Digitally Controlled, 2-Phase Interleaved Power-Factor Correction (ILPFC) Converter Using C2000™ Piccolo-A Microcontroller

User's Guide

Literature Number: SPRUI55
February 2016

User's Guide
SPRUI55–February 2016

Digitally Controlled, 2-Phase Interleaved Power-Factor Correction (ILPFC) Converter Using C2000™ Piccolo-A Microcontroller

Abstract

This guide presents the implementation details of a digitally controlled, 2-phase interleaved power-factor correction (ILPFC) converter with an integrated powerSUITE™ tool. A C2000™ Piccolo-B control card and a 700-W ILPFC EVM are used to implement the complete system.

With various regulations limiting the input current harmonic content (especially with the IEC 61000-3-2 standard that defines the harmonic components that an electronic load may inject into the supply line), a power-factor correction (PFC) stage has become an integral part of most rectifier designs. The PFC stage usually forms the front end of an isolated AC-DC rectifier system, as shown in Figure 1.

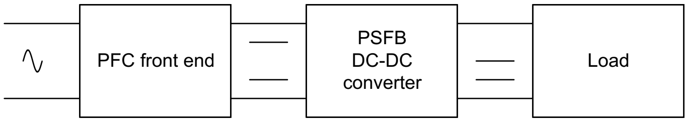


Figure 1. Isolated AC-DC Rectifier Block Diagram

The PFC converter provides power to nonlinear loads from the AC mains while maintaining AC-input current of the same wave-shape and phase of the AC-mains voltage. The PFC converter regulates its output DC voltage to provide a regulated high-voltage bus to any downstream DC-DC converter connected to it. The downstream DC-DC converter is usually a phase-shifted full bridge (PSFB) converter that converts the high-DC bus voltage from the PFC stage to a lower voltage such as 12 V or an intermediate distribution voltage (closer to 48 V). This guide focuses on the implementation details of the PFC stage. This guide elaborates on the digital control implementation of the PFC stage with the powerSUITE tool. This tool from TI allows for the following:

- A GUI-based, PFC control-loop compensation design
- A control-loop frequency response measurement using software frequency response analyzer (SFRA)
- A solution adapter for adapting to TI ILPFC solution

This guide provides details on using this ILPFC board and the accompanying software with the different powerSUITE tools. This guide uses a step-by-step method to explain the content, starting with open-loop operation and working towards a complete closed-loop PFC system.

NOTE: This PFC EVM comes with a Piccolo-B control card. The controller resources used for the PFC implementation show that other low-pin count Piccolo™ microcontrollers (MCUs) can also be used to implement full control of the PFC stage.

1 Introduction

The PFC stage converts the AC-mains voltage to a regulated DC-bus voltage, while drawing a sine wave input current in phase with the AC-input voltage. This functionality is implemented using a bridge rectifier followed by a boost PFC stage. With its on-chip pulse width modulation(PWM) and analog-to-digital converter (ADC) modules, a C2000 Piccolo™ MCU can control an ILPFC system digitally.

1.1 PFC Stage Implementation

Figure 2 shows a C2000 controller based on an ILPFC converter control system. The input AC voltage is applied to the PFC converter through the input EMI filter, an inrush current limit, and a bridge rectifier. The PFC stage consists of two interleaved boost converters operating at 200 kHz and phase shifted by 180°.

Inductor L1, MOSFET switch Q1, and diode D1 form one boost stage. L2, Q2, and D2 form the other boost stage. A capacitor Cb at the boost converter output acts as an energy reservoir. This energy reservoir and closed-loop PFC control provide regulated DC voltage to the PFC-load RL.

Figure 2 indicates all the interface signals required for full control of this ILPFC converter using a C2000 MCU. The MCU controls the hardware using four feedback signals and two PWM outputs. The signals sensed and directed to the MCU include the line, neutral voltages (Vin_L and Vin_N), the PFC input current (Irect), and the DC-bus output voltage (Vbus). These sensed signals implement the voltage and current-control loops for this ILPFC converter. For phase-current balancing, two PFC switch currents (Isw1 and Isw2) can also be monitored. This EVM does not implement this feature.

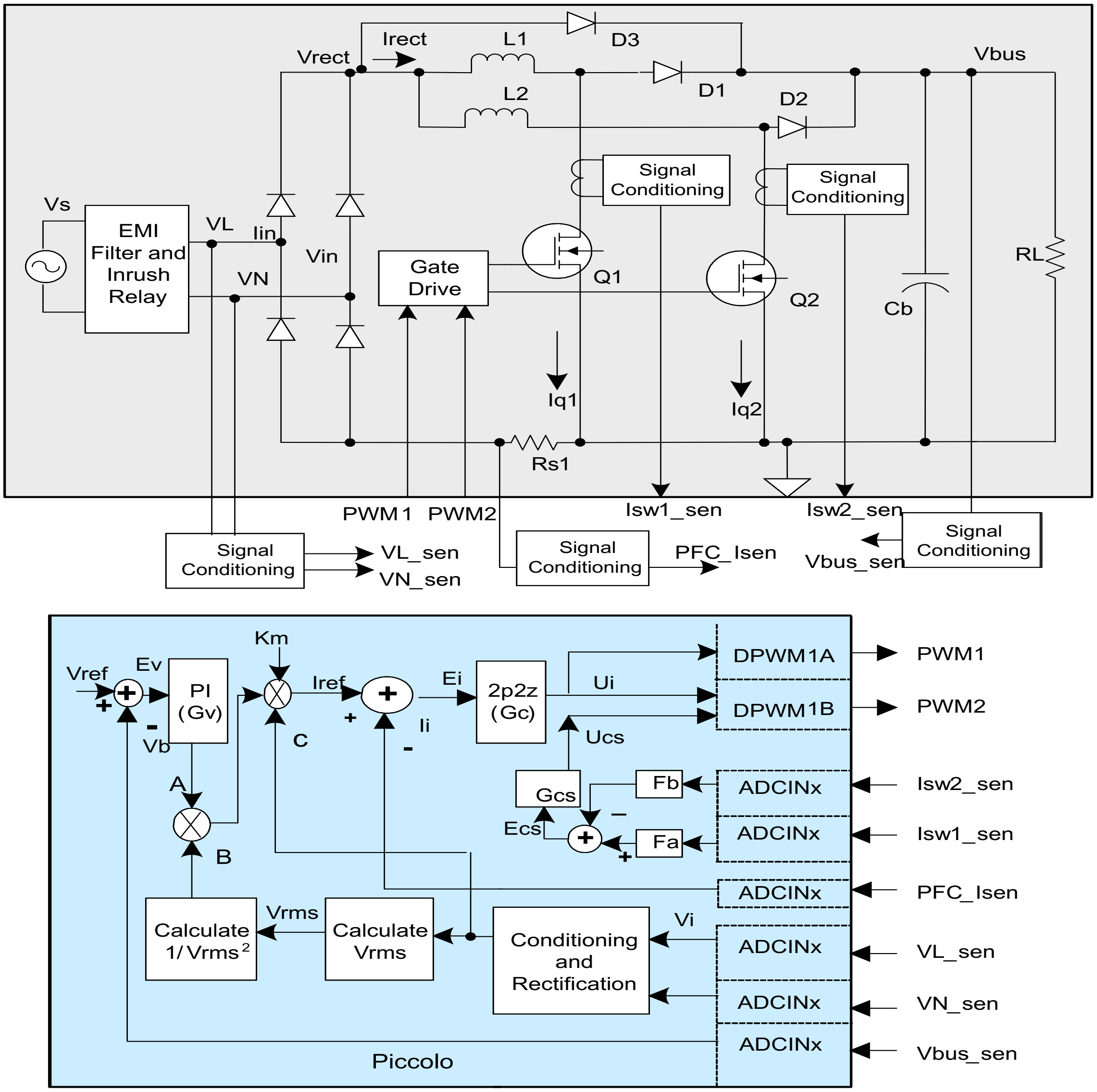


Figure 2. Interleaved PFC Converter Control Using C2000 MCU

Sensed through one of the ADC channels, the DC-bus voltage Vbus is compared with the reference bus voltage Vref. The error signal Ev is input to the voltage-loop controller Gv that regulates the bus voltage at the reference level to minimize Ev. The voltage controller Gv is a 2-pole, 2-zero (2P2Z) compensator.

Noted by the letter A in Figure 2, the output of Gv is proportional to the power transfer by the PFC converter. This output A is multiplied by three parameters (indicated by B, C, and Km in Figure 2) to form the reference current command Iref for the PFC current control loop. The signal indicated by B is the inverse of the square of the RMS input voltage, which also enables fast feed-forward control of the PFC system. The signal C is proportional to the rectified input voltage that modulates the voltage controller output A so that the PFC input current has the same shape as that of the PFC input voltage. The parameter Km is the multiplier gain that is used to adjust the range of Iref corresponding to the full-input voltage range of the PFC converter. The output of the multiplier block provides the reference signal Iref that is used to control the total average inductor current (that is, the PFC input current). This reference current command Iref for the PFC current control loop is compared to the sensed PFC input current Ii sensed through one ADC channel. The resulting current error signal Ei is input to the current-loop controller Gc that generates the PFC duty ratio command d so that the PFC input current tracks the reference current Iref.

In addition to implementing the voltage and current-loop controllers, the C2000 MCU also uses the sensed line and neutral voltage signals to determine the polarity of the input voltage (+ve and –ve half cycle) and calculates the rectified input voltage, the RMS input voltage, RMS input power, and the input-line frequency. These time-critical functions are implemented in a fast-sampling loop enabled by the C2000 MCU high-speed CPU, interrupts, on-chip 12-bit ADC module, and high-frequency PWM modules. This user's guide provides a detailed description of the software algorithm in the following chapters.

1.2 ILPFC Electrical Specifications

The following is a list of the key highlights of the C2000 ILPFC EVM:

- Input AC voltage: 100–260 V (maximum), 47–63 Hz, output voltage 390 VDC
- Rated output 700 W at 220-V input, 550 W at 110-V input
- Full load efficiency: 97% at 220-V input
- Power factor at 50% or greater load: 0.98 (minimum)
- Input power monitoring

2 Software Overview

2.1 Software Control Flow

The Code Composer Studio™ (CCS) project for C2000 powerSUITE™ interleaved PFC (ILPFC) uses of the *C-background/C-ISR* framework and C-callable assembly functions. The main fast ISR (100 kHz) runs in C environment. The C-callable assembly functions are called from this fast C-ISR. A second slower ISR (10 kHz) is also run from C environment. This slow ISR is made interruptible by the fast ISR. Figure 3 shows the IL PFC software flow diagram.

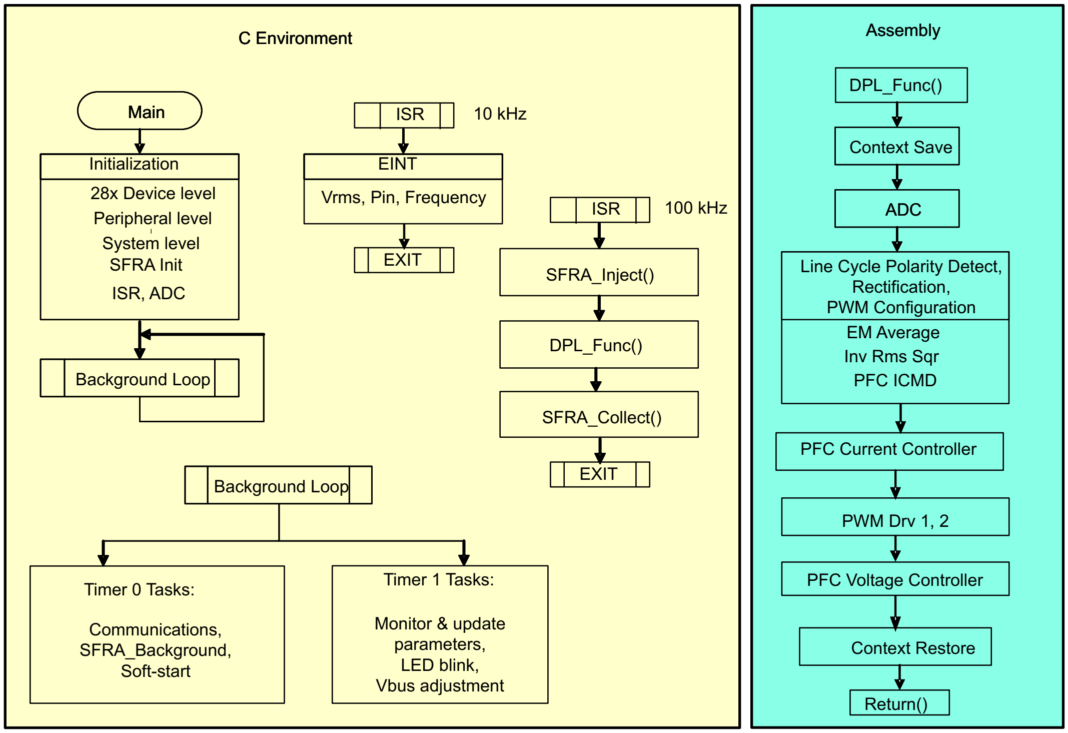
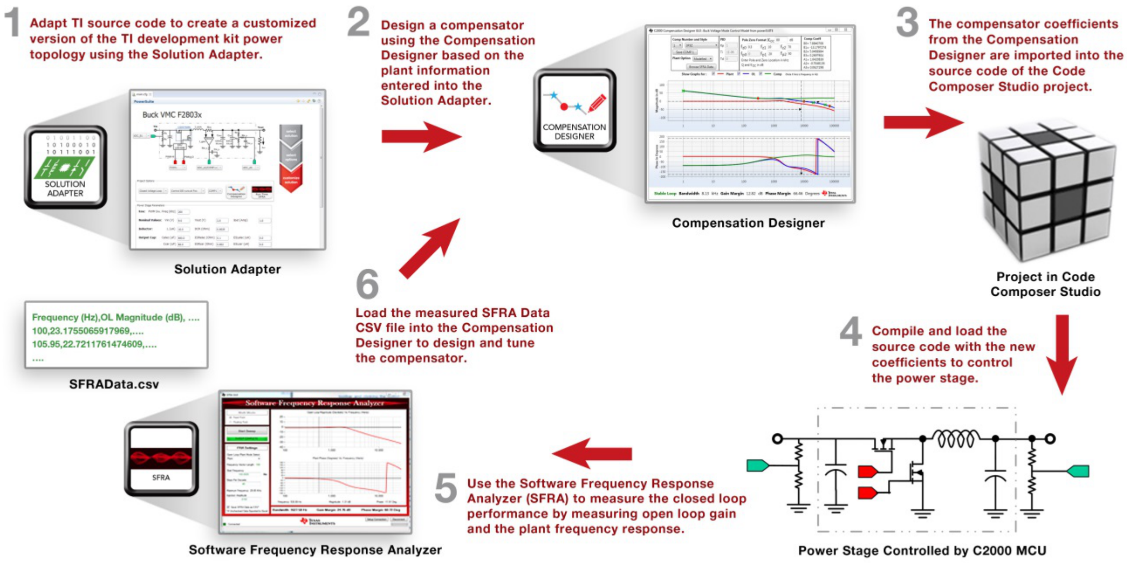


Figure 3. IL PFC Software Flow Diagram

The CCS project uses C-code as the main supporting program for the application and is responsible for system management tasks, monitoring, decision making, intelligence, and host interaction. The assembly function is limited to the fast ISR (C-ISR) runs the critical control code. These tasks include reading ADC values, input-line cycle polarity detect, sensed-line volt rectification, control calculations, and PWM updates. The slower ISR in the C environment calculates the RMS input voltage, RMS input current, RMS input power, and frequency of the input-line voltage. Figure 3 shows the general software flow for this project. Note the references to SFRA library functions SFRA_Inject() and SFRA_Collect().

The TI SFRA library is designed to enable frequency response analysis of digitally controlled power converters. This tool is implemented completely in software. This implementation enables performance of SFRA of the power converter without external connections or equipment. The optimized SFRA library can be used in high-frequency power conversion applications to identify the frequency response characteristics of the power stage (plant) and the control loop (loop-gain Bode plots) under a closed-loop power converter operation. The control-loop stability margin information such as bandwidth, gain margin, and phase margin can be determined to evaluate the performance of the digital-control loop. For more information, see the TI SFRA library documentation.

In addition to SFRA, this kit supports the use of other powerSUITE tools including the compensation designer and the solution adapter. These tools help users evaluate the complete system, adapt it for their applications, and tune it to improve performance. Figure 4 describes the process flow for designing and tuning such a system using the powerSUITE tools.


1 Adapt TI source code to create a customized version of the TI development kit power topology using the Solution Adapter.

2 Design a compensator using the Compensation Designer based on the plant information entered into the Solution Adapter.

3 The compensator coefficients from the Compensation Designer are imported into the source code of the Code Composer Studio project.

4 Compile and load the source code with the new coefficients to control the power stage.

5 Use the Software Frequency Response Analyzer (SFRA) to measure the closed loop performance by measuring open loop gain and the plant frequency response.

6 Load the measured SFRA Data CSV file into the Compensation Designer to design and tune the compensator.

Frequency (Hz),OL Magnitude (dB), ...
100,23.1755065917969,....
105.95,22.7211761474609,....
....

SOLUTION ADAPTER
Buck VMC F2803x
Solution Adapter
COMPENSATION DESIGNER
Compensation Designer
Project in Code Composer Studio
SFRAData.csv
SFRA
Software Frequency Response Analyzer
Software Frequency Response Analyzer
Power Stage Controlled by C2000 MCU

Figure 4. Using PowerSUITE to Adapt, Design, and Tune a Digital Control Loop

The solution adapter tool lets users to adapt existing code examples from TI digital power supply kits (that is, the ILPFC kit in this case) and configure them to run on their custom digital ILPFC board using the same topology and similar resources (PWM outputs, ADC channels, GPIOs, and so forth). The GUI provides you guidance through the process of selecting the solution to adapt, selecting the relevant options for that solution, and customizing those options to adapt the software solution to the desired custom hardware design.

The compensation designer tool allows the design of varying styles of compensators to achieve the desired closed-loop performance. Using the measured power stage frequency response data from the SFRA tool or the modeled power stage transfer function that comes as part of the solution adapter, this compensator can be designed. The digital controller coefficients that must be programmed on the C2000 MCU device are generated by the solution adapter and can be copied into the code directly. The default software project uses a 2P2Z controller for both the PFC current and voltage loops.

The key framework C files used in this project are as follows:

ILPFC-Main.c —This file is used to initialize, run, and manage the application.

IPFC-DevInit_F2803x.c —This file is used for 2803x controller initialization. A 2803x control card is provided with the ILPFC EVM. This file is responsible for a 1-time initialization and configuration of the F280xx device, and includes functions such as setting up the clocks, PLL, GPIO, and so forth.

The fast C-ISR consists of the following file:

ILPFC-DPL-ISR.asm —This file contains all time critical *control type* code in a C-callable assembly function _DPL_Func(). This file has an initialization section (1-time execute) and a run-time section which executes at half the rate (100 kHz) as the PWM time-base (200 kHz) used to trigger it.

The slow C-ISR consists of the following file:

SineAnalyzer.h— This file contains code for calculating the RMS voltage, RMS input current, RMS input power, and frequency of the input-line voltage. This file has an initialization section (1-time execute) and a run-time section which executes at 10-kHz rate.

The power library functions (modules) are *called* from the fast ISR framework. Library modules may have both a C and an assembly component. In this project, seven library modules are used. Table 1 lists the C and corresponding assembly module names.

Table 1. Library Modules

C Configure Function	ASM Initialization Macro	ASM Run-Time Macro
PWM_2ch_UpDwnCnt_Cnf.c	PWMDRV_2ch_UpDwnCnt_INIT n	PWMDRV_2ch_UpDwnCnt n
ADC_SOC_Cnf.c	ADCDRV_1ch_INIT m,n,p,q	ADCDRV_1ch m,n,p,q
	PFC_InvRmsSqr_INIT n	PFC_ InvRmsSqr n
	MATH_EMAVG_INIT n	MATH_EMAVG n
	PFC_ICMD_INIT n	PFC_ICMD n
	CNTL_2P2Z_INIT n	CNTL_2P2Z n

Figure 5 shows the modules.

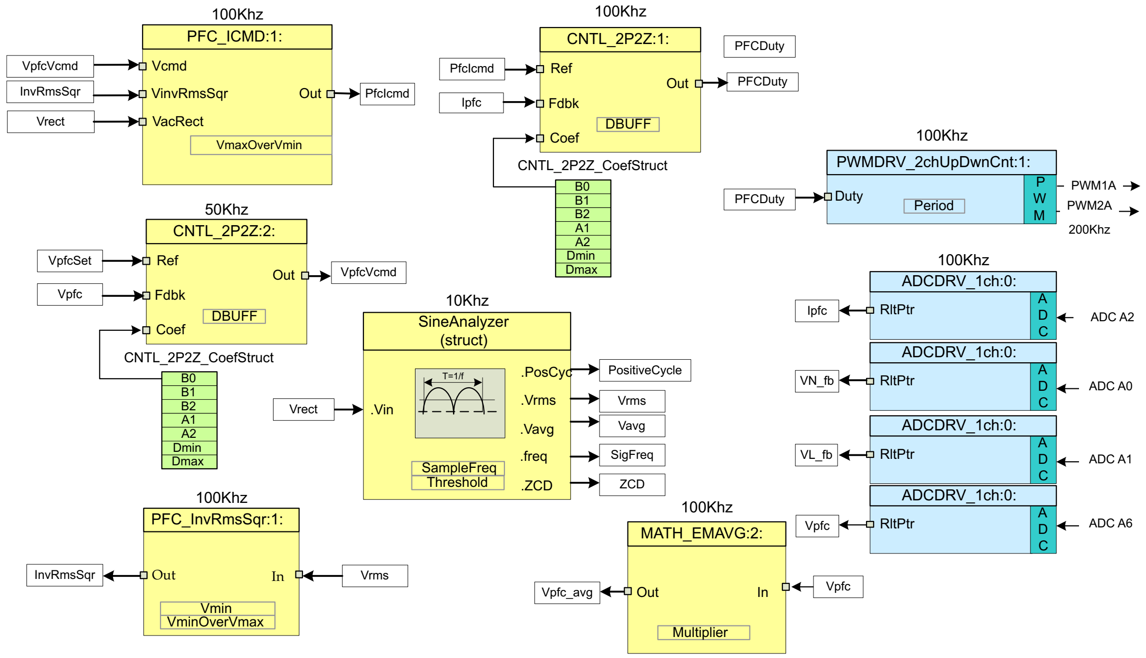


Figure 5. C2000 ILPFC Software Modules

Note the color coding used for the modules in Figure 5. The dark blue blocks represent the on-chip hardware modules in the C2000 controller. The blue blocks are the software drivers associated with these modules. The yellow blocks are part of the computation carried out on various signals. The controllers used for voltage and current loops have the form of a 2P2Z compensator. These controllers can be PI, PID, 3-pole 3-zero (3P3Z), or other controllers for the application. The modular library structure makes it convenient to visualize the complete system software flow. as shown in Figure 5. The library also allows for easy use and modifications of various functions. This fact is demonstrated in this project by implementing an incremental build approach. This use and modification is discussed in more detail in Section 3

The ILPFC system is controlled by two feedback loops. The outer-voltage loop regulates the DC-bus voltage, while a faster inner-current loop shapes the input current to maintain a high-input power factor. Figure 6 also shows the rate at which the software modules are executed. For example, for TI ILPFC kit the current controller is executed at a rate of 100 kHz (half of the PWM switching frequency) while the voltage controller is executed at a rate of 50-kHz.

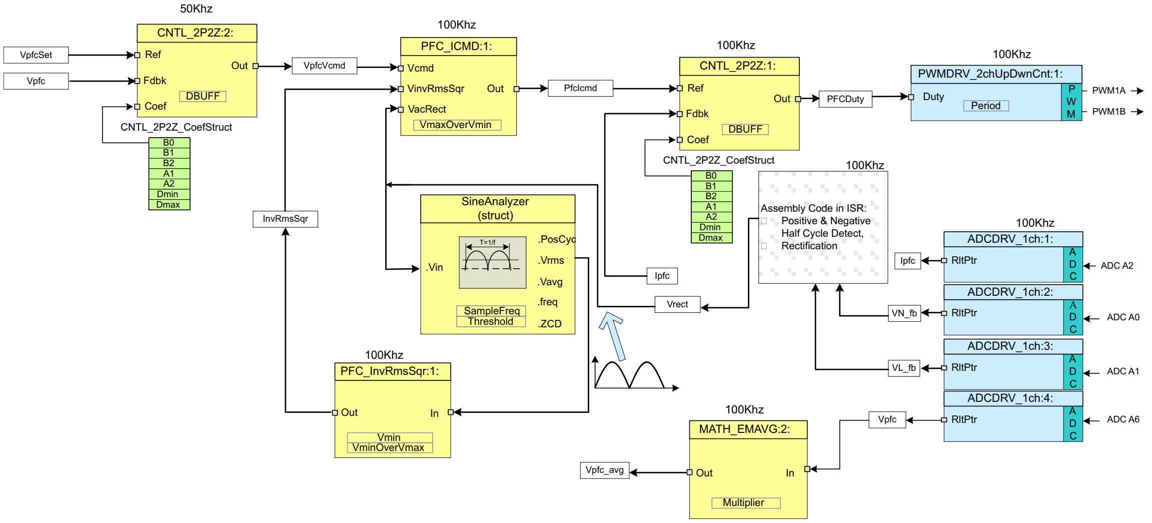


Figure 6. C2000 ILPFC Software Control Flow

3 Procedure to Start CCS and Open the PFC powerSUITE Project

Do as follows to open the project:

1. Connect USB connector to the Piccolo controller board for emulation.
2. Connect the 12-V bias supply output (external bias supply provided with the PFC EVM) to JP1.
3. Set the SW1 switch to the *Ext* position to apply this bias voltage to the board.

NOTE: The Piccolo control card jumpers (see the Piccolo control card documentation) are configured so that the device boot from Flash. Change these jumper settings to allow code execution under CCS control.

4. Change the jumper settings to allow code execution under CCS control.
5. Start CCS.
6. Select a workspace folder for the ILPFC project.
7. When CCS opens, click View→Resource Explorer.
8. From the *TI Resource Explorer* tab, click the arrow to the left of *controlSUITE*.
9. Click the arrow to the left of the *English* folder.
10. Click the arrow to the left of the *powerSUITE* folder.
11. Click *Solution Adapter*.
12. Right-click *PFC* from the *Solution Adapter* page.
13. From the *Interleaved Power Factor Correction* page, right-click on *Power-Factor Correction (PFC)*.
14. Select *Save the project into workspace*.

15. Click *OK*.

NOTE: CCS imports the PFC project and displays on the screen as Figure 7 shows.

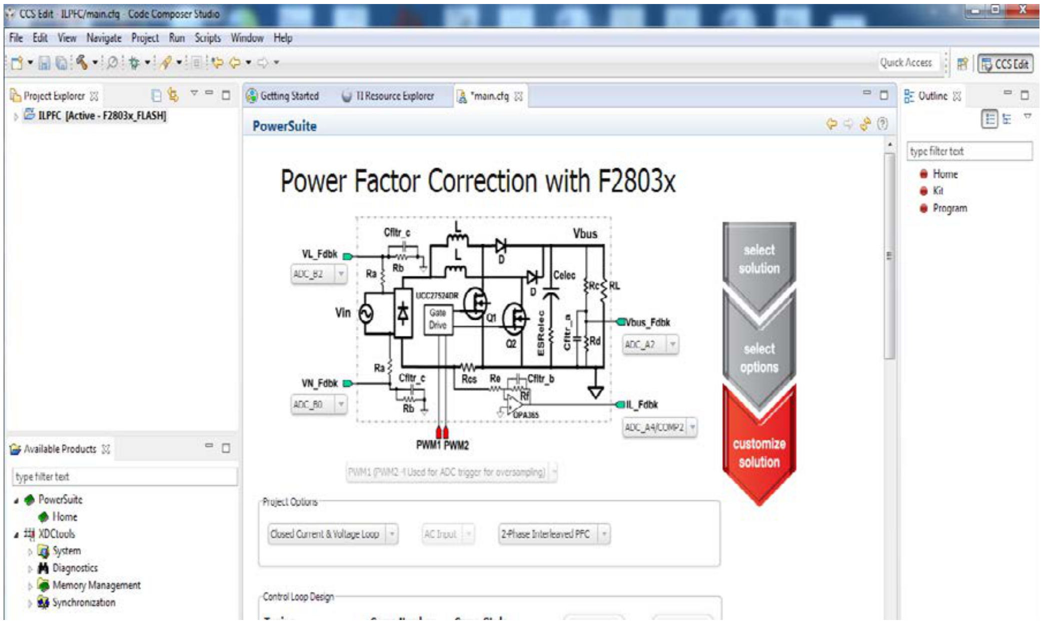


Figure 7. C2000 ILPFC PowerSUITE Project View

16. In the *Project Explorer* window, click the arrow sign to the left of ILPFC project.

NOTE: Your project window should look like Figure 8.

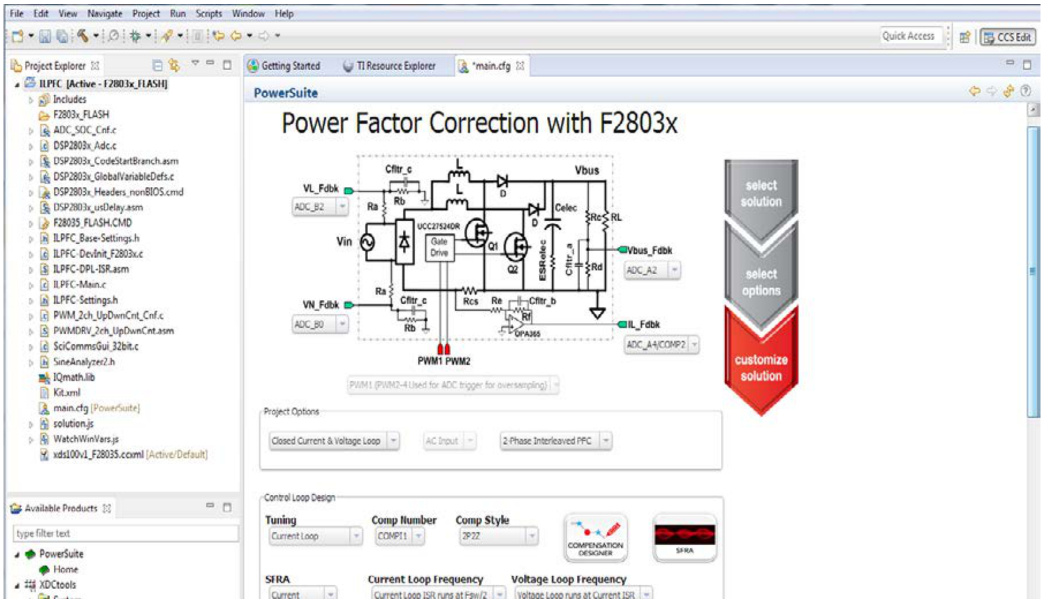


Figure 8. CCS Project Window

17. Click the main.cfg file if not open.

The file is the solution adapter GUI for the ILPFC kit. The solution adapter tool lets users adapt existing code examples from TI on the digital ILPFC kit and configure them to run on their custom digital ILPFC board that uses the same PFC topology and similar C2000 resources. Explore the GUI from the main.cfg file to become familiar with the options it provides.

As shown in Figure 8, this GUI displays the interleaved PFC topology used in the TI ILPFC kit and shows the PWM and ADC channels in this design. The associated pulldown boxes let users change these PWM and ADC channels when designing a custom PFC board. The *Solution Adaptor* page also shows *Project Options*, *Control Loop Design* options, and *Power Stage Parameter* section.

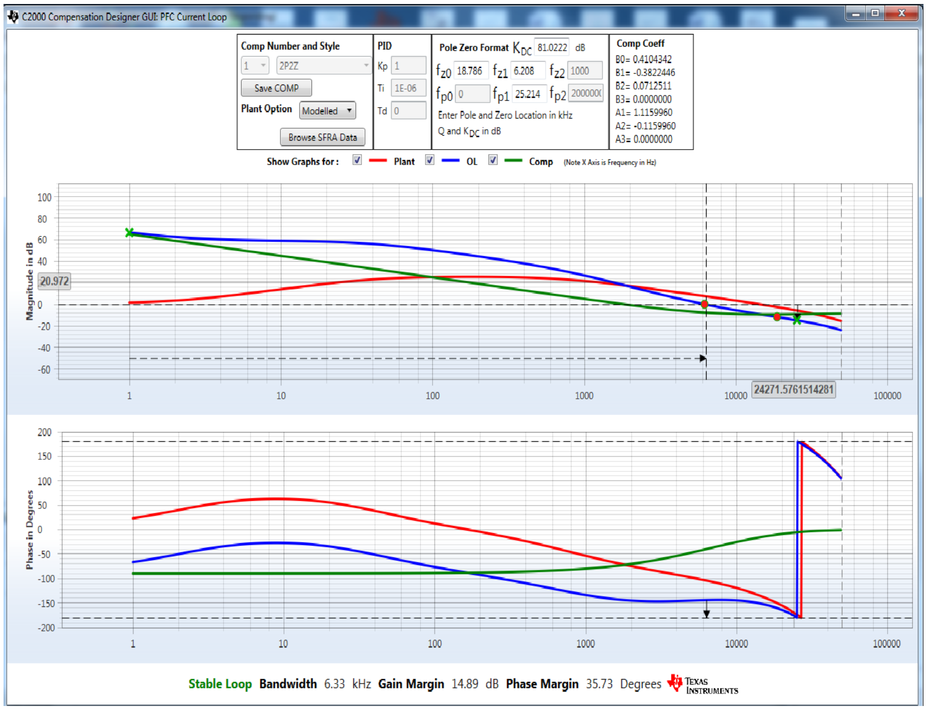
The *Project Options* section shows three different pulldown boxes that allow for customization of the PFC design. The first box lets the user to select between open-loop and closed-loop operation of PFC. The second box gives the choice of AC or DC input. The DC-input option is primarily used for PFC-current loop frequency response testing using DC-input voltage. In this case, the PFC circuit is operated as a current controlled DC-DC boost converter. The third box selects between 2-phase, ILPFC and single-phase PFC. PFC power stage parameters are entered from the *Power Stage Parameters* section.

From the *Control Loop Design* section, the user can select between tuning the current loop or the voltage loop, specify the controller number being designed (COMPI1, COMPI2, and so forth), select between types of controller such as 2P2Z, 3P3Z, and so forth, specify current and voltage loop sampling frequencies, and use the *Compensation Designer* to design the digital controller.

18. Click the *Compensation Designer* tool.

Figure 9 shows a snapshot of this tool. This snapshot shows the frequency response of the PFC power stage (Plant), the loop (OL), and the digital compensator (Comp). The plot indicates the bandwidth, the gain margin, and the phase margin of the loop.

The tool also lets the user select between the *Modeled* plant and the measured plant based on SFRA Data and then change the compensation parameters in frequency domain (gain, pole frequency, and zero frequency). When the compensation design is complete, this tool generates the digital controller coefficients (Comp Coeff) that are saved in the *ILPFC_Base-Settings.h* file with the appropriate controller number (Comp Number). With these generated digital controller coefficients, the user can compile, load, and run the code. The user can then apply power to run the PFC at rated voltage and power levels. The user can then run the SFRA tool. The SFRA tool usage is detailed in Section 5.3.1. The SFRA run generates the measured power stage response and saves that data as the SFRA_Data file inside the ILPFC project. Use this measured plant frequency response data to retune the loop response using the compensation designer GUI.



C2000 Compensation Designer GUI: PFC Current Loop

Comp Number and Style: 1, 2P2Z

Plant Option: Modelled

PID: Kp 1, Ti 1E-06, Td 0

Pole Zero Format: K_{DC} 81.0222 dB

f_{z0} 18.786, f_{z1} 6.208, f_{z2} 1000

f_{p0} 0, f_{p1} 25.214, f_{p2} 200000(

Enter Pole and Zero Location in kHz

Q and K_{DC} in dB

Comp Coeff

B0= 0.4104342
B1= -0.3822446
B2= 0.0712511
B3= 0.0000000
A1= 1.1159960
A2= -0.1159960
A3= 0.0000000

Show Graphs for : ☑ Plant ☑ OL ☑ Comp (Note X Axis is Frequency in Hz)

Stable Loop Bandwidth 6.33 kHz Gain Margin 14.89 dB Phase Margin 35.73 Degrees

Figure 9. Compensation Designer Tool

4 Incremental Builds

The complete CCS project for ILPFC is divided into three incremental builds. This approach provides a step-by-step method to become familiar with the software and understand how it interacts with the IL PFC hardware. This approach also simplifies the task of debugging and testing the boards.

Table 2 shows the build options.

1. To select a particular build option, click the *main.cfg* file.

> **NOTE:** This file shows the PFC power stage schematic, the ADC and PWM interface, *Project Options*, *Control Loop Design* options, and *Power Stage Parameters* entry fields.
>
> Under *Project Options*, there are three pulldown menus.

2. Use the first pulldown menu to select open or closed loop.

> **NOTE:** The second pull-down menu gives options to select AC Input or DC Input only during current loop controller design with an open-voltage loop. Under open-loop and fully closed loop condition (closed voltage and current loops) this second pulldown menu (for AC or DC input selection) remains inactive and the user must use only the AC input.

3. Save the *main.cfg* file when these two pulldown menu selections are complete.

This save automatically sets the INCR_BUILD parameter to the corresponding build selection as shown in Table 2. This parameter is in the IPFC_Base-Settings.h file. When the build option is selected and saved, compile the complete project by selecting rebuild-all compiler option. Table 2 provides more details to run each build option.

Table 2. Incremental Build Options for C2000 ILPFC

Incremental Build Options for PFC	
INCR_BUILD = 1	Open- loop operation, test PWM, and ADC interface, AC input
INCR_BUILD = 2	Open-voltage and closed-current loop control, AC or DC input
INCR_BUILD = 3	Closed-voltage loop and closed-current loop control, AC input

4.1 Procedure to Run the Incremental Builds

Software files related to this C2x-controlled ILPFC system (for example, the main source files, ISR assembly files, and the project file for C framework) are in the directory …\controlSUITE\development_kits\ILPFC_v1.1\IL_PFC_F28035_powerSUITE.

The projects included with this software are targeted for CCS6.

> **CAUTION**
>
> The board has high voltages. The board must only be handled only in a lab environment and only by experienced power supply professionals. To safely evaluate this board, an isolated AC source must be used to power up the unit. Only a voltmeter and an appropriate resistive load must be attached to the output before AC power is applied to the board. This resistive load discharges the bus capacitor quickly when the AC power is turned off. The board has not been tested with electronic load and must not be used with one. There is no output overcurrent protection on the board. Take appropriate measures to prevent any output short-circuit condition. The ILPFC board must always be started with 110 Vac (60 Hz). When the board is running, the input voltage can be changed to any other voltage within the specification.

Do as follows to build and run the example in the PFC software.

4.2 Procedure for Running the Incremental Builds

4.2.1 Build 1: Open-Loop Operation

Objective

The objectives of this build are as follows:

- Evaluate IL PFC PWM and ADC software driver modules.
- Verify the MOSFET gate driver circuit, voltage, and current-sensing circuit.
- Become familiar with the operation of CCS.

Under this build, the system runs in open-loop mode and the measured ADC values are used for circuit verification and instrumentation purposes. The following steps are for building and running a project.

Overview

The software in Build 1 has been configured so the user can quickly evaluate the PWM driver module (software driver) by viewing the related waveforms on a scope and observing the effect of duty-cycle change on PFC output voltage. The user can adjust the PWM duty cycle from CCS watch window. The user can also evaluate the ADC driver module (software driver) by viewing the ADC sampled data in the CCS watch window. The PWM and ADC driver macro instantiations are executed inside the ILPFC-DPL-ISR.asm file that contains the _DPL_Func C-callable assembly function. Figure 10 shows the software blocks used inside _DPL_Func for this build. The two PWM signals for the two PFC switches are obtained from ePWM module 1. ePWM1A drives one of the PFC switches while ePWM1B drives the other. The signals that are sensed and input to the MCU include the following:

- Line and neutral voltages (VL_fb, VN_fb)
- PFC-input current (Ipfc)
- DC-bus voltage (Vpfc)

These quantities are read using the ADC driver module and are indicated in Figure 10. The ADC driver module converts the 12-bit ADC result to a 32-bit Q24 value. A few lines of code in the ISR implements the detection of input AC line half-cycle (positive and negative half-cycles) and the rectification of the input voltage. This section of code generates the input rectified signal Vrect.

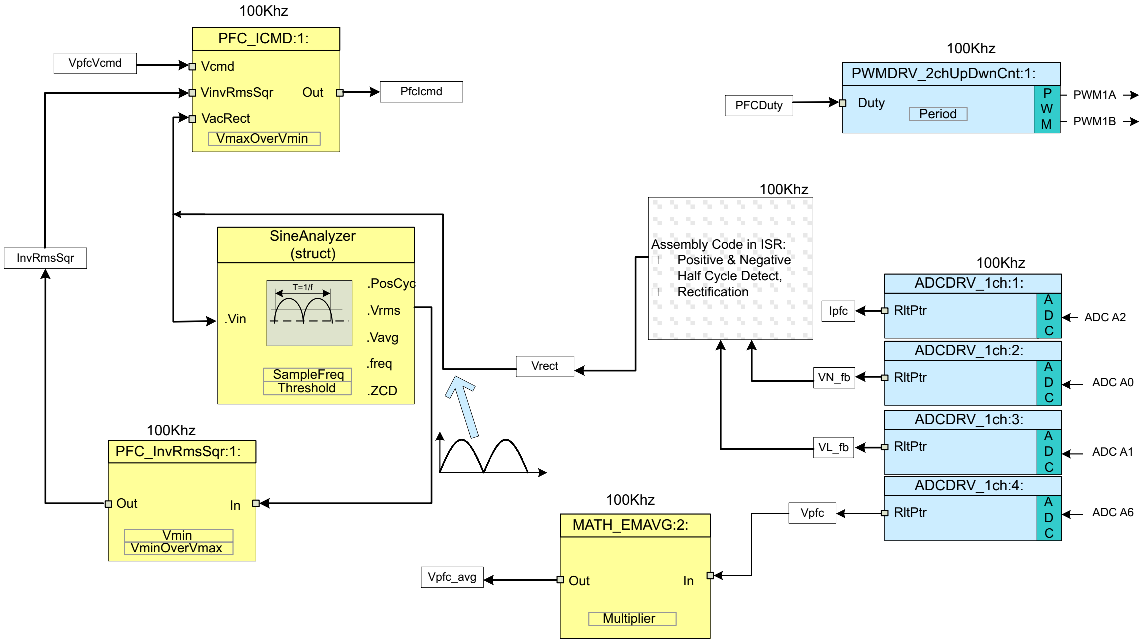


Figure 10. Build 1 Software Blocks

The PWM signals are generated at a frequency of 200 kHz (that is, a period of 5 µs). With the controller operating at 60 MHz, one count of the time-base counter of ePWM1 corresponds to 16.6667 ns. A PWM period of 5 µs is equivalent to 300 counts of the time-base counter (TBCNT1). The ePWM1 module is configured to operate in up-down count mode as shown in Figure 11. A time-base period value of 150 (period-register value) provides a PWM-period value of 300 counts (that is, 5 µs). This PWM-period value is generated based on the input on the main.cfg page.

PFC total inductor current is sampled at the midpoint of the PWM ON pulse because the sampled value represents the average inductor current under CCM (continuous conduction mode) condition. Under DCM condition, the sampled current value also represents an approximate average inductor current because of the oversampling action. PFC inductor current is also oversampled eight times during each 10-µs time period when both the PFC switches turn on when in their respective 5-µs time slot (200-kHz PWM). These 8 sampled values are then used to calculate the average PFC inductor current (see Figure 11).

The voltage signal conversions are also initiated when the PFC switch is on (indicated in Figure 11). The flexibility of ADC and PWM modules on C2000 devices allow for precise and flexible ADC start of conversions. The time-base counters (TBCNTx) of ePWM1 to approximately ePWM4 are used as four time bases to generate the start of conversion (SOC) triggers. Figure 11 shows eight SOC triggers are generated when the following occurs:

- TBCNT1 reaches zero and period.
- TBCNT2 reaches the preset CMPA values during up and down count.
- TBCNT3 reaches zero and preset CMPB value during up count.
- TBCNT4 reaches the preset CMPA and CMPB values during up count.

A dummy ADC conversion is also performed at TBCNT3 zero point to ensure the integrity of the ADC results. Figure 11 shows the PWM outputs generated using the ePWM1 module. PFC current is converted, averaged, and saved as Ipfc for PFC current-loop control.

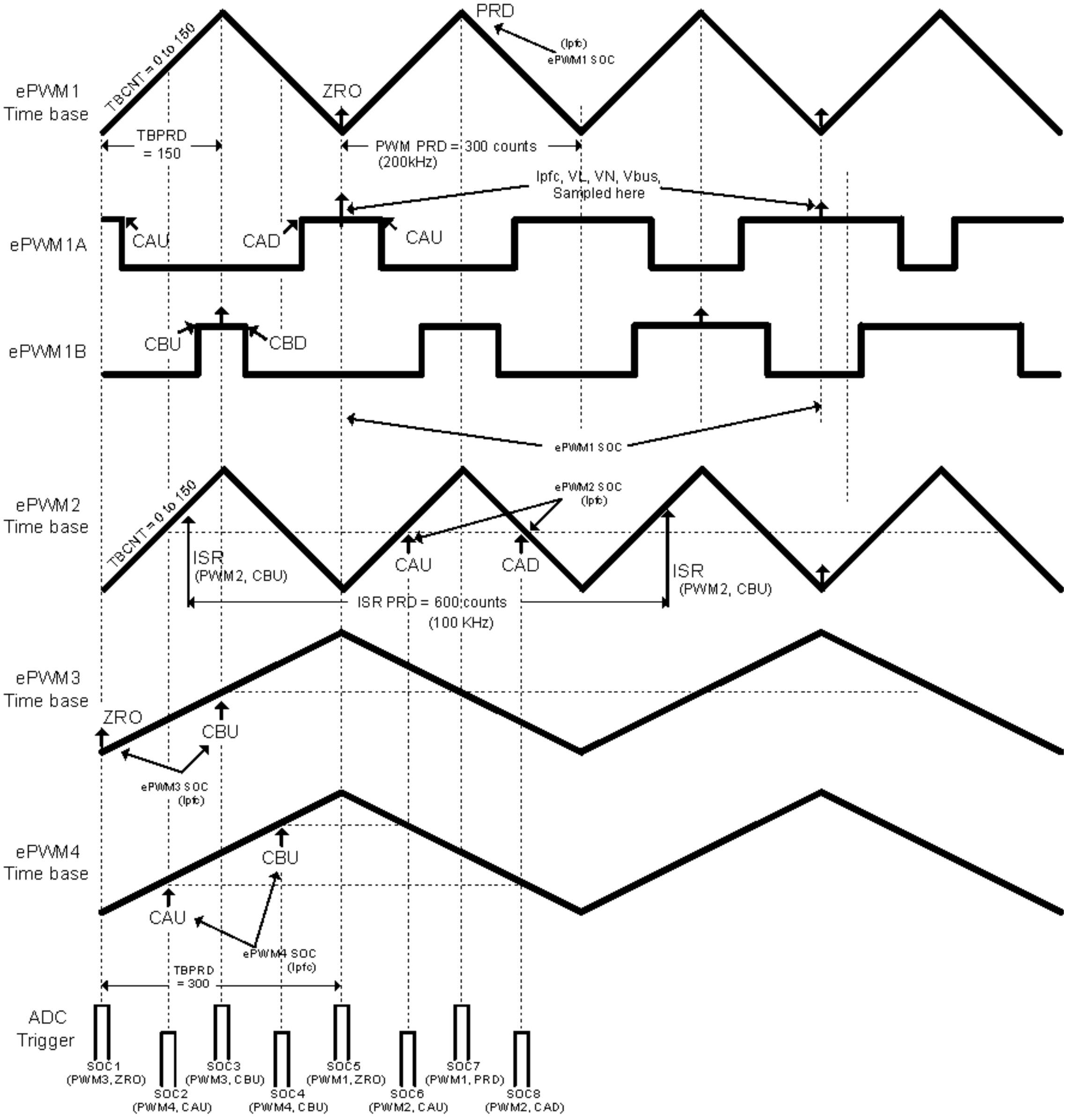


Figure 11. PWM Generation and ADC Sampling

ADC results are read in the fast (100 kHz) ISR labeled ILPFC_ISR_wFRA(). This ISR code are in the ILPFC-main.c file. In this ISR, a C-callable assembly function (_DPL_Func) executes the ADC driver module.

This ISR is triggered by EPWM2 on a CMPB match event on an up count. CMPB is set to 80 so that the ISR is triggered only after the ADC conversions finish. In the ISR function _DPL_Func, the macros PWMDRV_2ch_UpDwnCnt are executed and the PWM compare shadow registers are updated. These macros are loaded into the active register at the next TBCNT1 = ZERO event. The ISR trigger frequency is half that of the PWM switching frequency, as shown in Figure 11. This frequency is the default ISR frequency setting for the TI ILPFC board. For a custom PFC board, this parameter can be changed from the *main.cfg* file using the *Current Loop Frequency* pulldown menu in the *Control Loop Design* section.

Protection

An overvoltage protection mechanism is implemented in software for this ILPFC EVM. The sensed DC-bus output voltage from the ADC input is compared against the overvoltage protection threshold set by the user. The OV-threshold point for this ILPFC EVM has been set to 435 V. This threshold parameter is VBUS_OVP_THRSHLD in the ILPFC_Base-Settings.h file. In case of an OV condition, the PWM outputs are shut off using the trip zone (TZ) registers. The flexibility of the trip mechanism on C2000 devices offers the ability to take actions on trip events. In this project, PWM outputs are driven low in case of a trip event. Both outputs are held in this state until a device reset executes.

4.3 Procedure to Start CCS and Open a Project

Do as follows to execute this build:

NOTE: Do not make any changes to the source files.

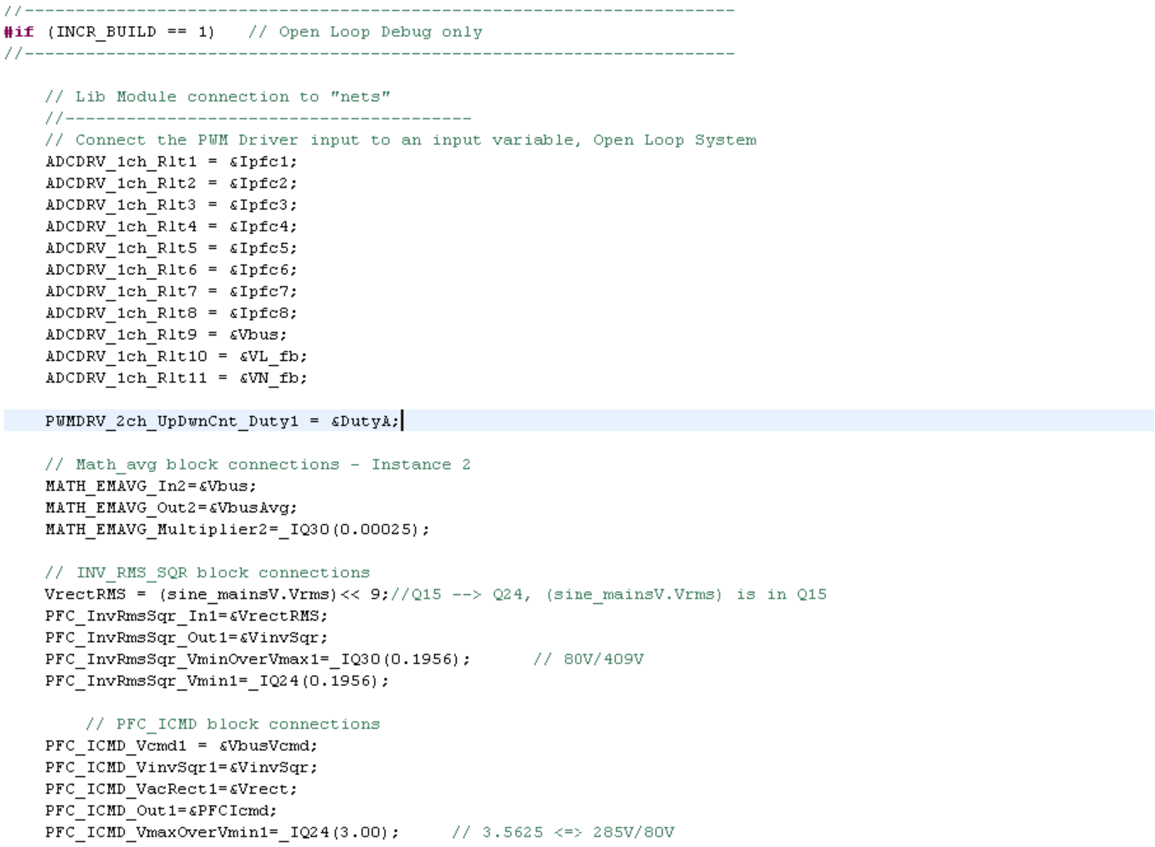
1. Double-click the filename in the project window to open and inspect the ILPFC-DevInit_F2803x.c files.

NOTE: That system clock, peripheral clock prescale, and peripheral clock enables have been set up. The shared GPIO pins have been configured.

2. Open and inspect the ILPFC-Main.c file.

NOTE: Notice the code for different incremental build options, the ISR initialization and the runtime code for the fast 100-kHz C-ISR (ILPFC_ISR_wFRA) and the slower 10-kHz C-ISR (SECONDARY_ISR). The code for the background for(;;) loop is also in this file.

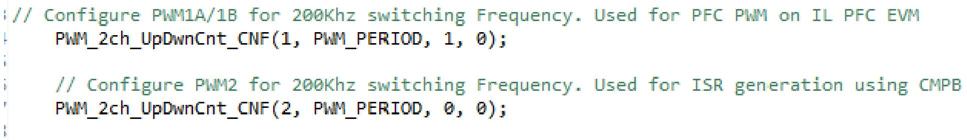
3. Locate and inspect the following code in the ILPFC-Main.c file under initialization code specific for build 1.



```
//----------------------------------------------------------------------
#if (INCR_BUILD == 1)    // Open Loop Debug only
//----------------------------------------------------------------------

    // Lib Module connection to "nets"
    //----------------------------------------
    // Connect the PWM Driver input to an input variable, Open Loop System
    ADCDRV_1ch_Rlt1 = &Ipfc1;
    ADCDRV_1ch_Rlt2 = &Ipfc2;
    ADCDRV_1ch_Rlt3 = &Ipfc3;
    ADCDRV_1ch_Rlt4 = &Ipfc4;
    ADCDRV_1ch_Rlt5 = &Ipfc5;
    ADCDRV_1ch_Rlt6 = &Ipfc6;
    ADCDRV_1ch_Rlt7 = &Ipfc7;
    ADCDRV_1ch_Rlt8 = &Ipfc8;
    ADCDRV_1ch_Rlt9 = &Vbus;
    ADCDRV_1ch_Rlt10 = &VL_fb;
    ADCDRV_1ch_Rlt11 = &VN_fb;

    PWMDRV_2ch_UpDwnCnt_Duty1 = &DutyA;

    // Math_avg block connections - Instance 2
    MATH_EMAVG_In2=&Vbus;
    MATH_EMAVG_Out2=&VbusAvg;
    MATH_EMAVG_Multiplier2=_IQ30(0.00025);

    // INV_RMS_SQR block connections
    VrectRMS = (sine_mainsV.Vrms)<< 9;//Q15 --> Q24, (sine_mainsV.Vrms) is in Q15
    PFC_InvRmsSqr_In1=&VrectRMS;
    PFC_InvRmsSqr_Out1=&VinvSqr;
    PFC_InvRmsSqr_VminOverVmax1=_IQ30(0.1956);      // 80V/409V
    PFC_InvRmsSqr_Vmin1=_IQ24(0.1956);

        // PFC_ICMD block connections
    PFC_ICMD_Vcmd1 = &VbusVcmd;
    PFC_ICMD_VinvSqr1=&VinvSqr;
    PFC_ICMD_VacRect1=&Vrect;
    PFC_ICMD_Out1=&PFCIcmd;
    PFC_ICMD_VmaxOverVmin1=_IQ24(3.00);     // 3.5625 <=> 285V/80V
```

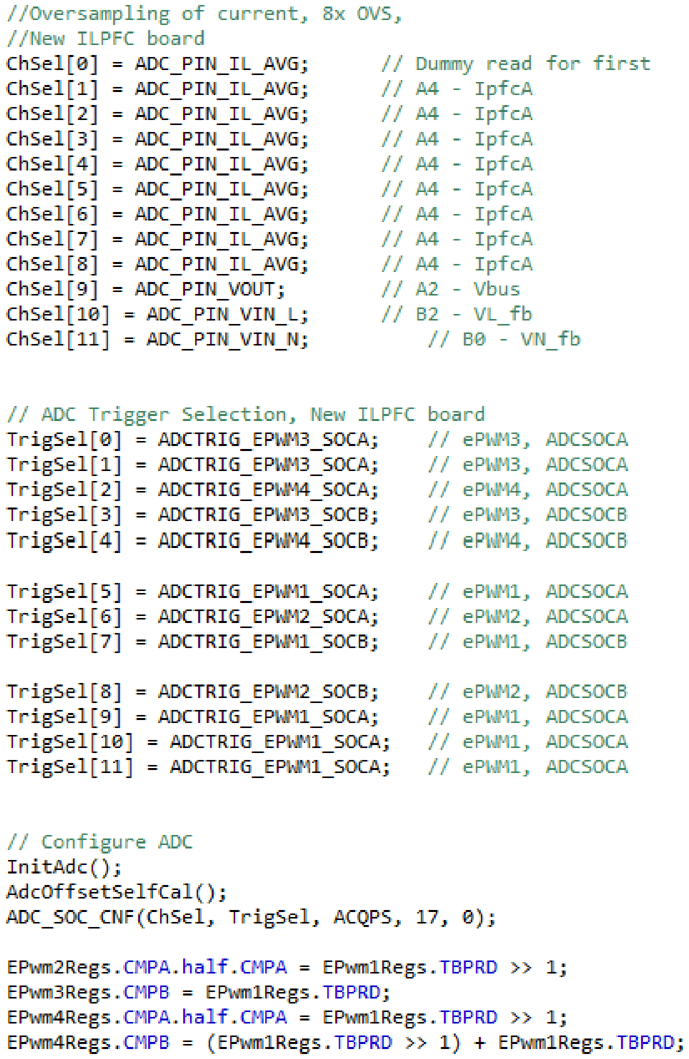
NOTE: The PWMDRV_2ch_UpDwnCnt and ADCDRV_1CH blocks are connected in the control flow in this code.

4. Locate and inspect the following code in the main file under initialization code.



```
// Configure PWM1A/1B for 200Khz switching Frequency. Used for PFC PWM on IL PFC EVM
    PWM_2ch_UpDwnCnt_CNF(1, PWM_PERIOD, 1, 0);

    // Configure PWM2 for 200Khz switching Frequency. Used for ISR generation using CMPB
    PWM_2ch_UpDwnCnt_CNF(2, PWM_PERIOD, 0, 0);
```

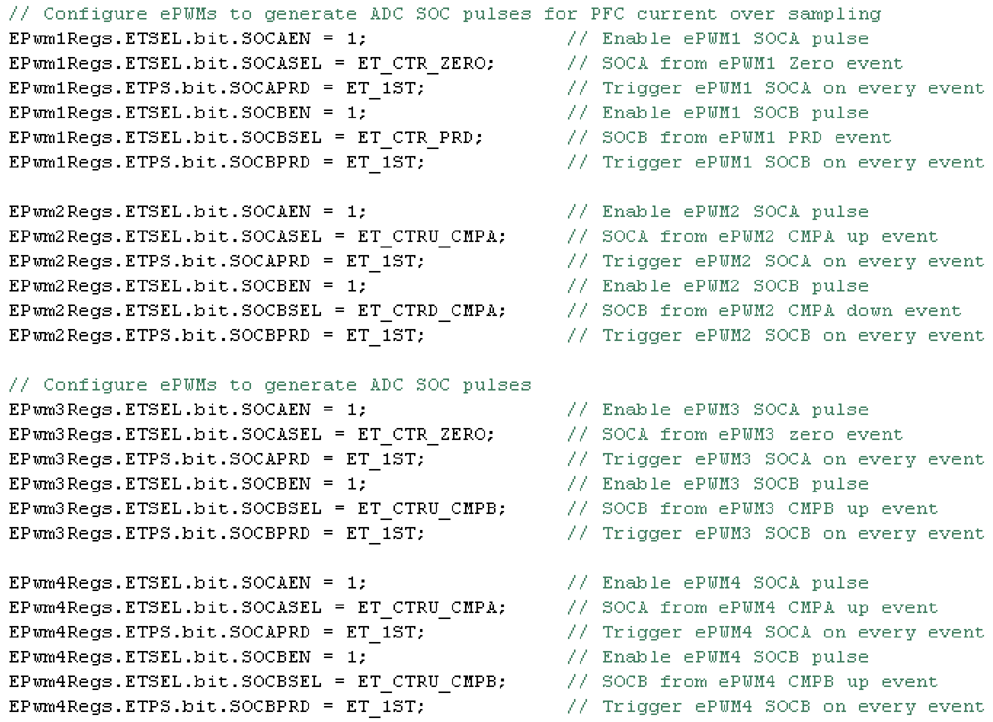
NOTE: The PWMDRV_2ch_UpDwnCnt block is configured and initialized in this code. This configuration and initialization is common for incremental builds. This PWM driver module inputs the total PWM period value of PWM_PERIOD which is generated based on the input on the *main.cfg* page. This PWM_PERIOD parameter is then saved in the ILPFC_Base-Settings.h file. From this value, the PWM driver internally calculates the period register value.



```
//Oversampling of current, 8x OVS,
//New ILPFC board
ChSel[0] = ADC_PIN_IL_AVG;      // Dummy read for first
ChSel[1] = ADC_PIN_IL_AVG;      // A4 - IpfcA
ChSel[2] = ADC_PIN_IL_AVG;      // A4 - IpfcA
ChSel[3] = ADC_PIN_IL_AVG;      // A4 - IpfcA
ChSel[4] = ADC_PIN_IL_AVG;      // A4 - IpfcA
ChSel[5] = ADC_PIN_IL_AVG;      // A4 - IpfcA
ChSel[6] = ADC_PIN_IL_AVG;      // A4 - IpfcA
ChSel[7] = ADC_PIN_IL_AVG;      // A4 - IpfcA
ChSel[8] = ADC_PIN_IL_AVG;      // A4 - IpfcA
ChSel[9] = ADC_PIN_VOUT;        // A2 - Vbus
ChSel[10] = ADC_PIN_VIN_L;      // B2 - VL_fb
ChSel[11] = ADC_PIN_VIN_N;          // B0 - VN_fb


// ADC Trigger Selection, New ILPFC board
TrigSel[0] = ADCTRIG_EPWM3_SOCA;    // ePWM3, ADCSOCA
TrigSel[1] = ADCTRIG_EPWM3_SOCA;    // ePWM3, ADCSOCA
TrigSel[2] = ADCTRIG_EPWM4_SOCA;    // ePWM4, ADCSOCA
TrigSel[3] = ADCTRIG_EPWM3_SOCB;    // ePWM3, ADCSOCB
TrigSel[4] = ADCTRIG_EPWM4_SOCB;    // ePWM4, ADCSOCB

TrigSel[5] = ADCTRIG_EPWM1_SOCA;    // ePWM1, ADCSOCA
TrigSel[6] = ADCTRIG_EPWM2_SOCA;    // ePWM2, ADCSOCA
TrigSel[7] = ADCTRIG_EPWM1_SOCB;    // ePWM1, ADCSOCB

TrigSel[8] = ADCTRIG_EPWM2_SOCB;    // ePWM2, ADCSOCB
TrigSel[9] = ADCTRIG_EPWM1_SOCA;    // ePWM1, ADCSOCA
TrigSel[10] = ADCTRIG_EPWM1_SOCA;   // ePWM1, ADCSOCA
TrigSel[11] = ADCTRIG_EPWM1_SOCA;   // ePWM1, ADCSOCA


// Configure ADC
InitAdc();
AdcOffsetSelfCal();
ADC_SOC_CNF(ChSel, TrigSel, ACQPS, 17, 0);

EPwm2Regs.CMPA.half.CMPA = EPwm1Regs.TBPRD >> 1;
EPwm3Regs.CMPB = EPwm1Regs.TBPRD;
EPwm4Regs.CMPA.half.CMPA = EPwm1Regs.TBPRD >> 1;
EPwm4Regs.CMPB = (EPwm1Regs.TBPRD >> 1) + EPwm1Regs.TBPRD;
```

5. Find and inspect the following code in the main file under the initialization code.

NOTE: This code is where the ADCDRV_1CH block is configured and initialized. This configuration and initialization is also common for all incremental builds.



```
// Configure ePWMs to generate ADC SOC pulses for PFC current over sampling
EPwm1Regs.ETSEL.bit.SOCAEN = 1;                 // Enable ePWM1 SOCA pulse
EPwm1Regs.ETSEL.bit.SOCASEL = ET_CTR_ZERO;      // SOCA from ePWM1 Zero event
EPwm1Regs.ETPS.bit.SOCAPRD = ET_1ST;            // Trigger ePWM1 SOCA on every event
EPwm1Regs.ETSEL.bit.SOCBEN = 1;                 // Enable ePWM1 SOCB pulse
EPwm1Regs.ETSEL.bit.SOCBSEL = ET_CTR_PRD;       // SOCB from ePWM1 PRD event
EPwm1Regs.ETPS.bit.SOCBPRD = ET_1ST;            // Trigger ePWM1 SOCB on every event

EPwm2Regs.ETSEL.bit.SOCAEN = 1;                 // Enable ePWM2 SOCA pulse
EPwm2Regs.ETSEL.bit.SOCASEL = ET_CTRU_CMPA;     // SOCA from ePWM2 CMPA up event
EPwm2Regs.ETPS.bit.SOCAPRD = ET_1ST;            // Trigger ePWM2 SOCA on every event
EPwm2Regs.ETSEL.bit.SOCBEN = 1;                 // Enable ePWM2 SOCB pulse
EPwm2Regs.ETSEL.bit.SOCBSEL = ET_CTRD_CMPA;     // SOCB from ePWM2 CMPA down event
EPwm2Regs.ETPS.bit.SOCBPRD = ET_1ST;            // Trigger ePWM2 SOCB on every event

// Configure ePWMs to generate ADC SOC pulses
EPwm3Regs.ETSEL.bit.SOCAEN = 1;                 // Enable ePWM3 SOCA pulse
EPwm3Regs.ETSEL.bit.SOCASEL = ET_CTR_ZERO;      // SOCA from ePWM3 zero event
EPwm3Regs.ETPS.bit.SOCAPRD = ET_1ST;            // Trigger ePWM3 SOCA on every event
EPwm3Regs.ETSEL.bit.SOCBEN = 1;                 // Enable ePWM3 SOCB pulse
EPwm3Regs.ETSEL.bit.SOCBSEL = ET_CTRU_CMPB;     // SOCB from ePWM3 CMPB up event
EPwm3Regs.ETPS.bit.SOCBPRD = ET_1ST;            // Trigger ePWM3 SOCB on every event

EPwm4Regs.ETSEL.bit.SOCAEN = 1;                 // Enable ePWM4 SOCA pulse
EPwm4Regs.ETSEL.bit.SOCASEL = ET_CTRU_CMPA;     // SOCA from ePWM4 CMPA up event
EPwm4Regs.ETPS.bit.SOCAPRD = ET_1ST;            // Trigger ePWM4 SOCA on every event
EPwm4Regs.ETSEL.bit.SOCBEN = 1;                 // Enable ePWM4 SOCB pulse
EPwm4Regs.ETSEL.bit.SOCBSEL = ET_CTRU_CMPB;     // SOCB from ePWM4 CMPB up event
EPwm4Regs.ETPS.bit.SOCBPRD = ET_1ST;            // Trigger ePWM4 SOCB on every event
```

6. Open and inspect the ILPFC-DPL-ISR.asm file. Notice the _DPL_Init and _DPL_Func sections under Build 1. This is where the PWM and ADC driver macro instantiation is done for initialization and runtime, respectively.

4.4 Procedure to Build and Load the Project for Build 1

To build and load the project for Build 1, do as follows:

1. From the *main.cfg* file, select Project Options as *Open Loop.*
2. In the CCS *Project Explorer* window, click *ILPFC [Active-F2803x_Flash]*.
3. Right-click and select *Rebuild Project*.
4. Click 🐞 ▾ from the CCS menu bar.

NOTE: The program builds and loads into the flash and takes you to the start of Main().

5 Debug Environment Windows

A standard debug practice is to watch local and global variables while debugging code. There are various methods for viewing these variables in CCS, such as memory views and watch views. If a watch view does not open when the debug environment launches, open a new watch view and add various parameters.

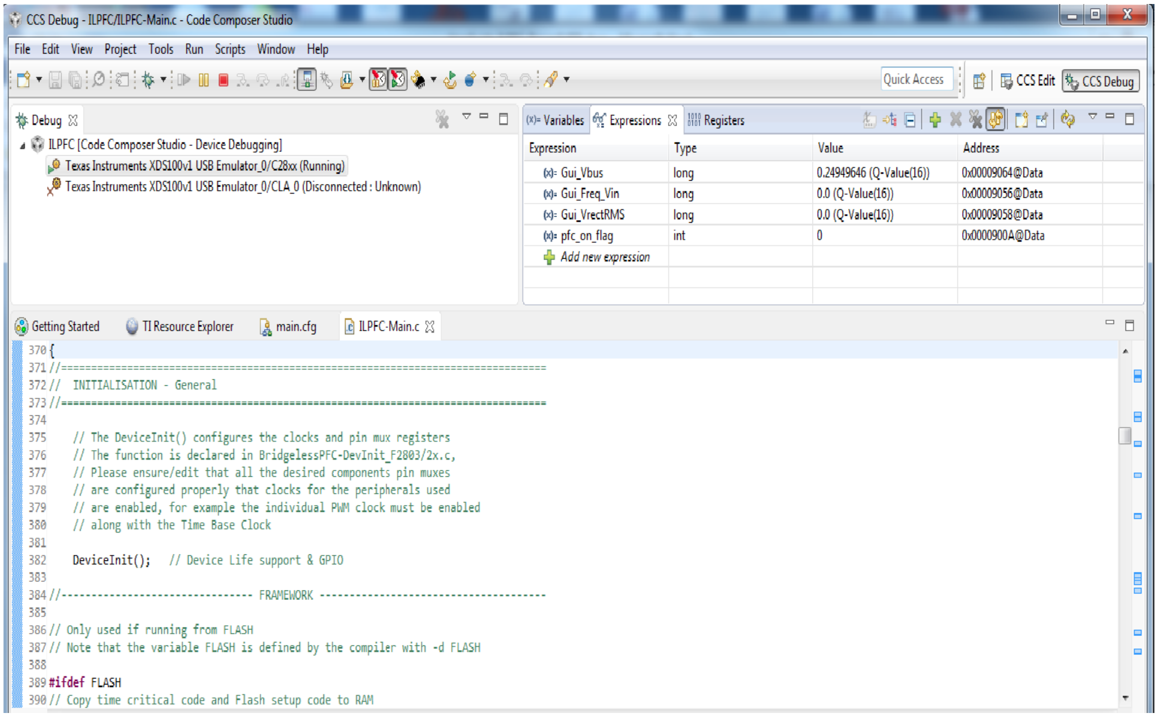
To add a new watch view and add those parameters, do as follows:

1. Click *View* → *Expressions* on the menu bar.
2. Click the *Expressions* tab at the top of the watch view.

   To add expressions to the watch view, do as follows:
3. Type the symbol names of the expressions in the empty boxes in the *Expression* column and press *Enter*.
4. Modify the *Value* as needed.

   To view the watch expressions related to this PFC project, do as follows:
5. Click *View* → *Scripting Console*.
6. From the *Scripting Console* tab, click 📂.
7. Browse to the ILPFC project folder.
8. Select the ILPFC project folder.
9. For the files names, select *All Files(*).*
10. Find the WatchWinVars.js file.
11. Open this file.
12. Enable *Real-time Mode*.
13. Select *Continuous Refresh* for the watch window expressions.

14. Run the program (see Figure 12 for an example of the watch view).



ILPFC [Code Composer Studio - Device Debugging]
Texas Instruments XDS100v1 USB Emulator_0/C28xx (Running)
Texas Instruments XDS100v1 USB Emulator_0/CLA_0 (Disconnected : Unknown)

Expression	Type	Value	Address
Gui_Vbus	long	0.24949646 (Q-Value(16))	0x00009064@Data
Gui_Freq_Vin	long	0.0 (Q-Value(16))	0x00009056@Data
Gui_VrectRMS	long	0.0 (Q-Value(16))	0x00009058@Data
pfc_on_flag	int	0	0x0000900A@Data
Add new expression			

```
{
//=============================================================================
//  INITIALISATION - General
//=============================================================================

    // The DeviceInit() configures the clocks and pin mux registers
    // The function is declared in BridgelessPFC-DevInit_F2803/2x.c,
    // Please ensure/edit that all the desired components pin muxes
    // are configured properly that clocks for the peripherals used
    // are enabled, for example the individual PWM clock must be enabled
    // along with the Time Base Clock

    DeviceInit();   // Device Life support & GPIO

//------------------------------ FRAMEWORK -------------------------------------

// Only used if running from FLASH
// Note that the variable FLASH is defined by the compiler with -d FLASH

#ifdef FLASH
// Copy time critical code and Flash setup code to RAM
```

Figure 12. The CCS Watch View for Build 1

5.1 Run the Code

To run the code, do as follows:

1. In the watch view, add *DutyA* as a variable.
2. Set this variable to 0.1 (1677721 in Q24).

NOTE: This variable sets the duty cycle for the PFC converter.

3. Apply an appropriate resistive load to the PFC system at the DC-output terminals (this resistor rating can be 8 kΩ/20 W, 4 kΩ/40 W, or 2 kΩ/80 W).
4. Slowly apply AC power to the board.
5. Measure and verify the DC-bus voltage corresponding to the applied input voltage and the PWM duty ratio.

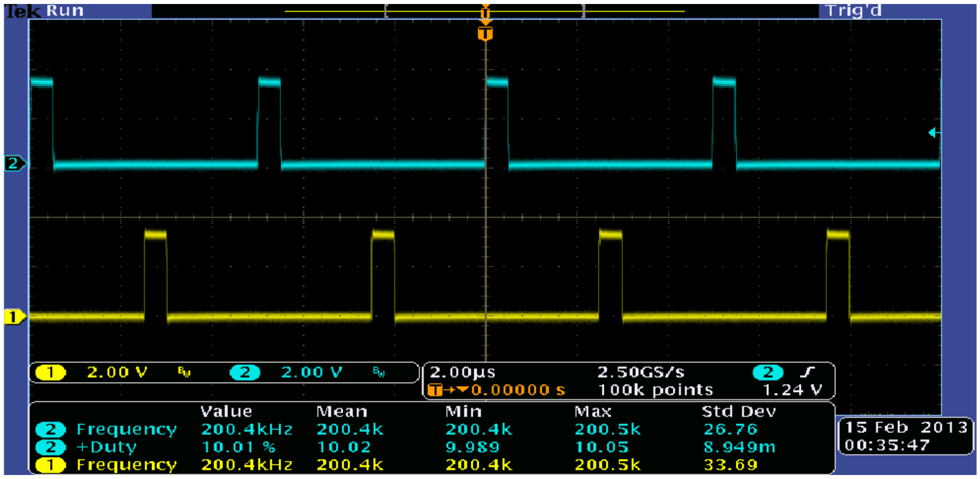
6. Use *DutyA* to slowly change the duty from the watch window.

NOTE: The boost converter output voltage changes accordingly. The CCS watch window shows the correct DC-bus voltage, input AC-line frequency, and AC rms voltage. Notice the Q-format used for each variable.

Figure 13 shows two PWM outputs (Ch1 and Ch2) with duty ratio set to 10%. The PWM frequency is measured to be 200 kHz.

CAUTION

Do not let the output voltage exceed the maximum-voltage rating of the board (400 V).



	Value	Mean	Min	Max	Std Dev
2 Frequency	200.4kHz	200.4k	200.4k	200.5k	26.76
2 +Duty	10.01 %	10.02	9.989	10.05	8.949m
1 Frequency	200.4kHz	200.4k	200.4k	200.5k	33.69

Figure 13. Oscilloscope With Two PWM Outputs With a Duty Ratio Set to 10%

7. Try different duty cycle values.
8. Observe the corresponding ADC results.
9. Increase duty cycle value in small steps.

WARNING

Always observe the output voltage carefully. Do not let the output voltage exceed the maximum voltage rating of the board. Use an oscilloscope to probe and verify the waveforms, like the PWM gate drive signals, input voltage, input current, and output voltage. Take appropriate safety measures while probing these high-voltage signals.

To halt the MCU when in real-time mode, do as follows:

1. Ensure the AC input is off.
2. Wait until the DC-bus capacitor is discharged completely.
3. Press *Halt* on the toolbar to halt the processor.
4. Take the MCU out of *Real-time Mode*.
5. Reset the MCU.

End of Exercise

5.2 Build 2: ILPFC With Closed-Current Loop

Objective

The objective of this build is to verify the operation of the ILPFC in closed-current loop mode.

Overview

Figure 14 shows the software blocks used in this build. One additional software block is added to the build 1 diagram (Figure 10) to implement this closed-current loop system. The SineAnalyzer block calculates the RMS voltage and frequency of the input voltage. PFC InvRmsSqr block calculates the inverse of the square of the RMS input voltage. This calculated value with the rectified voltage (Vrect) is used in the software block PFC_ICMD to generate the reference current command PfcIcmd for the PFC current control loop. The PFC_ICMD block uses a third input VbusVcmd to control for the magnitude of the reference current command. Because this software build implements only the PFC current loop (open-voltage loop), this parameter VbusVcmd must be varied from the CCS watch window to adjust the magnitude of the reference current and the PFC bus voltage. A 2P2Z controller implements the current-control loop. Figure 14 shows this controller as the fourth software block in as CNTL_2P2Z:1. Depending on the control-loop requirements, other control blocks such as a PI or a 3P3Z controller can also be used.

Figure 14 shows that the current loop control block is executed at a 100-kHz rate. CNTL_2P2Z is a second-order compensator realized from an IIR filter structure. This function is independent of any peripherals and does not require a CNF function call.

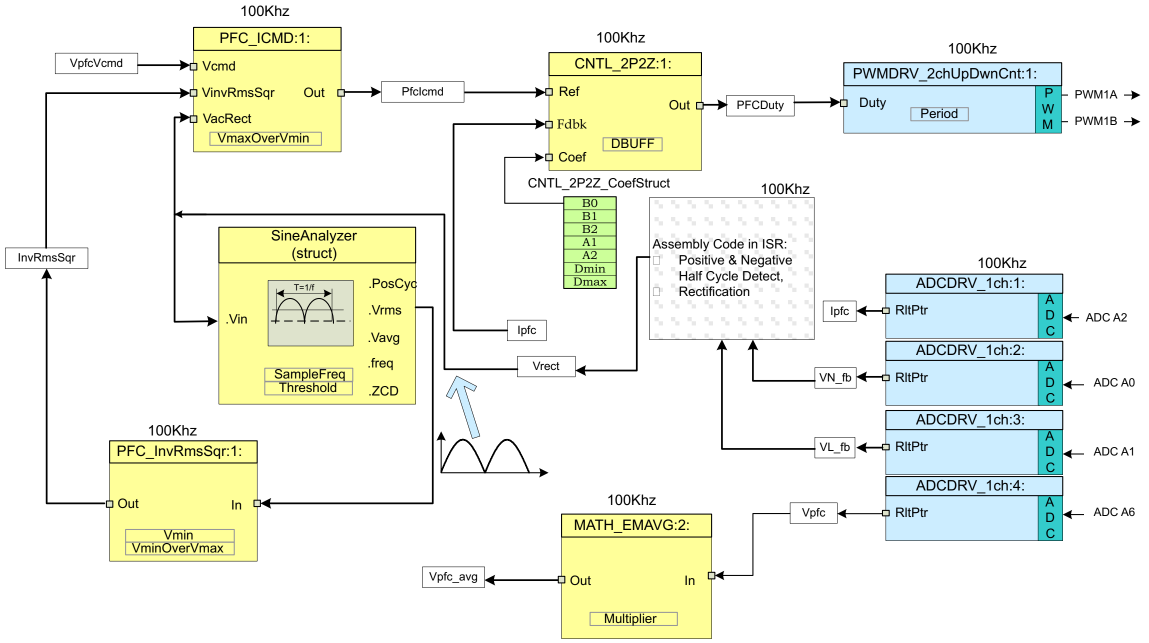


Figure 14. Build 2 Software Blocks

This 2P2Z controller requires five control coefficients. These coefficients and the clamped output of the controller are stored as the elements of a structure named CNTL_2P2Z_CoefStruct1. The CNTL_2P2Z block can be instantiated multiple times if the system requires multiple loops. Each instance can have a separate set of coefficients. The CNTL_2P2Z instance for the current loop uses the coefficients stored as the elements of structure CNTL_2P2Z_CoefStruct1. A second instantiation of CNTL_2P2Z with a different structure (CNTL_2P2Z_CoefStruct2) can be used for PFC voltage loop control as in Section 5.3.

You can use the Compensation Designer GUI to change the controller coefficients. This GUI modifies the values for B0, B1, B2, A1, and A2 in the CNTL_2P2Z_CoefStruct1 structure. The 2P2Z controller can be expressed in PID form. Using the Compensation Designer GUI, you can change the coefficients by changing the PID coefficients The equations relating the five controller coefficients to the three PID gains follow. For the current loop, these P, I, and D coefficients are named Pgain_I, Igain_I, and Dgain_I, respectively. For the voltage loop in Build 3, these coefficients are named Pgain_V, Igain_V, and Dgain_V, respectively. These coefficients are used in the Q26 format.

The compensator block (CNTL_2P2Z) has a reference input and a feedback input. The feedback input labeled Fdbk comes from the ADC. The reference input labeled Ref comes from PFC_ICMD block. The z-domain transfer function for CNTL_2P2Z is given as Equation 1:

$$\frac{U(z)}{E(z)} = \frac{b_0 + b_{1z}^{-1} + b_{2z}^{-2}}{1 + a_{1z}^{-1} + a_{2z}^{-2}} \tag{1}$$

The recursive form of the PID controller is given by Equation 2:

$$u(k) = u(k-1) + b_{0e}(k) + b_{1e}(k-1) + b_{2e}(k-2) \tag{2}$$

where Equation 3:

$$\begin{aligned} b_0 &= K_p + K_i + K_d \\ b_1 &= -K_p + K_i - 2K_d \\ b_2 &= K_d \end{aligned} \tag{3}$$

The z-domain transfer function of this PID is as Equation 4:

$$\frac{U(z)}{E(z)} = \frac{b_0 + b_{1z}^{-1} + b_{1z}^{-2}}{1 - z^{-1}} \tag{4}$$

Comparing Equation 4 with the general form, PID is a special case of CNTL_2P2Z control where Equation 5:

$$a_1 = -1 \text{ and } a_2 = 0 \tag{5}$$

The MATH_EMAVG (exponential moving average) block calculates the average of the output DC-bus voltage. The output from this block is used to detect an overvoltage condition followed by a PWM shutdown.

5.2.1 Procedure to Build and Load the Project

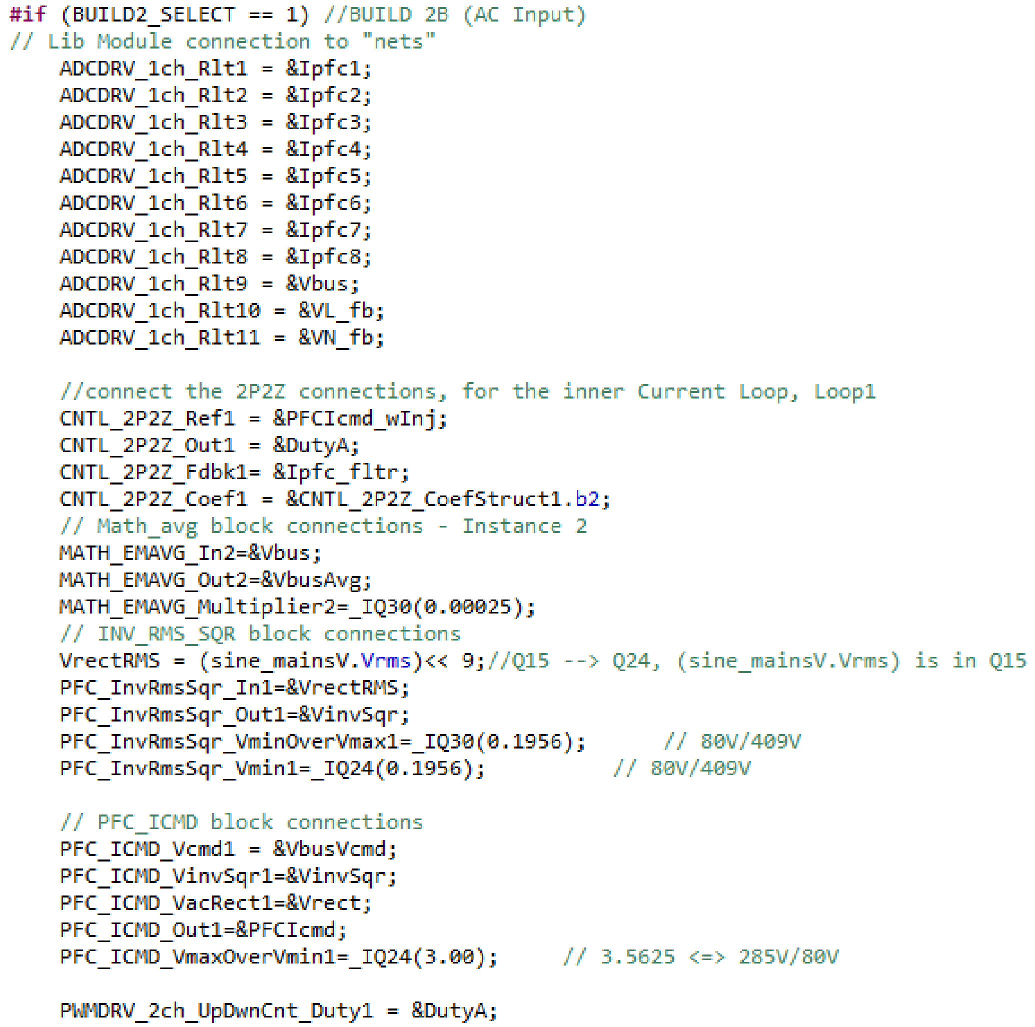
To build and load the project, do as follows:

1. From the *main.cfg* file, select *Closed-Current Loop* and *AC Input* as *Project Options*.
2. Save the file.
3. Open the ILPFC_Base-Settings.h file.
4. Ensure INCR_BUILD is set to 2.
5. Ensure BUILD2_SELECT set to 1.

NOTE: BUILD2_SELECT runs the ILPFC code that applies to the AC input only.

6. Find and inspect the following code in the ILPFC-Main.c file under the initialization code for Build 2 with an AC input.

NOTE: This code contains all the software blocks related to Build 2 are connected in the control flow.



```
#if (BUILD2_SELECT == 1) //BUILD 2B (AC Input)
// Lib Module connection to "nets"
    ADCDRV_1ch_Rlt1 = &Ipfc1;
    ADCDRV_1ch_Rlt2 = &Ipfc2;
    ADCDRV_1ch_Rlt3 = &Ipfc3;
    ADCDRV_1ch_Rlt4 = &Ipfc4;
    ADCDRV_1ch_Rlt5 = &Ipfc5;
    ADCDRV_1ch_Rlt6 = &Ipfc6;
    ADCDRV_1ch_Rlt7 = &Ipfc7;
    ADCDRV_1ch_Rlt8 = &Ipfc8;
    ADCDRV_1ch_Rlt9 = &Vbus;
    ADCDRV_1ch_Rlt10 = &VL_fb;
    ADCDRV_1ch_Rlt11 = &VN_fb;

    //connect the 2P2Z connections, for the inner Current Loop, Loop1
    CNTL_2P2Z_Ref1 = &PFCIcmd_wInj;
    CNTL_2P2Z_Out1 = &DutyA;
    CNTL_2P2Z_Fdbk1= &Ipfc_fltr;
    CNTL_2P2Z_Coef1 = &CNTL_2P2Z_CoefStruct1.b2;
    // Math_avg block connections - Instance 2
    MATH_EMAVG_In2=&Vbus;
    MATH_EMAVG_Out2=&VbusAvg;
    MATH_EMAVG_Multiplier2=_IQ30(0.00025);
    // INV_RMS_SQR block connections
    VrectRMS = (sine_mainsV.Vrms)<< 9;//Q15 --> Q24, (sine_mainsV.Vrms) is in Q15
    PFC_InvRmsSqr_In1=&VrectRMS;
    PFC_InvRmsSqr_Out1=&VinvSqr;
    PFC_InvRmsSqr_VminOverVmax1=_IQ30(0.1956);      // 80V/409V
    PFC_InvRmsSqr_Vmin1=_IQ24(0.1956);          // 80V/409V

    // PFC_ICMD block connections
    PFC_ICMD_Vcmd1 = &VbusVcmd;
    PFC_ICMD_VinvSqr1=&VinvSqr;
    PFC_ICMD_VacRect1=&Vrect;
    PFC_ICMD_Out1=&PFCIcmd;
    PFC_ICMD_VmaxOverVmin1=_IQ24(3.00);     // 3.5625 <=> 285V/80V

    PWMDRV_2ch_UpDwnCnt_Duty1 = &DutyA;
```

7. Open and inspect the ILPFC-DPL-ISR.asm file.
8. Notice the _DPL_Init and _DPL_Func sections under Build 2.

NOTE: In this code, the macro instantiations under Build 2 occur for initialization and runtime, respectively.

9. In the CCS *Project Explorer* window, click ILPFC [Active-F2803x_Flash].
10. Right-click and select *Rebuild Project*.
11. Click 🐞▾ from the CCS menu bar.

NOTE: The program builds and loads into the flash and proceeds to the start of Main().

12. Enable *Real-time Mode*.
13. Select *Continuous Refresh* for the watch window expressions.
14. Click *Run* (F8) on the toolbar to run the code.

15. In the watch view, add the VbusVcmd variable.
16. Set the variable to 0.025 (419430 in Q24).

NOTE: This variable sets the magnitude of the reference current command for the current-control loop.

17. Apply an appropriate resistive load to the PFC system at the PFC output terminals.

NOTE: For example, a 8.0-kΩ resistor with a 40-W rating can be used. This resistor provides a load of 18 W at 380-V bus voltage.

18. Slowly apply 82-V rms of AC power to the board from an isolated AC source.
19. Monitor the DC-bus voltage as the input voltage increases.

NOTE: With the 8-kΩ resistive load, the bus voltage should be approximately 380 V.

20. Use a current probe to observe the input current.

NOTE: With 82-Vrms input, an 8.0-kΩ resistive load, and bus voltage set to 380 V, you should see the following waveforms. Ch4 is the input current; Ch1 and Ch2 are the PWM outputs. With the current loop closed, the input current should have the same shape of the input voltage with good power factor.

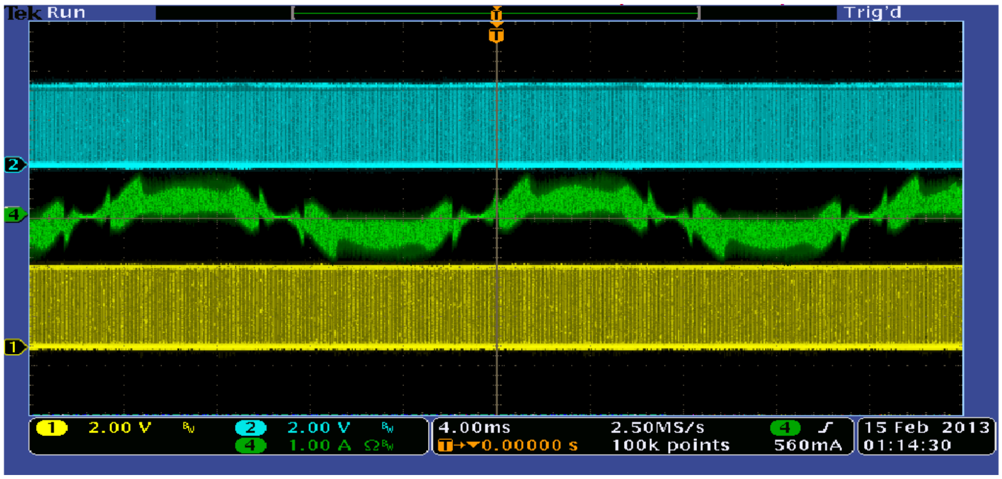


Figure 15. Waveform

21. Increase the VbusVcmd variable slightly (in steps of 0.002).
22. Observe the bus voltage settles to a higher value.

CAUTION

Increasing VbusVcmd also increases the magnitude of the current reference signal and, because the PFC-voltage loop is open, the bus voltage rises.

23. Change the input voltage or increase the load (reduce the load-resistance value) to see the PFC operation under current-control loop.

> **CAUTION**
>
> Reducing the load (increasing the load resistance) may cause higher DC bus voltage.

24. From the *main.cfg* page, ensure these power-stage parameter values: Vin (Vrms) is 120, Vbus (V) is 380, and Pout (W) is 400 to approximately 550.

NOTE: The values in step 24 force the compensation designer use the CCM model for the PFC current loop power stage.

25. Save the main.cfg file.
26. Click the *Compensation Designer* under the *Control Loop Design* section.

NOTE: The *Compensation Designer* GUI shows the current-loop Bode plots with the modelled PFC current-loop power stage (CCM model) and the current loop compensator. For the TI ILPFC board, the default-current loop compensator COMPI1 yields a loop bandwidth (BW) of 6.33 kHz with a phase margin (PM) of 36° and gain margin (GM) of 15 dB. The GUI indicates the style or type of compensator (2P2Z), the gain and the pole-zero frequencies and the corresponding compensation coefficients (An, Bn).

27. Note the default compensation coefficients used in this build (this is the current loop controller COMPI1 selected from the *main.cfg* file under *Control Loop Design* section).
28. Close the *Compensation Designer*.

NOTE: You can run the SFRA, measure the current loop Bode plots, and compare the measured plots with those noted in step 27.

29. Set the PFC resistive load such that it draws at least 400 W (550-W maximum) at 380-V bus with 120-VAC input (CCM operation).
30. Adjust the VbusVcmd variable to set the bus voltage to 380 V.
31. Use the SFRA tool to check the loop-gain plot for the current loop.

> **CAUTION**
>
> In this build, use caution doing this test because the voltage loop is open.

32. To run the SFRA tool, click *main.cfg*.
33. From the *Control Loop Design* section, click SFRA.
34. When the SFRA tool opens, click *Setup Connection*.
35. Select the appropriate COM port.
36. Uncheck *Boot on Connect*.
37. Click *OK*.
38. Click *Connect*.
39. Wait for the SFRA GUI to connect to the ILPFC hardware.

40. Click *Start Sweep*.

NOTE: The SFRA GUI runs, calculates the loop-gain response, and plots it.

41. Compare the BW, PM, and GM data from this GUI to those obtained in step 27.

NOTE: The differences in the plots are due mainly to current loop model parameters (PFC inductor, DCR, and so forth). The SFRA GUI saves the measured current-loop power stage frequency response data this can be used to retune the current-loop controller for optimum performance.

42. Close the SFRA GUI.
43. Open the *Compensation Designer*.
44. Under *Plant Option*, click *Browse SFRA Data*.
45. Select the measured power stage response data that the SFRA GUI saved in the ILPFC project folder from the most recent SFRA run.
46. Change the *Plant Option* to *SFRA Data*.
47. Adjust the compensator gain, poles, and zeros for optimum-loop frequency response.
48. Ensure that the loop-stability margins are maintained while designing this new controller.

NOTE: The GUI shows the *Comp Coeff* for the current loop. These coefficients can be used to replace the default compensation coefficients COMPI1 noted in step 27.

49. Click *Save COMP* from the compensation designer GUI.

NOTE: This action copies the new coefficients to the appropriate compensation settings (COMPI1) inside ILPFC_Base-Settings.h.

50. Turn off AC power.
51. Reset the MCU.
52. Recompile and reload the ILPFC project with the new coefficients.
53. Click *Run* on the toolbar.

NOTE: The Bode plots from the compensation designer and the SFRA should match more accurately.

54. Turn off AC power.
55. Reset the MCU.

End of Exercise

5.3 Build 3: ILPFC With Closed Voltage and Current Loop

Objective

The objective of this build is to verify the operation of the complete ILPFC project from the CCS environment.

Overview

Figure 16 shows the software blocks used in this build. Compared to build 2 in Figure 14, Build 3 uses an additional 2P2Z control block labeled as CNTL_2P2Z:2. This additional 2P2Z is the second instantiation of the 2P2Z control block to implement the ILPFC voltage loop control. For TI ILPFC EVM, this voltage-loop controller is executed at a rate of 50 kHz, which is half the rate for current loop. The output from this control block drives the input node VbusVcmd of the PFC_ICMD block. This output is the main difference compared to Build 2 where VbusVcmd is updated by user from CCS watch window in an open-voltage loop mode.

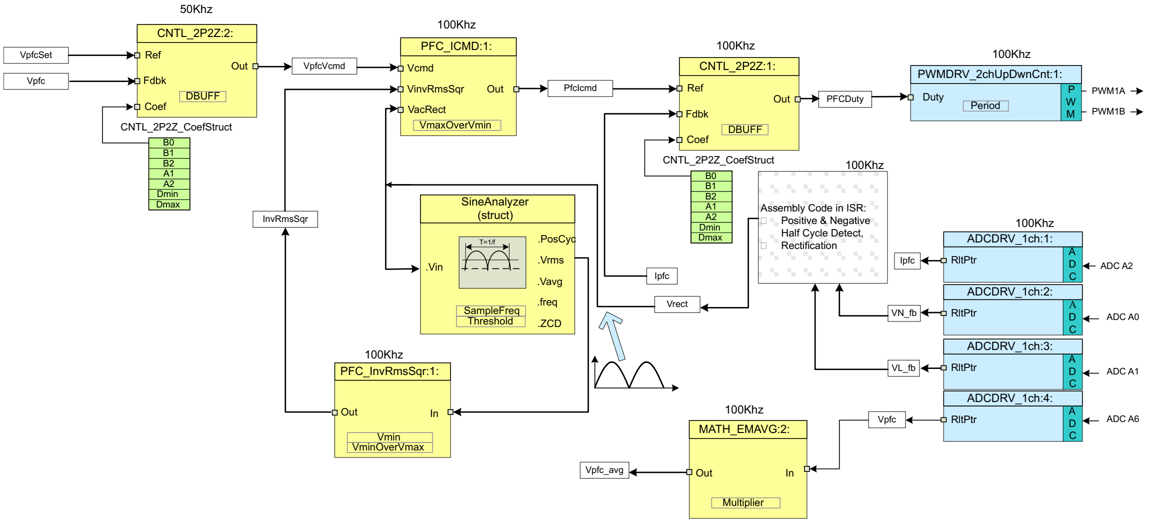


Figure 16. Build 3 Software Blocks

Similar to current-loop controller, this voltage loop controller, CNTL_2P2Z:2, also requires five control coefficients. These coefficients and the clamped output of the controller are stored as the elements of a second structure named CNTL_2P2Z_CoefStruct2. The coefficients for this controller can be changed from the compensation designer GUI. Using this GUI, you can modify the values for B0, B1, B2, A1, and A2 in the structure CNTL_2P2Z_CoefStruct2.

Start-Up, Inrush Current Control, and Slew-Limit

At startup, the controller monitors the PFC DC-bus voltage. When this voltage reaches a minimum level (this minimum level for this ILPFC EVM is set around 160 VDC) PFC action is enabled and the output-DC bus slowly ramps up to the pre-set value of about 390 VDC. This output-voltage level is set by the constant VBUS_RATED_VOLTS defined in the PFC header file ILPFC_Base-Settings.h. The ramp-up speed is set by the parameter VbusSlewRate defined and implemented in the soft-start state machine task C2. This part of the software can be quickly modified to implement any other desired mode for PFC startup.

Input Power Monitor

This ILPFC EVM is equipped with an input power monitoring feature. All the calculation for input power measurement is done inside a 10-kHz ISR running the SineAnalyzer macro (see Figure 16). This macro block takes the rectified-input voltage and the rectified-input current as its inputs. Based on this information, the block calculates the RMS-input power, RMS-input voltage, and RMS-input current using Equation 6 , Equation 7, and Equation 8, respectively:

$$V_{rms} = \sqrt{[\frac{1}{T}\int_{t}^{t+T} v^2(t)dt]} = \frac{1}{N}\sqrt{\sum_{n=1}^{N} V^2(n)}$$

$$N = T / T_s \qquad (6)$$

$$I_{rms} = \sqrt{[\frac{1}{T}\int_{t}^{t+T} i^2(t)dt]} = \frac{1}{N}\sqrt{\sum_{n=1}^{N} I^2(n)} \qquad (7)$$

$$P_{rms} = \frac{1}{T}\int_{t}^{t+T}[v(t)\times i(t)]dt = \frac{1}{N}\sum_{n=1}^{N} V(n)\times I(n) \qquad (8)$$

The time period *T* of the rectified input voltage and current is calculated by setting a low-threshold voltage level for the rectified-voltage signal and then by measuring the time between the consecutive points when the signal crosses this threshold level. Figure 17 illustrates this calculation.

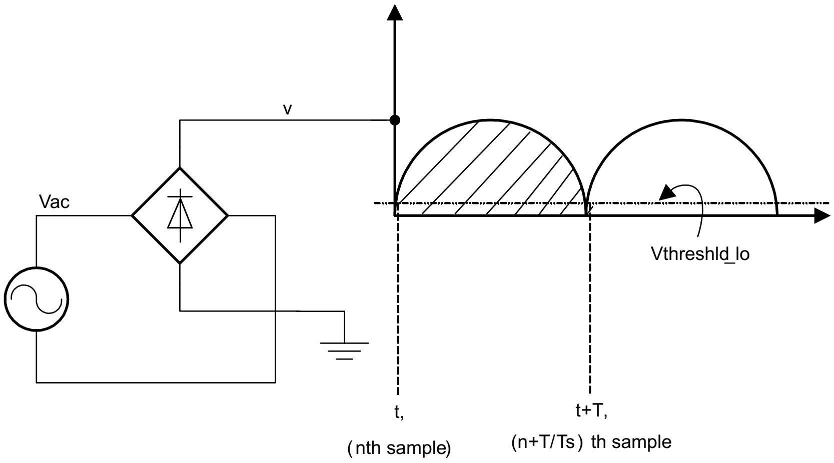


Figure 17. RMS Input Voltage and Current Calculation Scheme

5.3.1 Procedure to Build and Load the Project

To build and load the project, do as follows:

1. From the *main.cfg* file, ensure that the power stage parameters are set as follows: Vin (Vrms) is 120, Vbus (V) is 390, and Pout (W) is 400 is approximately 550.
2. Select *Project Options* as *Closed Current & Voltage Loop*.
3. From the *Control Loop Design* section, select *Voltage Loop* from the pulldown option called *Tuning*.

NOTE: This allows controller design for the PFC DC-bus voltage loop. Under *Comp Number*, notice the default voltage loop controller COMPV1 gets selected.

4. Change this to COMPV2.

NOTE: COMPV2 is the voltage loop controller that runs under steady-state condition. Under transient conditions, a COMPV1 is selected that provides higher-voltage loop bandwidth for faster transient response of PFC DC-bus voltage.

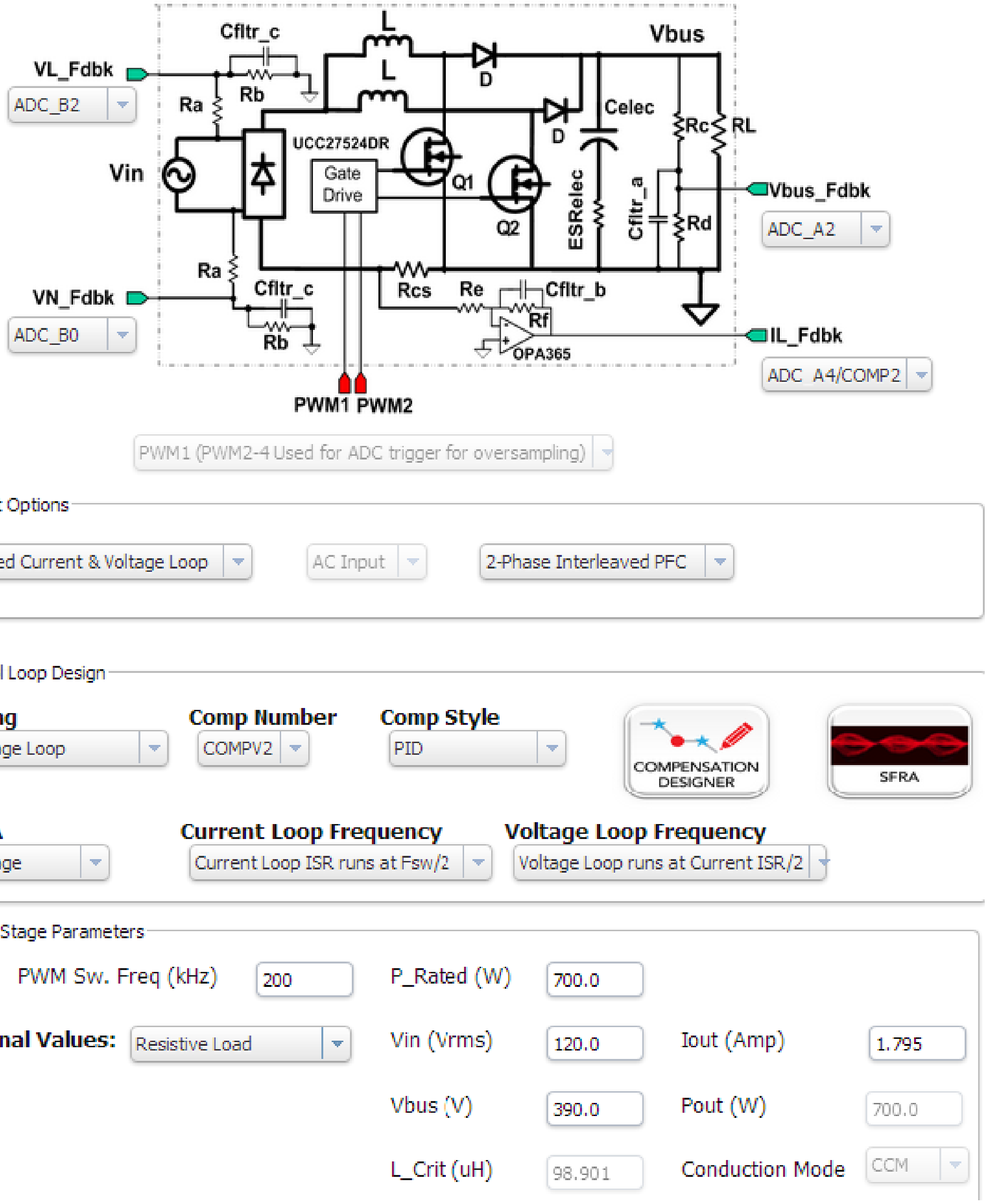
5. Select PID as the *Comp Style*.
6. For the *SFRA* pulldown option, select *Voltage*.

NOTE: When the SFRA GUI is run, it plots the loop response for the PFC-voltage loop.

7. For the *Voltage Loop Frequency* pulldown option, select *Voltage Loop runs at Current ISR/2*.

NOTE: The voltage-loop sampling frequency is half of the current-loop sampling frequency. This is the default setting for TI ILPFC EVM.

8. Save the main.cfg file (Figure 18 shows the powerSUITE GUI.)
9. Open the ILPFC_Base-Settings.h file.
10. Check that the build option parameters are set as follows: INCR_BUILD is set to 3 and VOLTAGE_LOOP_RUN_RATIO is set to 2.



VL_Fdbk: ADC_B2

VN_Fdbk: ADC_B0

Vbus_Fdbk: ADC_A2

IL_Fdbk: ADC_A4/COMP2

PWM1 (PWM2-4 Used for ADC trigger for oversampling)

Project Options

Closed Current & Voltage Loop | AC Input | 2-Phase Interleaved PFC

Control Loop Design

Tuning: Voltage Loop

Comp Number: COMPV2

Comp Style: PID

SFRA: Voltage

Current Loop Frequency: Current Loop ISR runs at Fsw/2

Voltage Loop Frequency: Voltage Loop runs at Current ISR/2

Power Stage Parameters

Fsw: PWM Sw. Freq (kHz): 200

P_Rated (W): 700.0

Nominal Values: Resistive Load

Vin (Vrms): 120.0

Iout (Amp): 1.795

Vbus (V): 390.0

Pout (W): 700.0

L_Crit (uH): 98.901

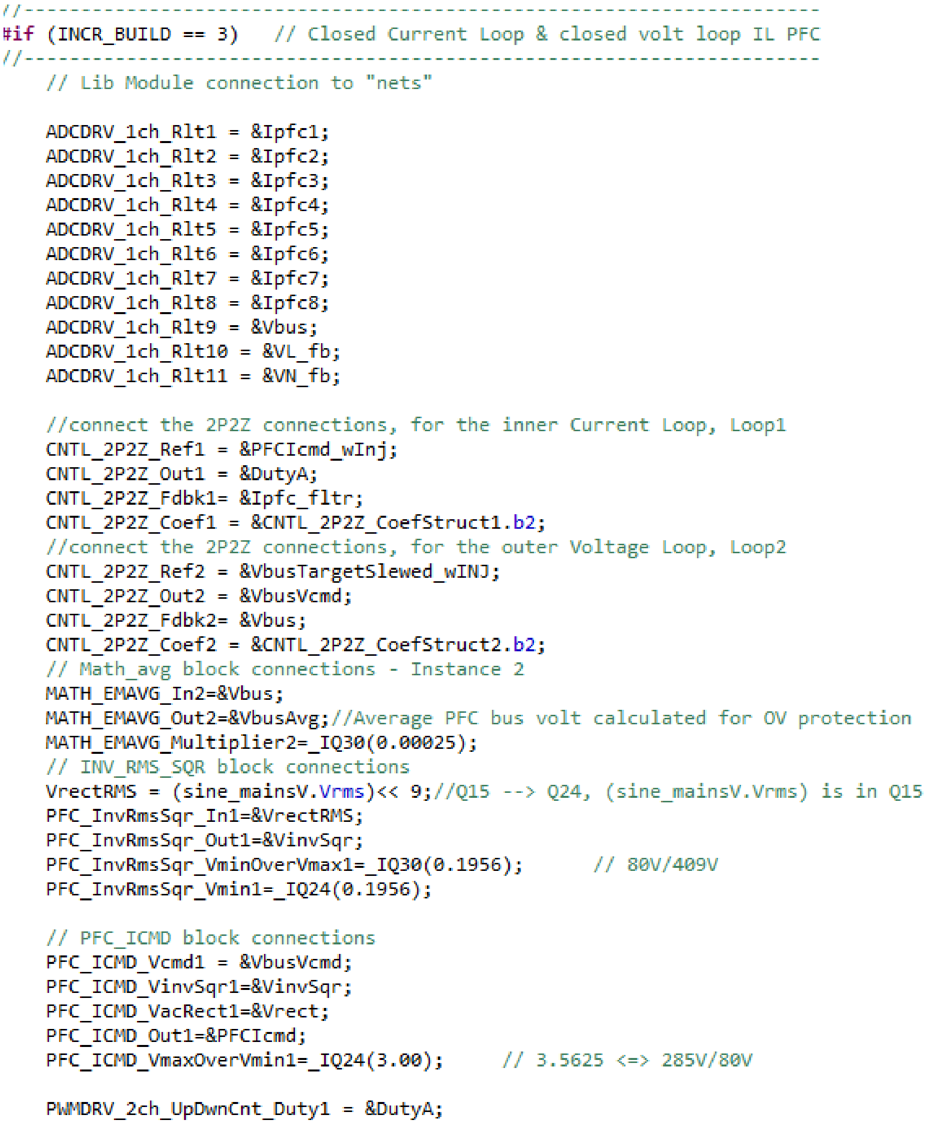
Conduction Mode: CCM

Figure 18. PowerSUITE GUI

To tune the current loop using the Compensation Designer and the SFRA tool, do as follows:

1. Select the correct options from the Control Loop Design section.
2. Follow the procedure in Section 5.2.1.
3. Locate and inspect the following code in the ILPFC-Main file under initialization code specific for build.

> **NOTE:** This code is where all the software blocks related to Build 3 are connected in the control flow.



```
//----------------------------------------------------------------------
#if (INCR_BUILD == 3)   // Closed Current Loop & closed volt loop IL PFC
//----------------------------------------------------------------------
    // Lib Module connection to "nets"

    ADCDRV_1ch_Rlt1 = &Ipfc1;
    ADCDRV_1ch_Rlt2 = &Ipfc2;
    ADCDRV_1ch_Rlt3 = &Ipfc3;
    ADCDRV_1ch_Rlt4 = &Ipfc4;
    ADCDRV_1ch_Rlt5 = &Ipfc5;
    ADCDRV_1ch_Rlt6 = &Ipfc6;
    ADCDRV_1ch_Rlt7 = &Ipfc7;
    ADCDRV_1ch_Rlt8 = &Ipfc8;
    ADCDRV_1ch_Rlt9 = &Vbus;
    ADCDRV_1ch_Rlt10 = &VL_fb;
    ADCDRV_1ch_Rlt11 = &VN_fb;

    //connect the 2P2Z connections, for the inner Current Loop, Loop1
    CNTL_2P2Z_Ref1 = &PFCIcmd_wInj;
    CNTL_2P2Z_Out1 = &DutyA;
    CNTL_2P2Z_Fdbk1= &Ipfc_fltr;
    CNTL_2P2Z_Coef1 = &CNTL_2P2Z_CoefStruct1.b2;
    //connect the 2P2Z connections, for the outer Voltage Loop, Loop2
    CNTL_2P2Z_Ref2 = &VbusTargetSlewed_wINJ;
    CNTL_2P2Z_Out2 = &VbusVcmd;
    CNTL_2P2Z_Fdbk2= &Vbus;
    CNTL_2P2Z_Coef2 = &CNTL_2P2Z_CoefStruct2.b2;
    // Math_avg block connections - Instance 2
    MATH_EMAVG_In2=&Vbus;
    MATH_EMAVG_Out2=&VbusAvg;//Average PFC bus volt calculated for OV protection
    MATH_EMAVG_Multiplier2=_IQ30(0.00025);
    // INV_RMS_SQR block connections
    VrectRMS = (sine_mainsV.Vrms)<< 9;//Q15 --> Q24, (sine_mainsV.Vrms) is in Q15
    PFC_InvRmsSqr_In1=&VrectRMS;
    PFC_InvRmsSqr_Out1=&VinvSqr;
    PFC_InvRmsSqr_VminOverVmax1=_IQ30(0.1956);      // 80V/409V
    PFC_InvRmsSqr_Vmin1=_IQ24(0.1956);

    // PFC_ICMD block connections
    PFC_ICMD_Vcmd1 = &VbusVcmd;
    PFC_ICMD_VinvSqr1=&VinvSqr;
    PFC_ICMD_VacRect1=&Vrect;
    PFC_ICMD_Out1=&PFCIcmd;
    PFC_ICMD_VmaxOverVmin1=_IQ24(3.00);     // 3.5625 <=> 285V/80V

    PWMDRV_2ch_UpDwnCnt_Duty1 = &DutyA;
```

4. Open and inspect the ILPFC-DPL-ISR.asm file.
5. Notice the _DPL_Init and _DPL_Func sections under Build 3.

> **NOTE:** In this code, the macro instantiations under Build 3 occur for initialization and runtime, respectively.

6. In the CCS Project Explorer window, click *ILPFC [Active-F2803x_Flash]*.
7. Right-click and select *Rebuild Project*.
8. After the project is built, click 🐞 ▾ from the CCS menu bar.

NOTE: The program builds and loads into flash and takes you to the start of Main().

9. Enable *Real-time Mode*.
10. Select *Continuous Refresh* for the watch window expressions.
11. Click *Run* on the toolbar to run the code.
12. In the watch view, add the VbusTargetSlewed expression.
13. Add the Vbus expression.
14. Set the Q-format to Q24.

NOTE: The expressions represent the pullup reference bus voltage and the feedback bus voltage, respectively (normalized or per unit values) and slowly increase to the setpoint value as the PFC starts when AC power is applied.

15. Apply an appropriate resistive load to PFC output.

NOTE: For example, a 8.0-kΩ resistor of 40-W rating can be used. This resistor provides a load of about 20 W at 390-V bus voltage.

16. Configure the isolated AC source to 130-V output, 60-Hz output, and the AC voltage output.
17. Use a voltmeter to monitor the DC-bus voltage.
18. Turn on the AC-source output for 130 Vrms.

NOTE: When the DC-bus voltage reaches 160 V, PFC action starts and the bus voltage slowly increase to about 390 V. Notice that VbusTargetSlewed and Vbus expressions on the watch window show a value of about 0.7414 (390 ÷ 526) when the Q format is set to Q24. The maximum bus voltage set by the Vbus sense resistors is about 526 V that corresponds to maximum ADC input of 3.3 V. The normalized or per unit value is about 0.7414 when the actual bus voltage is 390 VDC.

19. Use an oscilloscope with voltage and current probes to observe the input voltage and input current.

NOTE: With 110-Vrms input, 277-Ω resistive load, and bus voltage set to 390 V, the waveforms shown in Figure 19 should appear. In Figure 19, Ch2 is the input voltage and Ch4 is the input current.

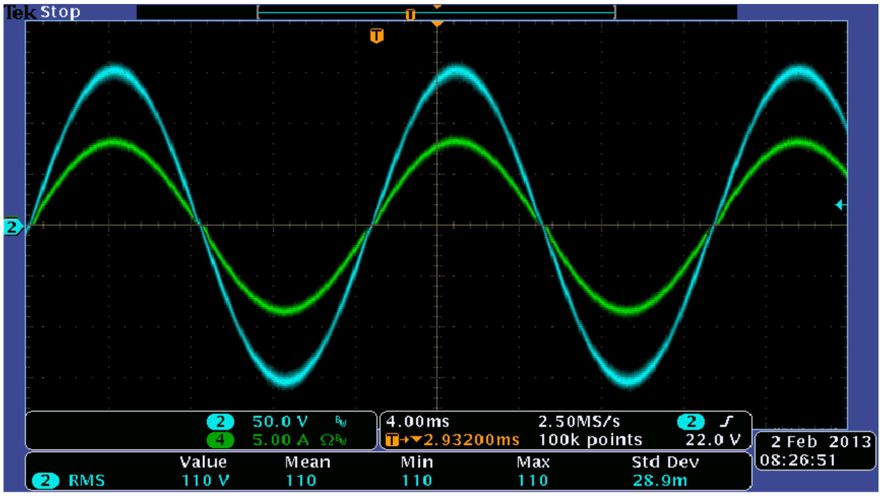


Figure 19. ILPFC Input Voltage and Current at 550-W Load and 110-Vrms Input

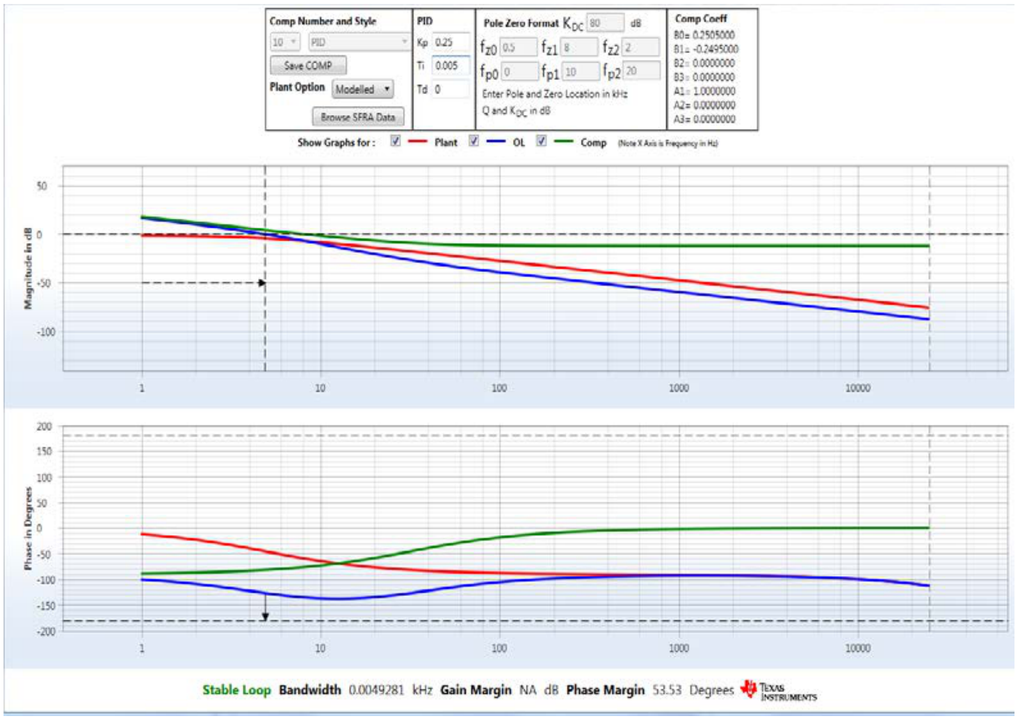
20. Change the input voltage (100 Vrms to approximately 260 Vrms) or the load resistance (0 to approximately 550 W at 110 Vin or 0 to approximately 700 W at 220 Vin) to see the PFC operation under closed current and voltage-control loop.
21. From the *main.cfg* page, click the *Compensation Designer* under the *Control Loop Design* section.

NOTE: The *Compensation Designer* GUI shows the voltage-loop Bode plots with the modeled PFC voltage-loop power stage and the voltage-loop compensator COMPV2.

22. Note the loop bandwidth (BW), the phase margin (PM) and the gain margin (GM) of the voltage loop.

NOTE: The GUI indicates the style or type of compensator (PID), the PID coefficients, and the corresponding compensation coefficients (An, Bn).

23. Note the default compensation coefficients used in this build. This is the voltage loop controller COMPV2. Figure 20 shows a snapshot of the compensation designer GUI.



Comp Number and Style: 10, PID

Plant Option: Modelled

PID: Kp 0.25, Ti 0.005, Td 0

Pole Zero Format: K_{DC} 80 dB

f_{z0} 0.5, f_{z1} 8, f_{z2} 2

f_{p0} 0, f_{p1} 10, f_{p2} 20

Enter Pole and Zero Location in kHz
Q and K_{DC} in dB

Comp Coeff
B0= 0.2505000
B1= -0.2495000
B2= 0.0000000
B3= 0.0000000
A1= 1.0000000
A2= 0.0000000
A3= 0.0000000

Show Graphs for : ☑ Plant ☑ OL ☑ Comp (Note X Axis is Frequency in Hz)

Stable Loop Bandwidth 0.0049281 kHz Gain Margin NA dB Phase Margin 53.53 Degrees

Figure 20. Compensation Designer GUI Snapshot

24. Close the *Compensation Designer*.

At this point, the user can run the SFRA, measure the voltage-loop Bode plots, and compare the measured plots with those noted in Step 23:

25. To run the SFRA, set the PFC resistive load so it draws at least 400 W (550-W maximum) at a 390-V bus.
26. Use the SFRA tool to check the loop-gain plot for the voltage loop.

NOTE: The SFRA GUI saves the measured voltage-loop power stage response data that can be used to retune the voltage-loop controller for optimum performance.

27. Close the SFRA GUI.
28. Open the *Compensation Designer*.
29. Under *Plant Option* selection, click *Browse SFRA Data*.

30. Select the measured power stage response data that the SFRA GUI saved in the ILPFC project folder from the most recent SFRA run.
31. Change the *Plant Option* to *SFRA Data*.
32. Adjust the voltage-loop compensator coefficients for optimum loop-frequency response.

NOTE: The GUI shows the *Comp Coeff* for the voltage loop.

33. Use these coefficients to replace the default compensation coefficients COMPV2 noted in Step 23.
34. Click *Save COMP* from the Compensation Designer GUI to copy the new coefficients to the appropriate compensation settings (COMPV2) in the ILPFC_Base-Settings.h file.

NOTE: Increasing the voltage loop bandwidth improves the transient performance and degrades the input-power factor. Some trade-off exists between the DC-bus transient performance and input-power factor. Use a second voltage-loop compensator only during the transient condition to avoid this trade-off. For TI ILPFC kit, this second compensator is COMPV1 that yields higher voltage-loop bandwidth during transient conditions. Under steady-state operation, the lower-bandwidth compensator COMPV2 is used.

35. Turn off AC power.
36. Reset the MCU.
37. Recompile and reload the ILPFC project with the new coefficients.
38. Reload the ILPFC project.
39. Enable *Real-time Mode.*
40. Select *Continuous Refresh* for the watch window expressions.
41. Click *Run* on the toolbar to check the performance of the new voltage controller.

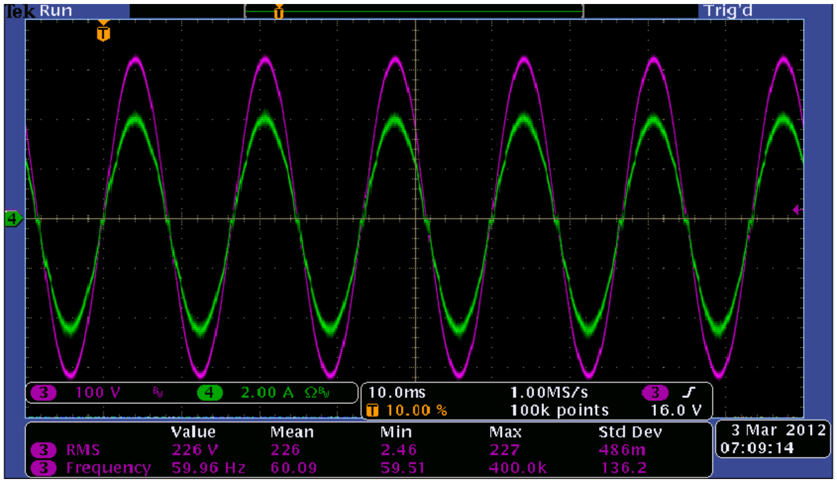
NOTE: The Bode plots from the Compensation Designer and the SFRA should match more closely.

42. Turn off AC power.
43. Reset the MCU.

End of Exercise

6 Additional Test Results

The following waveforms are additional test results.



3 100 V 4 2.00 A Ω

10.0ms T 10.00 %

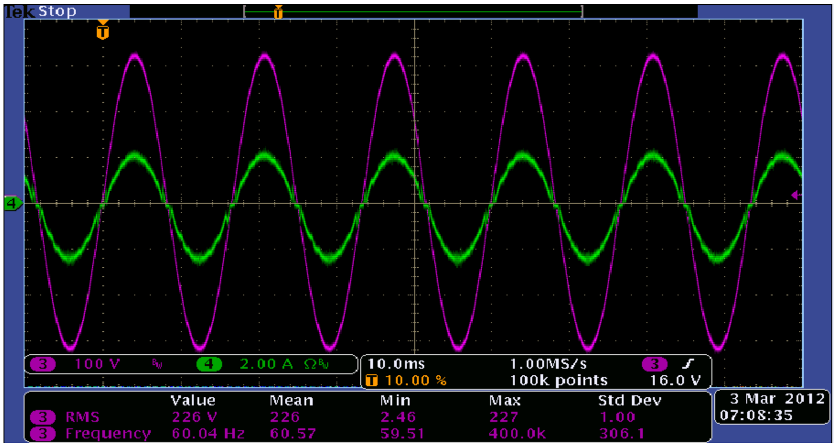
1.00MS/s 100k points

3 16.0 V

	Value	Mean	Min	Max	Std Dev
3 RMS	226 V	226	2.46	227	486m
3 Frequency	59.96 Hz	60.09	59.51	400.0k	136.2

3 Mar 2012 07:09:14

Figure 21. ILPFC Input Voltage and Current Waveforms, Ch3–Vin, Ch4-Iin, Vrms = 230 V, Vbus = 390 V, Pout = 700 W



3 100 V 4 2.00 A Ω

10.0ms T 10.00 %

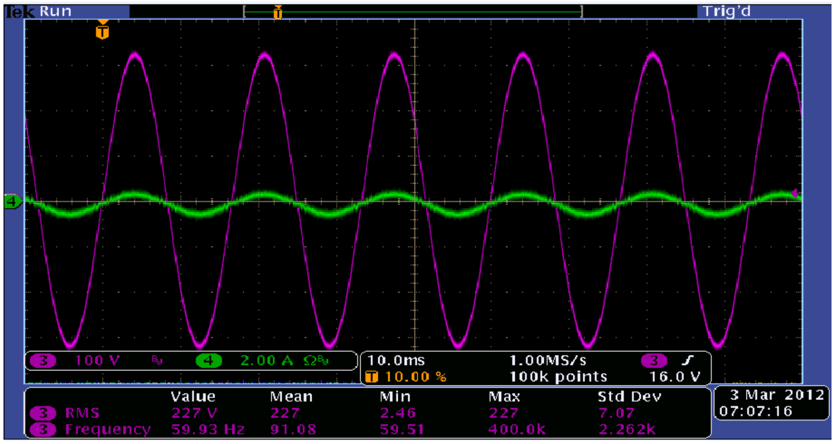
1.00MS/s 100k points

3 16.0 V

	Value	Mean	Min	Max	Std Dev
3 RMS	226 V	226	2.46	227	1.00
3 Frequency	60.04 Hz	60.57	59.51	400.0k	306.1

3 Mar 2012 07:08:35

Figure 22. ILPFC Input Voltage and Current Waveforms, Ch3–Vin, Ch4-Iin, Vrms = 230 V, Vbus = 390 V, Pout = 375 W



3 100 V 4 2.00 A Ω

10.0ms T 10.00 %

1.00MS/s 100k points

3 16.0 V

	Value	Mean	Min	Max	Std Dev
3 RMS	227 V	227	2.46	227	7.07
3 Frequency	59.93 Hz	91.08	59.51	400.0k	2.262k

3 Mar 2012 07:07:16

Figure 23. ILPFC Input Voltage and Current Waveforms, Ch3–Vin, Ch4-Iin, Vrms = 230 V, Vbus = 390 V, Pout = 80 W

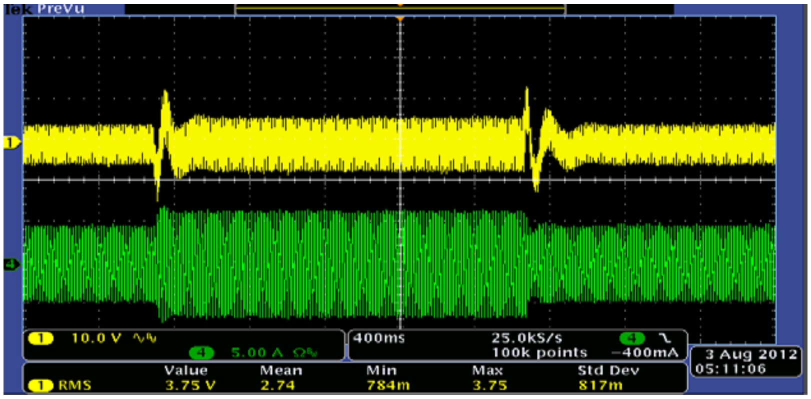


Figure 24. ILPFC DC Bus Load Transient Response, Ch1–Vbus, Ch4-Iin, Vrms = 120 V, Vbus = 390 V, Load Step 360 W to Approximately 520 W

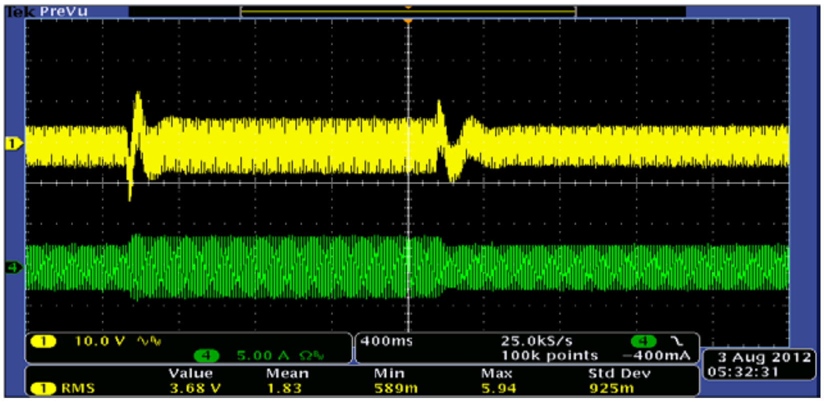


Figure 25. ILPFC DC Bus Load Transient Response, Ch1–Vbus, Ch4-Iin, Vrms = 220 V, Vbus = 390 V, Load Step 370 W to approximately 530 W

7 References

For more information, see the following guides:

- ILPFC-PowerSUITE Quick Start Guide—A quick-start guide for demonstration of the ILPFC EVM using a GUI interface (..\controlSUITE\development_kits\ILPFC_v1.1 \~Docs\ILPFC-PowerSUITE-QSG.pdf).
- ILPFC_Rel-1.0-HWdevPkg—A folder containing various files related to the Piccolo-B controller card schematics and the ILPFC schematic (..\controlSUITE\development_kits\ILPFC_v1.1\ILPFC_HWDevPkg).
- *TMS320F28032, TMS320F28033, TMS320F28034, TMS320F28035 Piccolo Microcontrollers* data manual (SPRS584)
- *TMS320F28027, TMS320F28026, TMS320F28023, TMS320F28022, TMS320F28021, TMS320F28020, TMS320F280200 Piccolo Microcontrollers* data manual (SPRS523)
- *TMS320F28069, TMS320F28068, TMS320F28067, TMS320F28066, TMS320F28065, TMS320F28064, TMS320F28063, TMS320F28062 Piccolo Microcontrollers* data manual (SPRS698)

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products (also referred to herein as “components”) are sold subject to TI’s terms and conditions of sale supplied at the time of order acknowledgment.

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

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

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 relating to any combination, machine, or process in which TI components or services are used. Information published by TI regarding third-party products or services does not constitute a license to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

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

Resale of TI components or services with statements different from or beyond the parameters stated by TI for that component or service voids all express and any implied warranties for the associated TI component or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

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

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

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

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

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

Products		**Applications**	
Audio	www.ti.com/audio	Automotive and Transportation	www.ti.com/automotive
Amplifiers	amplifier.ti.com	Communications and Telecom	www.ti.com/communications
Data Converters	dataconverter.ti.com	Computers and Peripherals	www.ti.com/computers
DLP® Products	www.dlp.com	Consumer Electronics	www.ti.com/consumer-apps
DSP	dsp.ti.com	Energy and Lighting	www.ti.com/energy
Clocks and Timers	www.ti.com/clocks	Industrial	www.ti.com/industrial
Interface	interface.ti.com	Medical	www.ti.com/medical
Logic	logic.ti.com	Security	www.ti.com/security
Power Mgmt	power.ti.com	Space, Avionics and Defense	www.ti.com/space-avionics-defense
Microcontrollers	microcontroller.ti.com	Video and Imaging	www.ti.com/video
RFID	www.ti-rfid.com		
OMAP Applications Processors	www.ti.com/omap	**TI E2E Community**	e2e.ti.com
Wireless Connectivity	www.ti.com/wirelessconnectivity		

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265