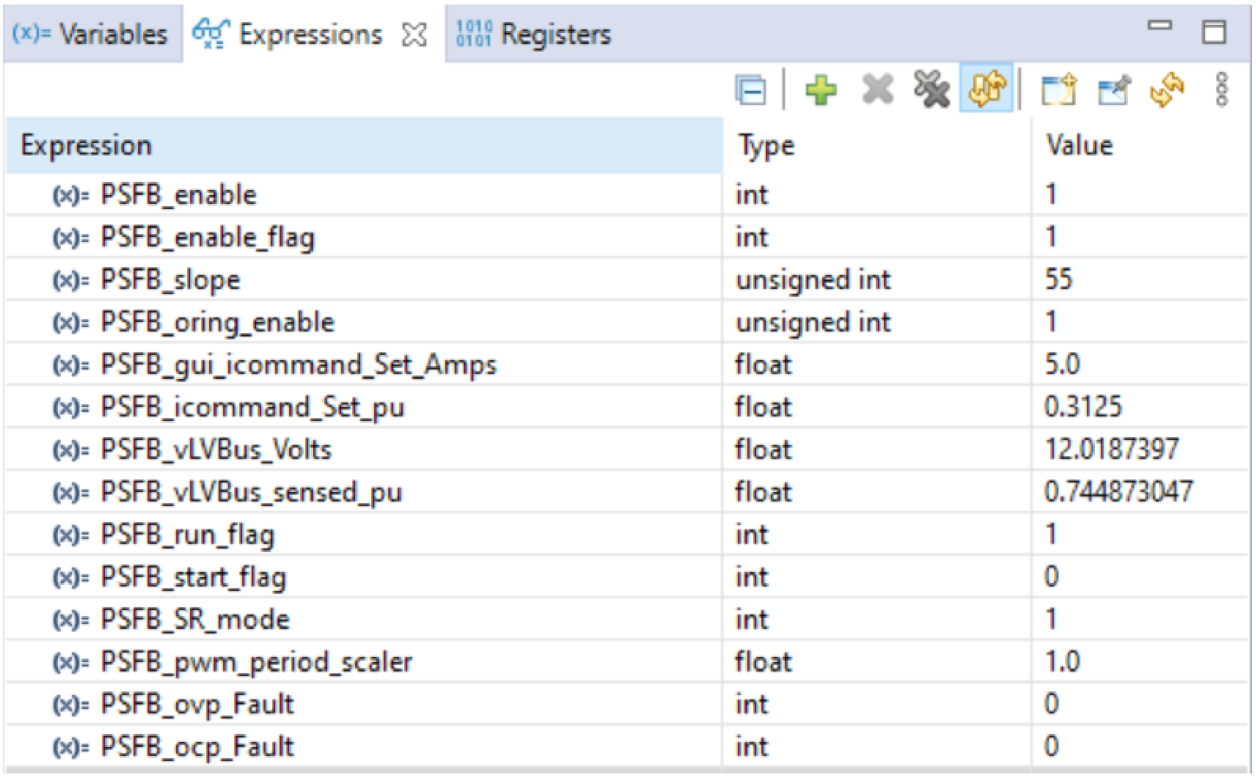


Expression	Type	Value	Address
PSFB_enable	int	0	0x0000800C@Data
PSFB_enable_flag	int	0	0x0000800D@Data
PSFB_slope	unsigned int	55	0x000080A2@Data
PSFB_oring_enable	unsigned int	0	0x0000B809@Data
PSFB_gui_icommand_Set_Am|	float	0.0	0x00008030@Data
PSFB_icommand_Set_pu	float	0.0299999993	0x0000802E@Data
PSFB_vLVBus_Volts	float	0.504157662	0x00008022@Data
PSFB_vLVBus_sensed_pu	float	0.0317382812	0x00008020@Data
PSFB_run_flag	int	0	0x00008010@Data
PSFB_start_flag	int	0	0x0000800E@Data
PSFB_SR_mode	int	0	0x00008008@Data
PSFB_pwm_period_scaler	float	1.0	0x00008050@Data
PSFB_ovp_Fault	int	0	0x00008011@Data
PSFB_ocp_Fault	int	0	0x00008012@Data

Figure 12 – 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 5A. The RT watch window in Figure 13 below shows that this is set to 5A. 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.



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	5.0
(x)= PSFB_icommand_Set_pu	float	0.3125
(x)= PSFB_vLVBus_Volts	float	12.0187397
(x)= PSFB_vLVBus_sensed_pu	float	0.744873047
(x)= PSFB_run_flag	int	1
(x)= PSFB_start_flag	int	0
(x)= PSFB_SR_mode	int	1
(x)= PSFB_pwm_period_scaler	float	1.0
(x)= PSFB_ovp_Fault	int	0
(x)= PSFB_ocp_Fault	int	0

Figure 13 - CCS Watch Window view 380Vin, RL=1.84 ohm, PSFB_gui_icommand_Set_Amps = 5A – Lab 1

13. Now apply 380 V DC input from the high voltage DC source. The output will rise to around 11V. 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 12. 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 16.13V 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 16.13V.

Use an oscilloscope to verify the output voltage, the primary side Leg 1 upper and lower PWMs, and the transformer primary current. Appropriate safety precautions should be taken while probing any high voltages and high currents for this isolated DC-DC converter. The scope capture Figure 14 below shows the output voltage (Ch 3), the primary transformer current (CH2), and the primary bridge PWM waveforms for PWM1A (Ch 1) and PWM1B (Ch 4)

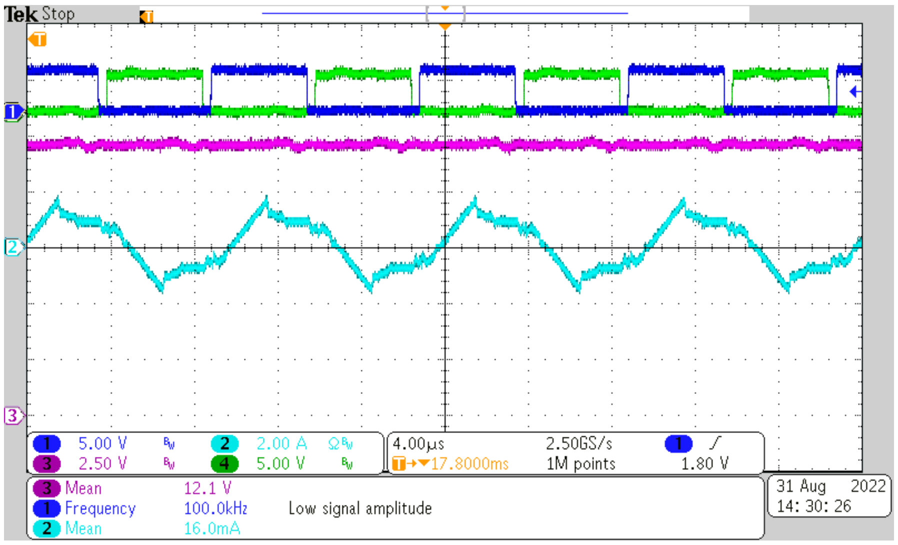


Figure 14 - Test Waveform – Lab 1

14. 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 6 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 6, the voltage loop controller is executed at a frequency of fs where, fs = 100 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.

To quickly run and verify the functionalities for this Lab2 software (s/w), follow the steps listed below:

1. Set the project for Lab 2 by changing the PSFB_INCR_BUILD to 2. This is defined in the *psfbpcmc_settings.h* file. All the other options in the *psfbpcmc_user_settings.h* file can be left at their default settings for now. Click Save.

2. Turn ON the bench power supplies to provide bias power to the board as explained in Lab 1. Click on the Debug button. The Lab 2 code should compile and load to the internal flash memory.

3. 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 the function *main().*

4. Populate the expressions window entries by clicking scripting console on the menu bar and then opening the setupdebugenv_allbuilds.js file from the project workspace directory (PCMC_PSFB_F28004x\ setupdebugenv_allbuilds.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.

5. 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.

6. Run the code by using the Run button on the toolbar.

7. Set the electronic load to constant current (CC) mode and use a current value of 10A.

8. 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.

9. Now slowly apply 380 V DC input from the high voltage DC source. The output voltage reading should be almost zero. Now from the watch window, set the variable PSFB_start_flag to 1. The output will start to rise to 12V, as in Figure 15. The steady state waveform is given in Figure 16. Observe the watch window variable PSFB_vLVbus_Volts for the final output voltage, which should be about 12V. 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.

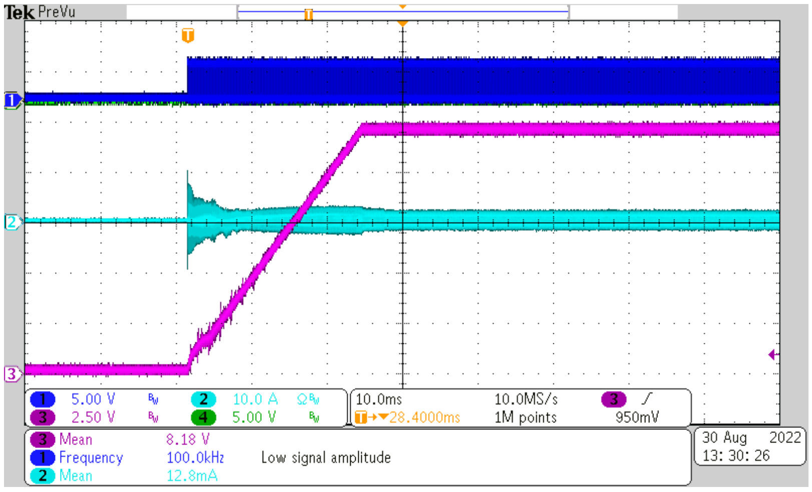


Figure 15 – Startup Waveform at 10A CC – Lab 2

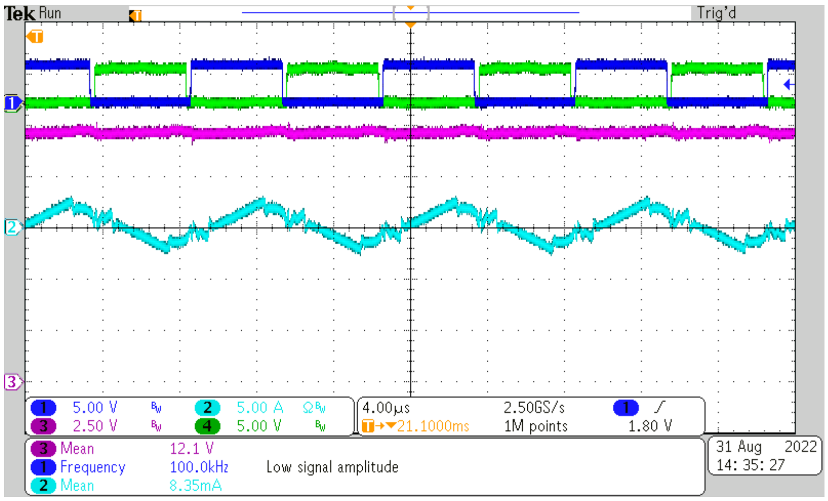
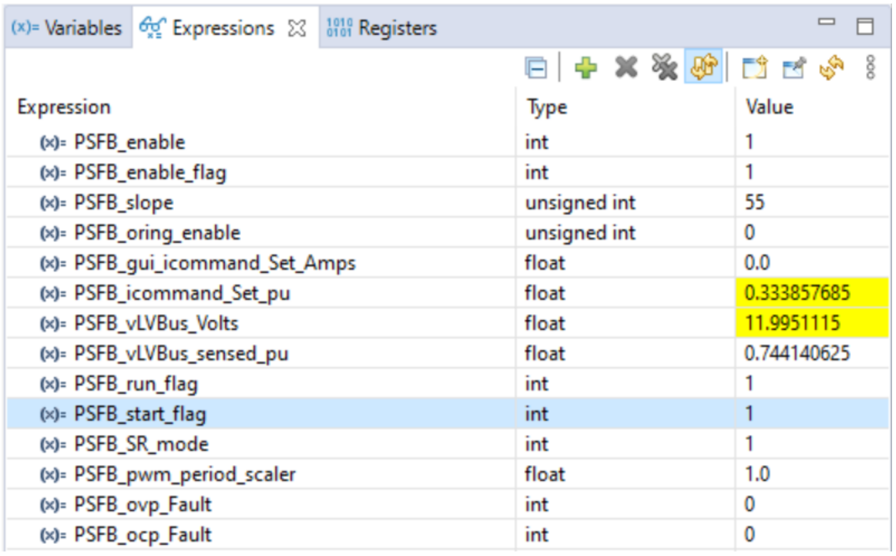


Figure 16– Steady state waveform at 10A CC – Lab 2



Expression	Type	Value
PSFB_enable	int	1
PSFB_enable_flag	int	1
PSFB_slope	unsigned int	55
PSFB_oring_enable	unsigned int	0
PSFB_gui_icommand_Set_Amps	float	0.0
PSFB_icommand_Set_pu	float	0.333857685
PSFB_vLVBus_Volts	float	11.9951115
PSFB_vLVBus_sensed_pu	float	0.744140625
PSFB_run_flag	int	1
PSFB_start_flag	int	1
PSFB_SR_mode	int	1
PSFB_pwm_period_scaler	float	1.0
PSFB_ovp_Fault	int	0
PSFB_ocp_Fault	int	0

Figure 17 - CCS Watch Window view Lab 2

10. 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.