Reference Designs: TIDM-DC-DC-BUCK

C2000™ Digital Power BoosterPack™

Description

TI Designs provide the foundation that you need including methodology, testing and design files to quickly evaluate and customize the system. TI Designs help *you* accelerate your time to market.

Resources

TIDM-DC-DC-BUCK	Tool Folder Containing Design Files

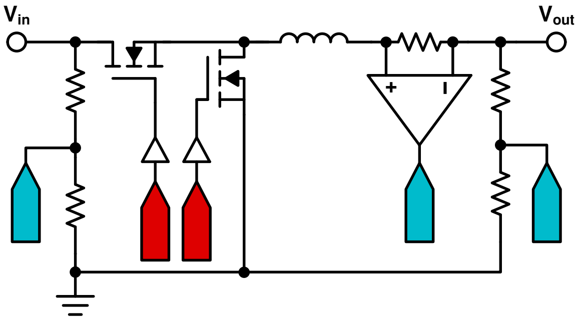
Ask our TI E2E™ support experts

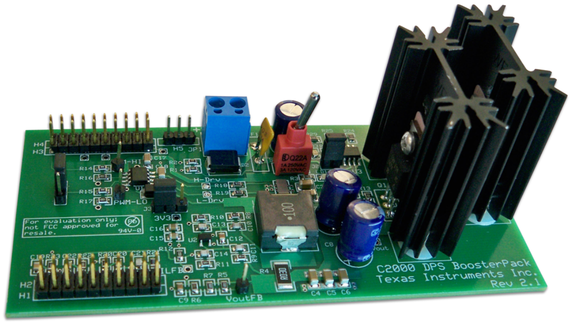
Features

- Features a Non-Isolated, Digitally-Controlled DC-DC Buck Converter
- Offers a Quick and Easy Way to Learn About Digital Power Supply Control Using C2000™ MCUs
  - Controlled by the F280049C LaunchPad
  - Voltage Mode Control (VMC)
  - Peak Current Mode Control (PCMC)
- Offers an Onboard Active Load for Transient Performance Testing
- Offers Various powerSUITE Tools
  - Software Frequency Response Analyzer (SFRA)
  - Compensation Designer
  - Solution Adapter

Applications

- Server Power Supplies
- Telecom Rectifiers
- Industrial Power Supplies
- UPS Systems
- Smart Grid and Energy
- Automotive Charging
- Data Storage





An IMPORTANT NOTICE at the end of this TI reference design addresses authorized use, intellectual property matters and other important disclaimers and information.

1 Introduction

This document offers a step-by-step guide to controlling the C2000™ Digital Power Buck Converter BoosterPack (BOOSTXL-BUCKCONV) power stage using Voltage Mode Control (VMC) and Peak Current Mode Control (PCMC). Both control mode examples start with open-loop excitation before proceeding to a tunable closed-loop system. The reader is expected to have a basic understanding of power converters and control using feedback loops.

The BOOSTXL-BUCKCONV provides a quick and easy way to learn about digital power supply control and design using C2000 devices. This board consists of a DC-DC synchronous buck power stage controlled by a compatible C2000 LaunchPad™. The example solutions for this guide use the LAUNCHXL-F280049C. [1]

The software accompanying this design is packaged in the DigitalPower SDK for C2000. These examples allow for programming the TMS320F280049C microcontroller (MCU) and experimenting with control parameters to tune the loop for optimal system performance. This kit supports powerSUITE tools including Compensation Designer, Software Frequency Response Analyzer (SFRA), and Solution Adapter for the evaluation of the complete system.

1.1 Trademarks

C2000, BoosterPack, TI E2E, LaunchPad, Code Composer Studio, NexFET are trademarks of Texas Instruments.
All other trademarks are the property of their respective owners.

2 Hardware and Resources Guide

To get started, this kit requires the following:

- TMS320F280049C LaunchPad (LAUNCHXL-F280049C)
- Micro-USB cable for powering and interfacing with the LAUNCHXL-F280049C
- C2000 Digital Power Buck Converter BoosterPack (BOOSTXL-BUCKCONV)
- 9-VDC, 2-A power supply with 1/4" or 6 mm of exposed wire leads
- 1/8" or 3-mm flat-head screwdriver
- Host computer with:
  - Code Composer Studio™ (CCS) v9.3 or later installed [2]
  - DigitalPower SDK for C2000 v3.00.00.00 or later installed [3]

[1] The first release of TIDM-DC-DC-BUCK used LAUNCHXL-F28069M as the controller LaunchPad. This F2806x-based user's guide and accompanying software can be downloaded with controlSUITE.

[2] The powerSUITE tools for TIDM-DC-DC-BUCK require CCS v9.3 or later.

[3] The examples for TIDM-DC-DC-BUCK are available in C2000Ware_DigitalPower_SDK v3.00.00.00 and later.

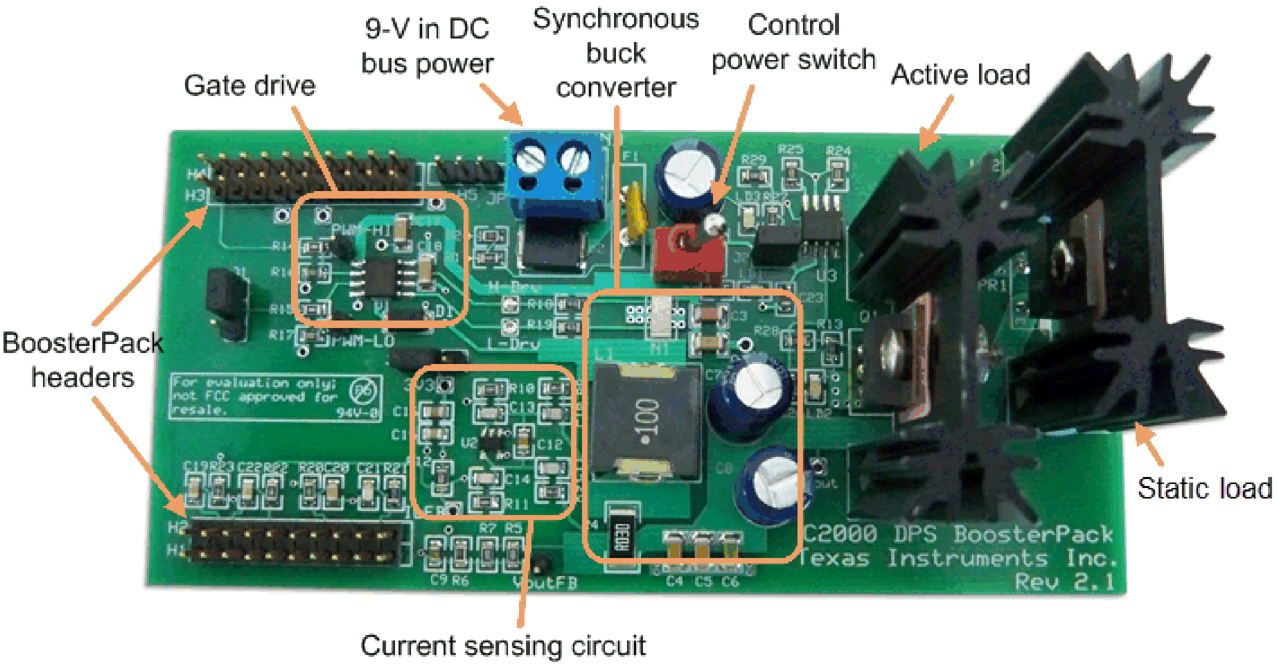


Figure 1. BOOSTXL-BUCKCONV and Key Components

The LaunchPad interfaces with the buck converter through the BoosterPack headers. The nominal input bus voltage is 9 VDC, which can be supplied through the power terminal block at JP1. The DC-DC synchronous buck power stage is designed to regulate the nominal output voltage at 2 VDC. The active load provides a software-controlled switched load, which is useful for testing transient performance and loop tuning.

Table 1. Key Components of BOOSTXL-BUCKCONV

Component	Description
9-VDC In (JP1)	Vin - DC-power supply (Minimum 8.5 VDC, Maximum 12 VDC) Recommended Supply: 9 VDC, 2 A, efficiency level V[1]
BoosterPack Headers	LaunchPad interface
Control Power Switch	SW1 - Power switch to connect the buck converter to Vin
Buck Converter	Synchronous buck power stage (includes TI NexFET™ Power Block)
Static Load	Permanently connected 7.5-Ω resistive load
Active Load	Software-controlled 2-Ω resistive load
Gate Drive	Buffered PWM drive to ensure quick and efficient turn on or off of the power FETs
Current Sensing	Inductor current feedback for current monitoring and overcurrent protection

[1] TI recommends using an external power supply that complies with applicable regional safety standards such as (by example) UL, CSA, VDE, CCC, PSE, and so forth.

The key interface signals between the LAUNCHXL-F280049C and the BOOSTXL-BUCKCONV are described in Table 2, followed by the associated portion of the schematic.

Table 2. Key Signal Connections

BOOSTXL-BUCKCONV	Description	LAUNCHXL-F280049C Site 1 (J1-J4)	LAUNCHXL-F280049C Site 2 (J5-J8) [(1)]
H3[1] PWM-HI	High-side drive signal for synchronous buck	J4[40] GPIO10 (EPWM6_A)	J8[80] GPIO0 (EPWM1_A)
H3[2] PWM-LO	Low-side drive signal for synchronous buck	J4[39] GPIO11 (EPWM6_B)	J8[79] GPIO1 (EPWM1_B)
H3[5] PWM-Load	PWM duty control signal for active load	J4[36] GPIO4 (EPWM3_A)	J8[76] GPIO2 (EPWM2_A)
H2[7] VoutFB	Output voltage feedback	J3[27] ADCINB2	J7[67] ADCINC3
H2[9] ILFB	Inductor current feedback (overcurrent protection)	J3[29] ADCINA9	J7[69] ADCINA3
H2[4] ILFB_AVG	Inductor current feedback (heavily filtered average)	J3[24] ADCINB0	J7[64] ADCINA2
H2[8] VinFB	Input voltage feedback	J3[28] ADCINC0	J7[68] ADCINC5

[(1)] The silk screen positions of J6 and J8 are reversed on the top-side of LAUNCHXL-F280049C Rev A (MCU025A). See the *LAUNCHXL-F280049C User's Guide* for more information.

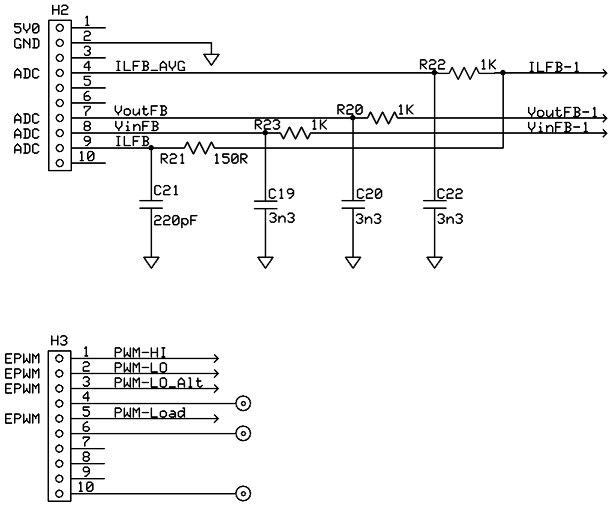


Figure 2. BoosterPack™ Signals

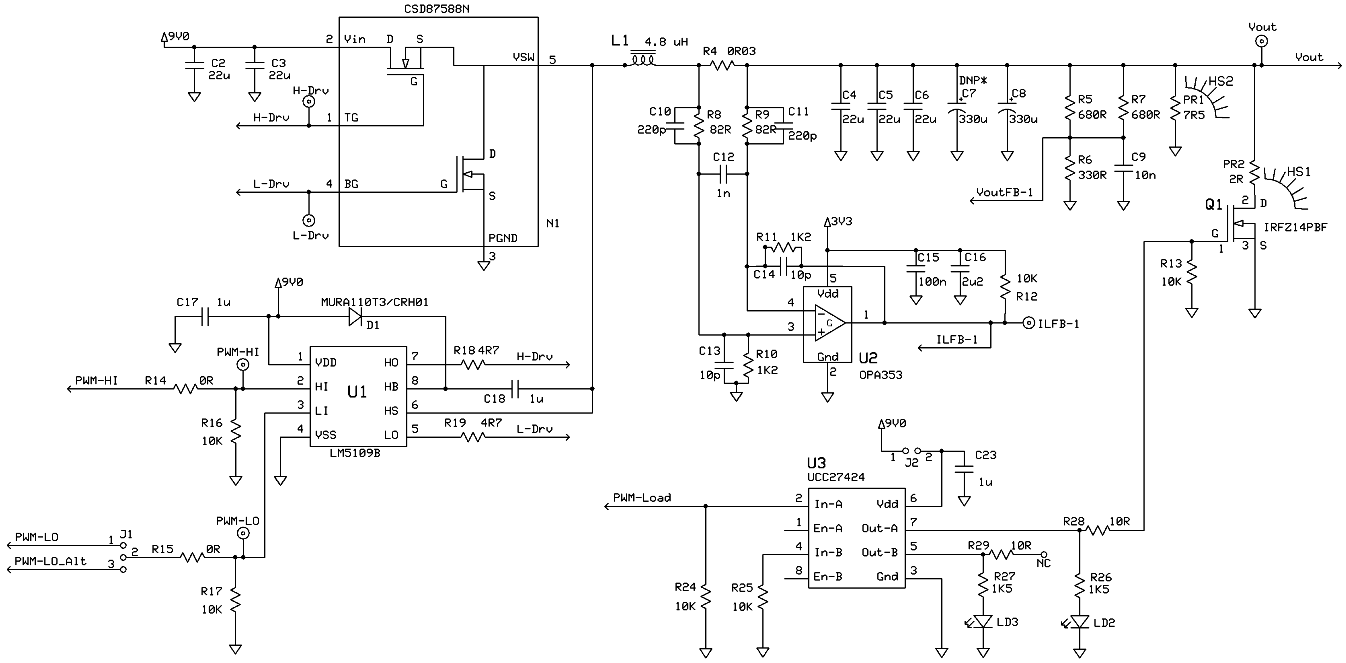


Figure 3. Power Stage Schematic

The F280049C MCU on the LAUNCHXL-F280049C controls the synchronous buck operation by driving the high-side and low-side switches of the NexFET power block N1 using on-chip PWM signals operated at 200-kHz. Resistor R4 and op amp U2 are used to sense the current of inductor L1 and feed the representative analog signal back to the MCU.

A static load PR1 is permanently connected between the synchronous buck output and ground. A dynamic active load PR2 can be switched in and out of the circuit by driving the MOSFET switch Q1 with a 50% duty cycle PWM signal. This capability provides an active load feature at run time to support transient performance testing and tuning. PR2 can also be turned on continuously to increase output load which causes the inductor current to stay positive and nonzero, thus providing better loop response measurements.

WARNING

This EVM is intended to be operated in a lab environment and is considered by TI to be an unfinished product fit for general consumer use. This EVM must be used only by qualified engineers and technicians familiar with the risks associated with handling electrical and mechanical components, systems, and subsystems.

3 Software Overview

3.1 Software Flow

The software project uses C-code to implement a C-Background and C-ISR framework. The background loop of the main application manages all system tasks, decision making, intelligence, and host interaction. The real-time critical actions such as ADC reading, control updates, and PWM updates are executed inside the Interrupt Service Routine (ISR). Figure 4 shows te software flow for this project.

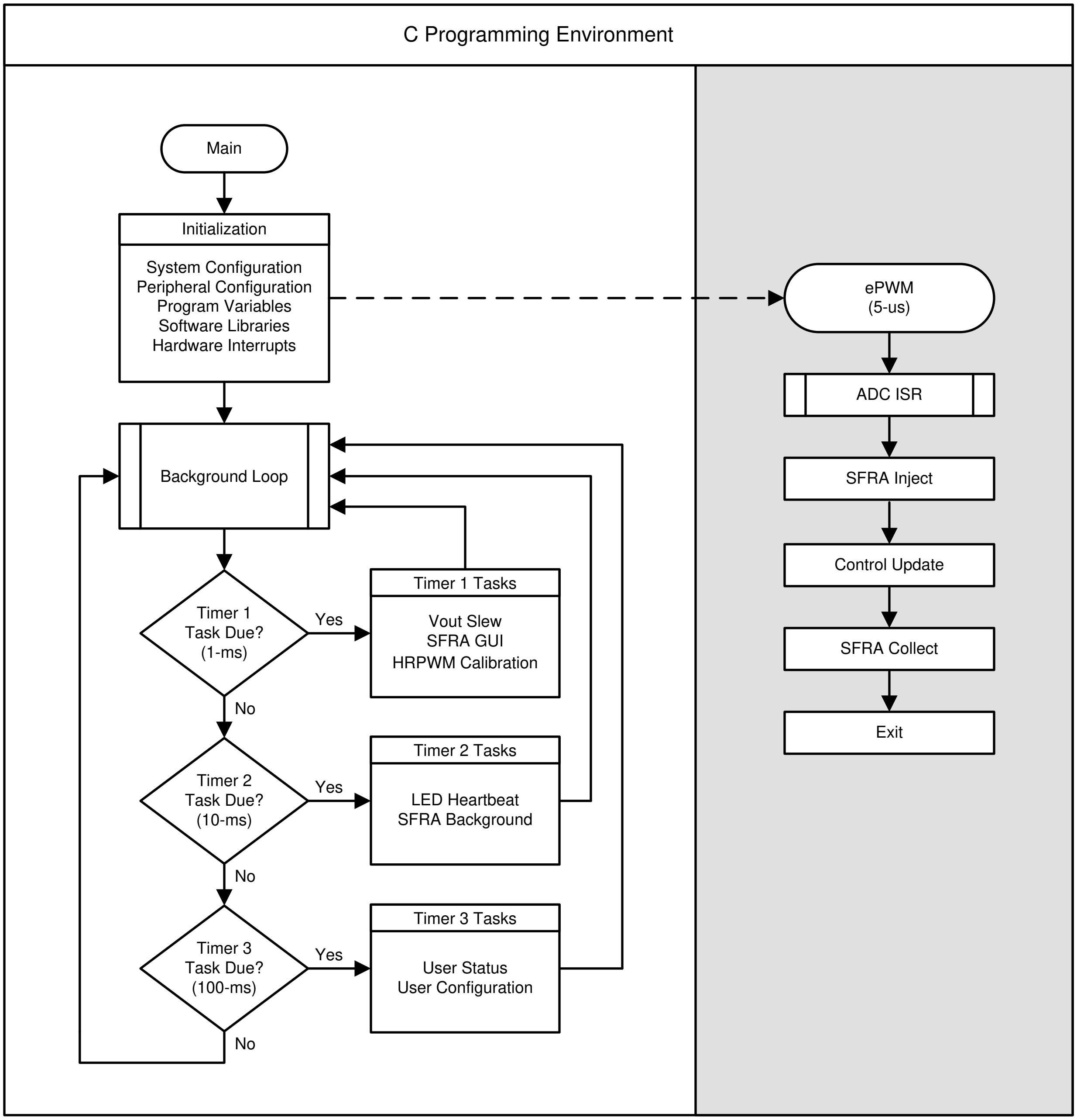
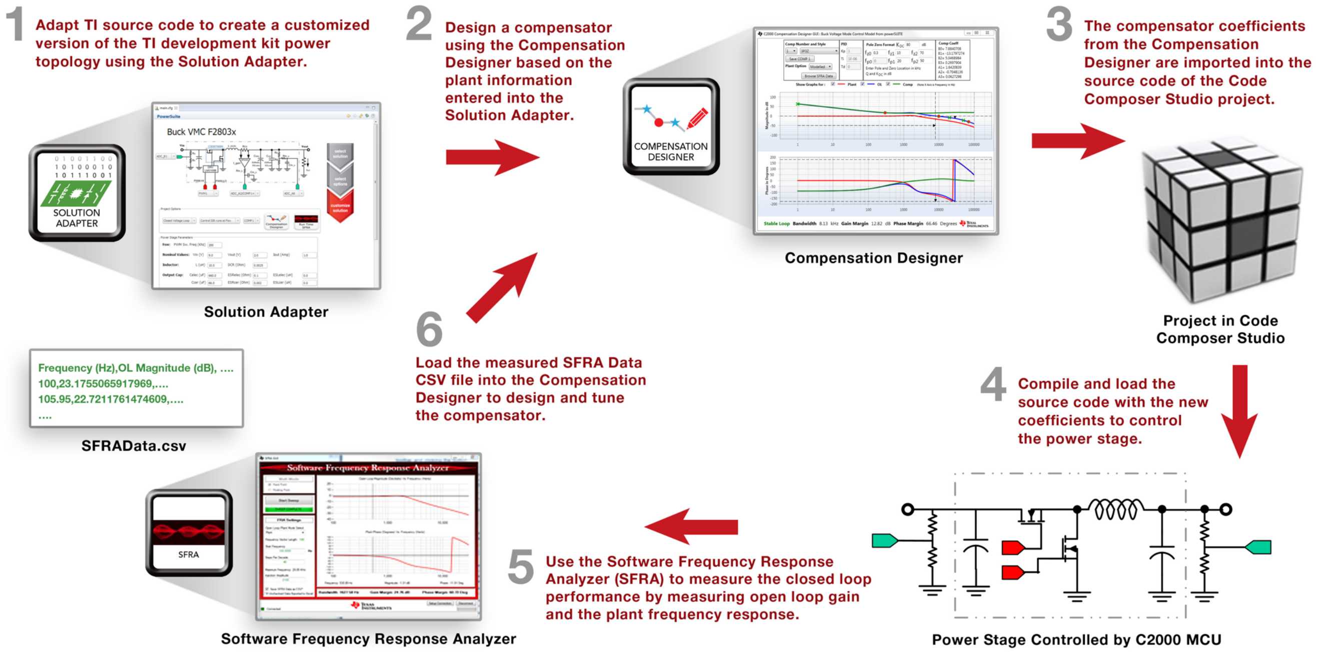


Figure 4. Software Flow Chart

The SFRA library is designed to enable frequency response analysis on digitally-controlled power converters using software, without requiring external lab equipment. For high-frequency power conversion applications, the SFRA library can measure the open-loop and plant characteristics of the system to calculate loop stability parameters such as bandwidth, gain margin, and phase margin. Refer to the C2000 Software Frequency Response Analyzer (SFRA) Library and Compensation Designer in SDK Framework User's Guide for more information.

This kit also supports Compensation Designer and Solution Adapter for system evaluation, tuning, and porting to custom hardware. Figure 5 shows the typical process flow for designing and tuning 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,....
....

Figure 5. Designing and Tuning a System Process Flow Chart

Solution Adapter demonstrates how the TI digital power supply kit example can be ported to a custom digital power supply system that uses the same topology and similar MCU resources. The GUI provides an easy interface for selecting the solution to adapt, selecting the relevant options for that solution, and customizing those options to other hardware designs.

Compensation Designer is a software utility that helps with visualizing the behavior of the compensated control system. The effects of the user-designed compensation scheme are estimated through the analysis of system response data from SFRA measurements or from a simulated model. Solution Adapter incorporates the generated compensator coefficients directly into the software project.

NOTE: In this guide, Compensation Designer support is only available for the VMC lab.

The key framework C files in this project are:

- buck_settings.h - Global definitions for Solution Adapter customization
- buck_user_settings.h - Global definitions for abstracting system-specific resources
- buck_main.c - Main function and background loop to initialize, run, and manage the application
- buck.c and buck.h - Functions that are common to managing a buck converter
- buck_hal.c and buck_hal.h - Functions to configure and control device-specific resources

The key libraries used in this project are:

- SFRA - SFRA framework
- DCL - C2000 Digital Control Library
- SFO - HRPWM calibration
- FPU and TMU - Optimized math functions
- driverlib - Peripheral driver library for configuring device-specific resources

The primary source files and CCS project are intended to be imported into the user workspace so that file modifications made in the workspace will not affect the original source files from the SDK directory.

3.2 Incremental Labs

This guide is divided into four incremental labs to introduce the system features in a logical progression:

Lab_1_OpenLoopVmc — Open-Loop check for VMC. Check PWM drive and sensing circuits.

Lab_2_ClosedLoopVmc — Closed-Loop operation with VMC. Evaluate full system functionality.

Lab_3_OpenLoopPcmc — Open-Loop check for PCMC. Check PWM drive and sensing circuits.

Lab_4_ClosedLoopPcmc — Closed-Loop operation with PCMC. Evaluate full system functionality.

4 Lab 1: Open-Loop Check for VMC

4.1 Objective

The objective of this lab is to verify the operation of the VMC hardware using an open-loop system. The user will become familiar with the system hardware and how the buck output voltage can be controlled using direct PWM duty cycle adjustments without a feedback loop. Because this system is running as an open-loop, the ADC measured values are used only for user observation in this build. The PWM duty cycle is adjusted manually using the Expressions window. A CCS graph will be used to visualize the sensed output voltage, and SFRA will be used to measure the frequency response of the plant at run-time.

4.2 Overview

Lab 1 is configured to accept user input from the CCS Expressions window to set the duty cycle of the PWM output that drives the complementary PWM-HI and PWM-LO signals on the BOOSTXL-BUCKCONV.

A single Enhanced Pulse Width Modulator (ePWM) module controls the PWM signals and also triggers ADC conversions to sense the plant feedback signals at a frequency of 200-kHz. Updates to the PWM duty cycle are prepared in the ADC ISR such that they take affect in the following PWM cycle.

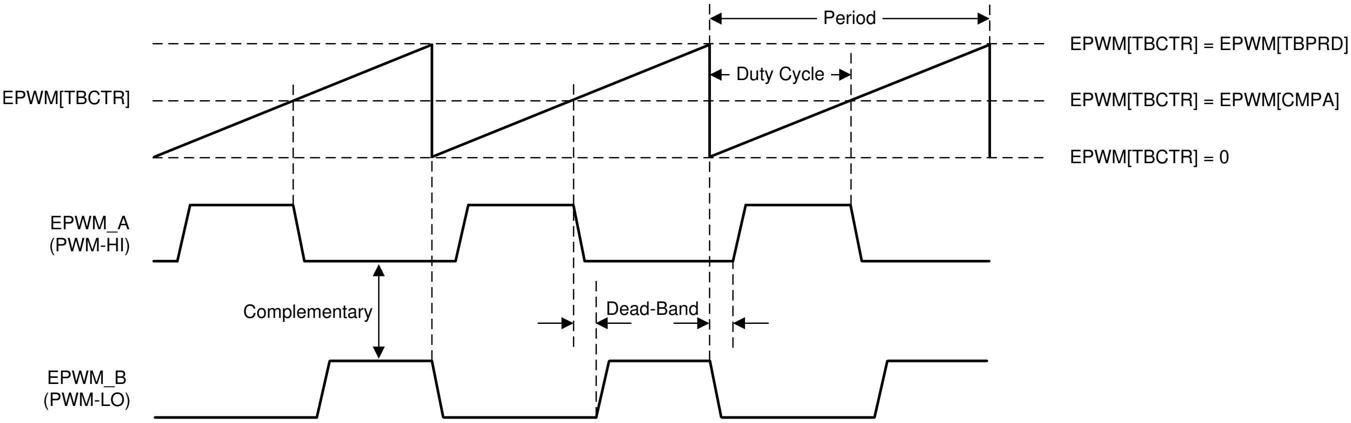


Figure 6. PWM Signal Generation for VMC

In this project, the ePWM Time Base Counter Register (TBCTR), is configured to count-up from 0 until it matches the Time Base Period Register (TBPRD). Once the period is reached, EPWM[TBCTR] will reset to 0 on the next clock cycle, which will mark the beginning of the next ePWM period. Therefore, the effective PWM frequency is determined by the amount of time needed for EPWM[TBCTR] to count from 0 to EPWM[TBPRD], plus one additional clock cycle to reset EPWM[TBCTR] back to 0.

The duty-cycle is controlled through the Counter Compare A Register (CMPA). At the beginning of each ePWM period, the PWM-HI signal (controlled by EPWM_A) is driven high until EPWM[TBCTR] matches EPWM[CMPA]. Upon reaching EPWM[CMPA], the ePWM will drive PWM-HI low until the next ePWM period.

The ePWM Dead-Band Generator Submodule is configured to output a PWM-LO signal (controlled by EPWM_B) that is complementary to PWM-HI, with inserted dead-band.

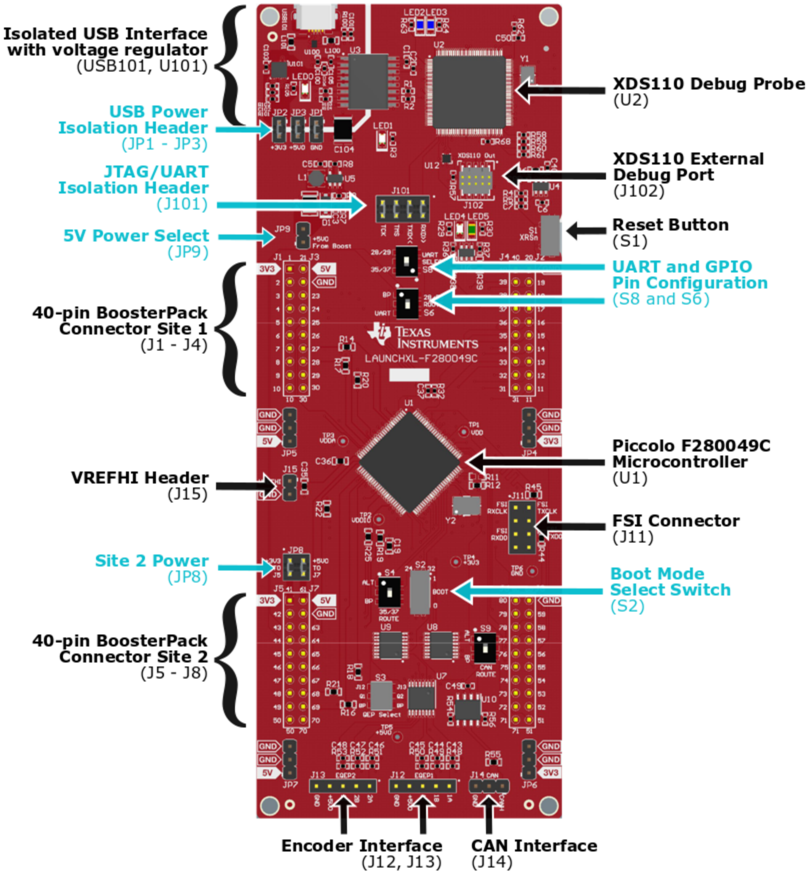
An on-chip analog Comparator Subsystem (CMPSS), with integrated reference DAC, is configured to continuously monitor the current sense feedback (ILFB) signal for overcurrent events. In this project, an overcurrent trip event will drive the PWM-HI signal low to stop conduction of the input voltage, and the PWM-LO signal will continue to operate for debug visibility.

The main() background loop uses a timer-based state machine to schedule low-priority recurring tasks, such as accepting user input. Three task scheduling queues are paced using three C28x CPU Timers that are configured to expire at interval periods of 1 ms (A-tasks), 10 ms (B-tasks), and 100 ms (C-tasks). Each priority queue will execute a single task in round-robin sequence whenever its respective pace timer expires and resets.

4.3 Procedure

4.3.1 Hardware Setup

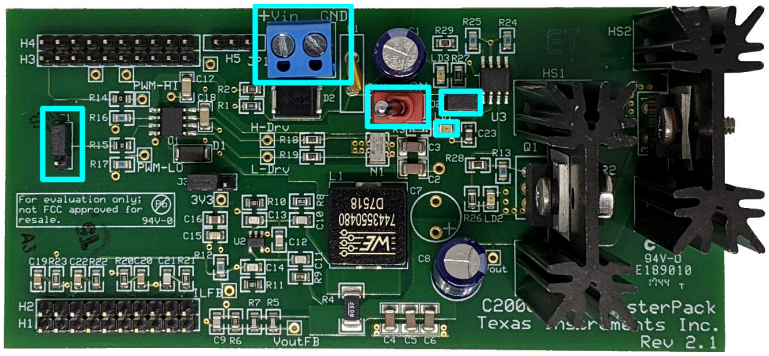
1. To get started with the BoosterPack, first ensure that the required components listed in Section 2 are available.
2. Configure the LAUNCHXL-F280049C:



- Install jumpers on JP1, JP2, and JP3 to share power between the XDS110 and the rest of the LaunchPad
- Install jumpers on RXD, TXD, TMS, and TCK of J101 to enable XDS110 emulation and COM port access
- Uninstall jumper on JP9 to use the 5-V supply from the USB port
- Install jumper on 3V3 of JP8 to make power available on Connector Site 2 [4] (this jumper is optional if using Connector Site 1)
- Select GPIO28 and GPIO29 for UART pin selection on S8
- Select UART for GPIO28 and GPIO29 routing on S6
- Select Wait Boot (GPIO24 = On and GPIO32 = Off) on S2

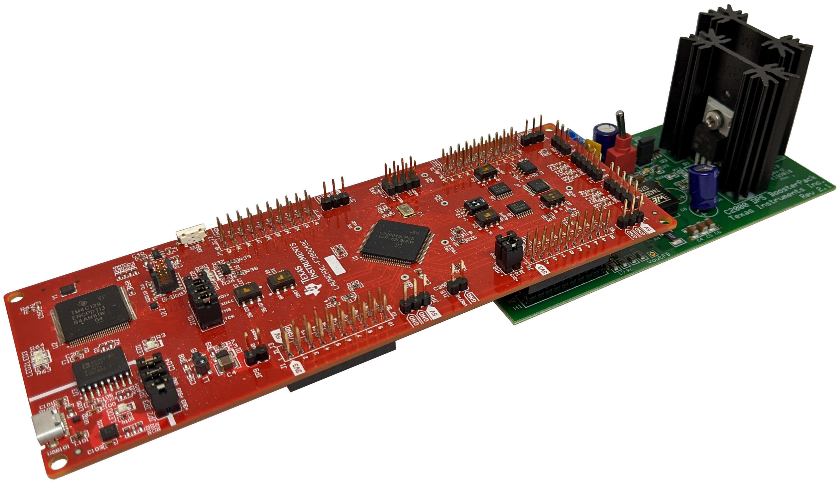
[4] A jumper on 5V0 of JP8 will have no effect.

3. Configure the BOOSTXL-BUCKCONV:



- Install a jumper between J1[1] and J1[2] to select signal H3[2] as the source for PWM-LO
- Install a jumper on J2 to provide power to the MOSFET driver of the active load
- Turn off toggle switch SW1 (pointing away from the heatsinks) to isolate JP1 from the plant
- Insert the lead wires of the *de-energized* 9-VDC power supply into terminal block JP1:
  - Follow the *+Vin* and *GND* polarity markings
  - Tighten the terminal screws with the screwdriver to secure the wires in place
  - Green light LD1 should be off

4. Plug the BOOSTXL-BUCKCONV into Site 2 (headers J5-J8) of the LAUNCHXL-F280049C [5]

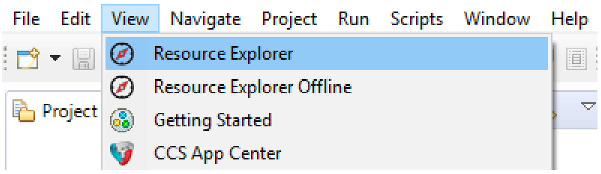


5. Connect the LAUNCHXL-F280049C to the host computer using the micro-USB cable
6. Red lights LED0 and LED1 on the LAUNCHXL-F280049C should turn on

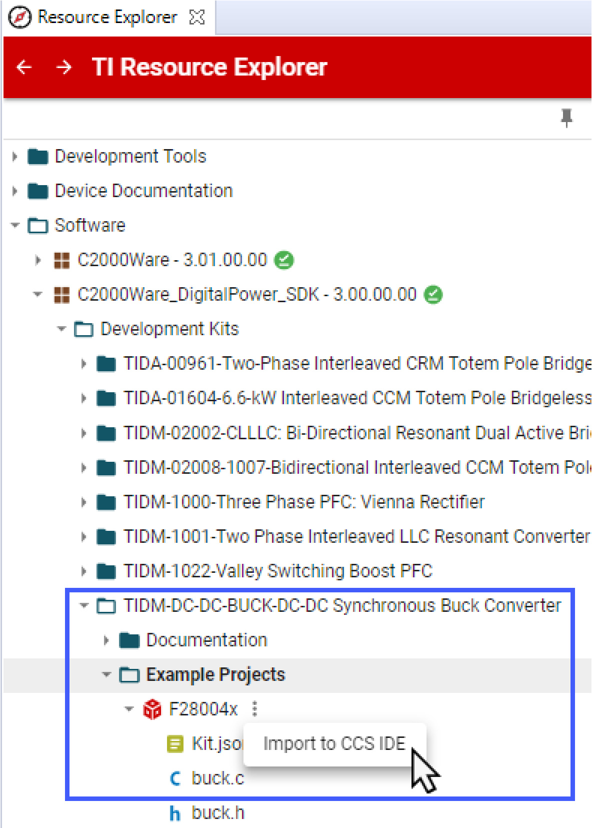
[5] See Section 9.1 for additional plug-in options.

4.3.2 Software Setup

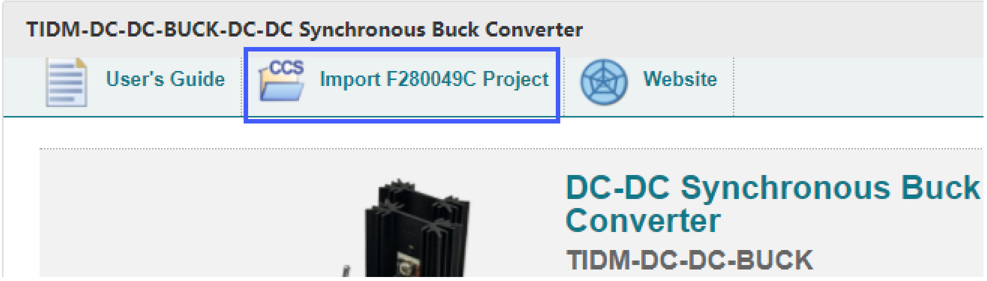
1. Open CCS
2. Open Resource Explorer:
   - View Menu » Resource Explorer



3. Navigate to the TIDM-DC-DC-BUCK Kit:
   - Software » C2000Ware_DigitalPower_SDK » Development Kits » TIDM-DC-DC-BUCK

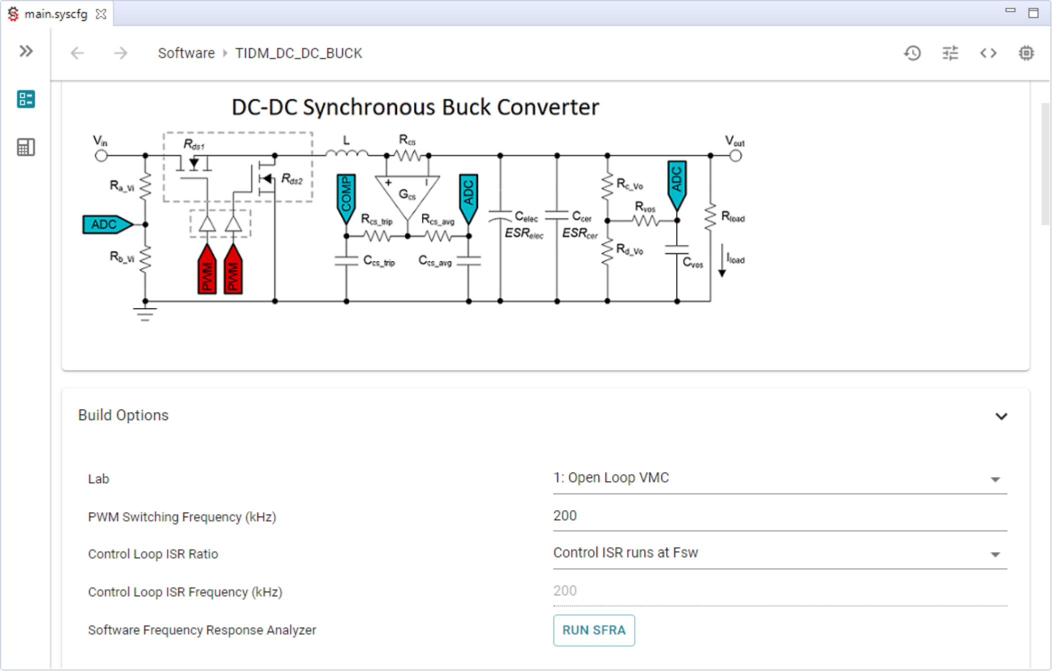

Resource Explorer
TI Resource Explorer
Development Tools
Device Documentation
Software
C2000Ware - 3.01.00.00
C2000Ware_DigitalPower_SDK - 3.00.00.00
Development Kits
TIDA-00961-Two-Phase Interleaved CRM Totem Pole Bridge
TIDA-01604-6.6-kW Interleaved CCM Totem Pole Bridgeless
TIDM-02002-CLLLC: Bi-Directional Resonant Dual Active Bri
TIDM-02008-1007-Bidirectional Interleaved CCM Totem Pol
TIDM-1000-Three Phase PFC: Vienna Rectifier
TIDM-1001-Two Phase Interleaved LLC Resonant Converter
TIDM-1022-Valley Switching Boost PFC
TIDM-DC-DC-BUCK-DC-DC Synchronous Buck Converter
Documentation
Example Projects
F28004x
Kit.jso
Import to CCS IDE
buck.c
buck.h

4. Click on either the *Import to CCS IDE* shortcut from the navigation tree or the *Import F280049C Project* shortcut from the kit page


TIDM-DC-DC-BUCK-DC-DC Synchronous Buck Converter
User's Guide
CCS
Import F280049C Project
Website
DC-DC Synchronous Buck Converter
TIDM-DC-DC-BUCK

5. If prompted, select *Save the project into workspace* and click the *OK* button

6. The *buck_F28004x* project will be imported into the Project Explorer window and the *main.syscfg* panel will launch automatically



Build Options	
Lab	1: Open Loop VMC
PWM Switching Frequency (kHz)	200
Control Loop ISR Ratio	Control ISR runs at Fsw
Control Loop ISR Frequency (kHz)	200
Software Frequency Response Analyzer	RUN SFRA

4.3.3 Build and Load Lab 1

1. In *main.syscfg*, configure the project settings as Table 3 shows:

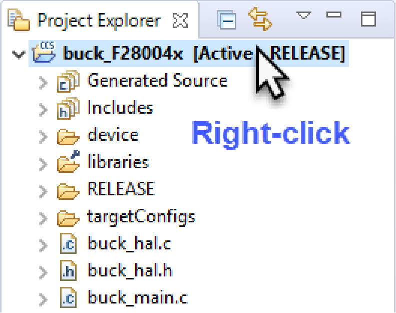
Table 3. Lab 1 Project Settings

Parameter	Value
Lab	1: Open Loop VMC
PWM Switching Frequency (kHz)	200
Control Loop ISR Ratio	Control ISR runs at Fsw
Plant	BOOSTXL-BUCKCONV
LaunchPad Site	Site 2 [1]

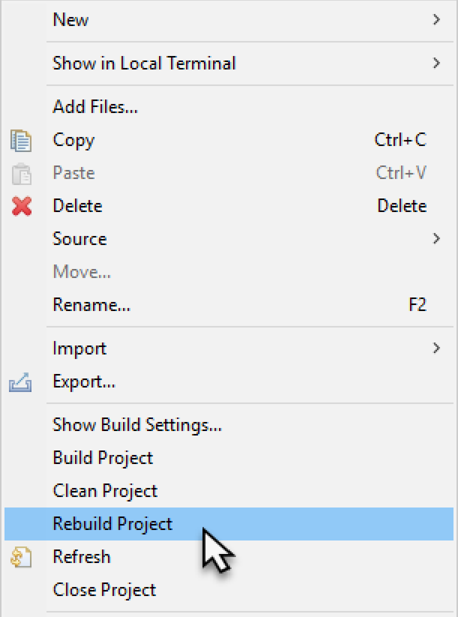
[1] Select *Site 1* if using headers J1-J4 instead of J5-J8.

NOTE: The *main.syscfg* panel can be launched manually by double-clicking on the *main.syscfg* file in the Project Explorer pane

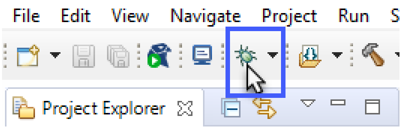
2. Right-click on the *buck_F28004x* project in the Project Explorer window to activate the context menu



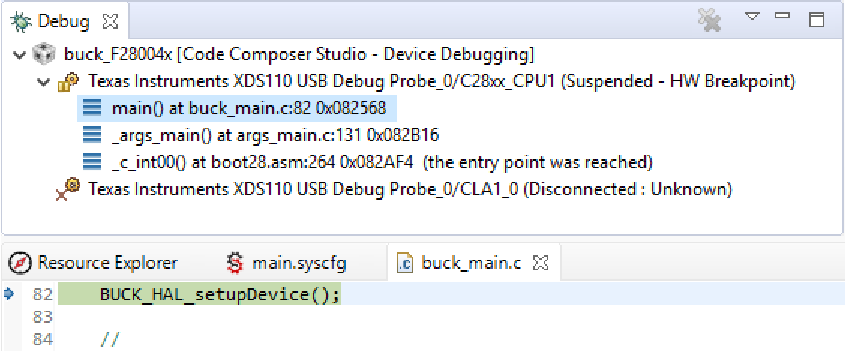
3. Select *Rebuild Project* to build the project using the settings configured with *main.syscfg*



4. The Console window will display build activity and indicate completion with **** *Build Finished* ****
5. Click on the *buck_F28004x* project in the Project Explorer window to select it as the focused project
6. Click on the *Debug* shortcut to launch the target configuration and load the project to LAUNCHXL-F280049C



7. The C28xx CPU of the F280049C should now be connected and halted in *buck_main.c*, ready to run the program in the CCS Debug perspective

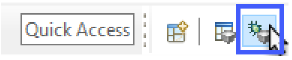


```
buck_F28004x [Code Composer Studio - Device Debugging]
  Texas Instruments XDS110 USB Debug Probe_0/C28xx_CPU1 (Suspended - HW Breakpoint)
    main() at buck_main.c:82 0x082568
    _args_main() at args_main.c:131 0x082B16
    _c_int00() at boot28.asm:264 0x082AF4  (the entry point was reached)
  Texas Instruments XDS110 USB Debug Probe_0/CLA1_0 (Disconnected : Unknown)
```

```
    BUCK_HAL_setupDevice();

    //
```

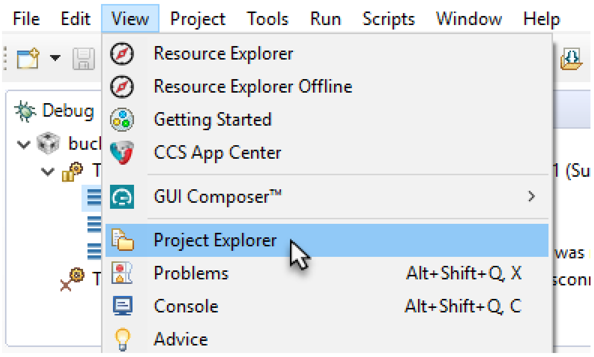
The Debug perspective is indicated in the top-right corner of CCS



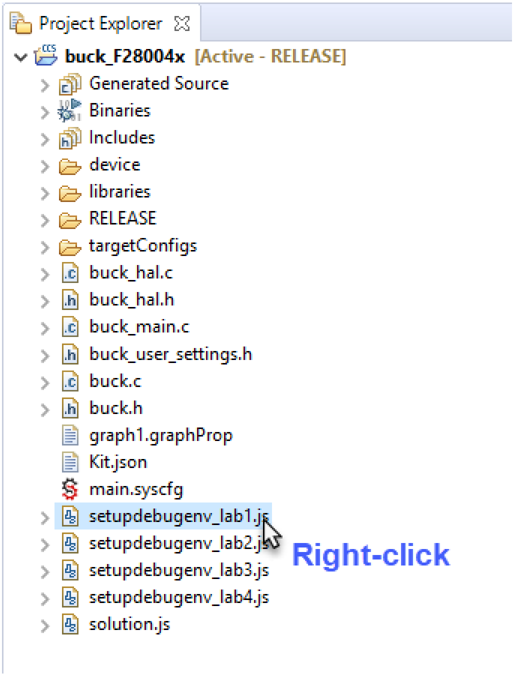
4.3.4 Prepare the Debug Environment

CCS provides the ability to inspect and modify the value of program variables during execution using the Expressions window. Array variables can also be visualized through the use of Graph plots. The imported project includes scripting and configuration files to quickly assemble the basic debug environment for each lab.

1. Open the Project Explorer window in the CCS Debug perspective:
   - View » Project Explorer



2. Populate the Expressions window:
   a. Right-click on the *setupdebugenv_lab1.js* file in the Project Explorer window to activate the context menu



b. Click on the *Run Script* option to clear and fill the Expressions window with the Lab 1 debug variables described in Table 4

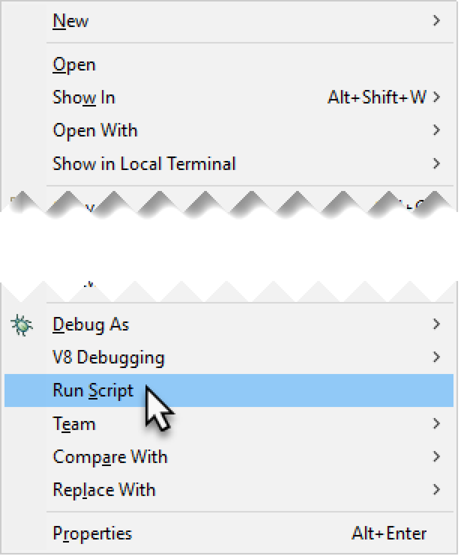
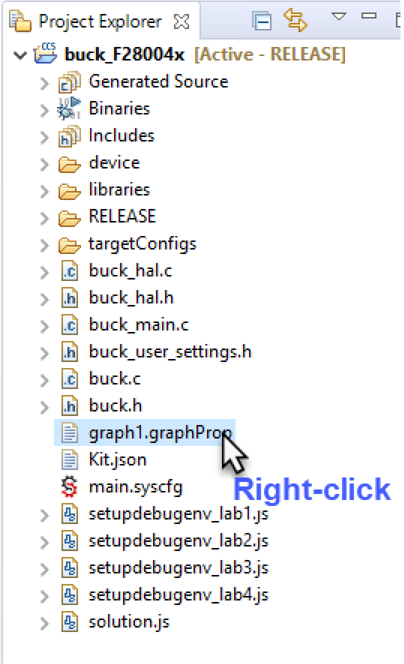


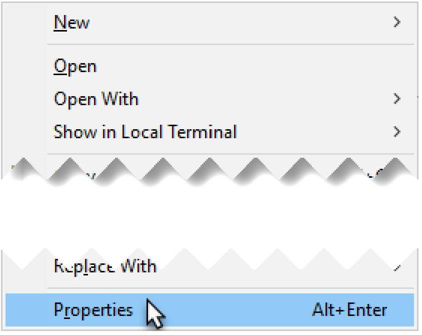
Table 4. Lab 1 Expressions Window Variables

Variable	User Access	Description
BUCK_labNumber	Read	Lab build that is running on the MCU
BUCK_dutySetRef_pu	Read and Write	PWM duty cycle requested by the user
BUCK_dutySet_pu	Read	PWM duty cycle used for programming the ePWM
BUCK_vInSensed_Volts	Read	Input voltage sensed from JP1
BUCK_vOutSensed_Volts	Read	Output voltage sensed from the plant
BUCK_iOutSensed_Amps	Read	Low-pass filtered (average) inductor current
BUCK_iOutTripSetRef_Amps	Read and Write	Overcurrent trip (shut-down) threshold for monitoring the unfiltered inductor current
BUCK_iOutTripFlag	Read	Overcurrent trip status 0 = Normal Operation 1 = Tripped
BUCK_iOutTripFlagClear	Write	Request to clear overcurrent trip status Write 0 = No effect Write 1 = Clear trip condition (variable will self-clear to 0)
BUCK_activeLoadEnable	Read and Write	Enable the active load 0 = Switched resistor isolated 1 = Switched resistor connected as determined by BUCK_activeLoadContEnable
BUCK_activeLoadContEnable	Read and Write	Enable continuous load (when BUCK_activeLoadEnable = 1) 0 = Switched resistor is connected and disconnected periodically 1 = Switched resistor is connected continuously
BUCK_vOutLogRef	Read and Write	Structure to configure sampling of the output voltage for graphing

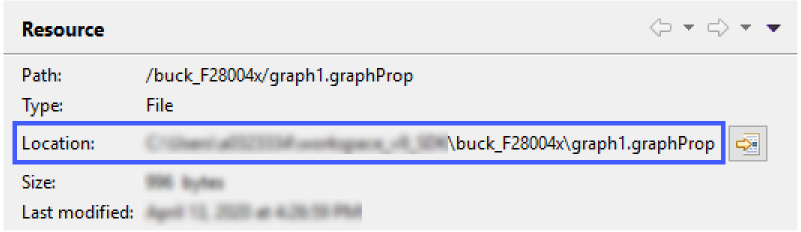
3. Add a graph for the output voltage:
   a. Right-click on the *graph1.graphProp* file in the Project Explorer window to activate the context menu



   b. Click on *Properties*



   c. Record this file location for use in Step 3f



Resource

Path: /buck_F28004x/graph1.graphProp

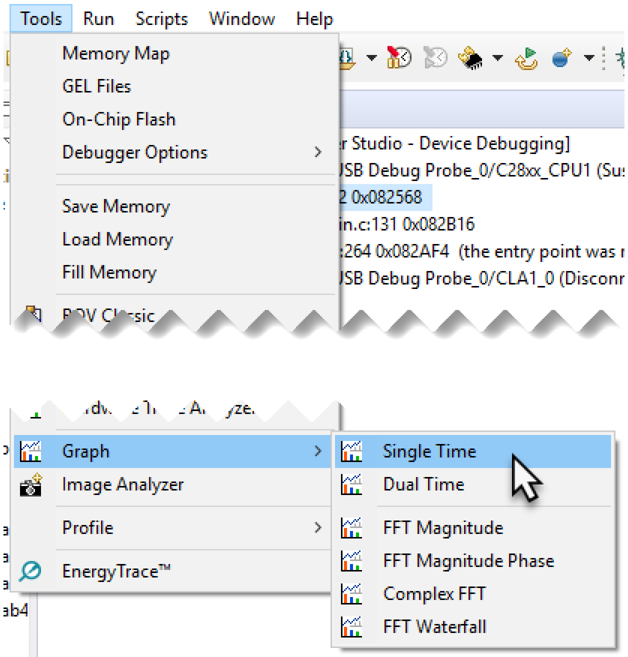
Type: File

Location: [redacted]\buck_F28004x\graph1.graphProp

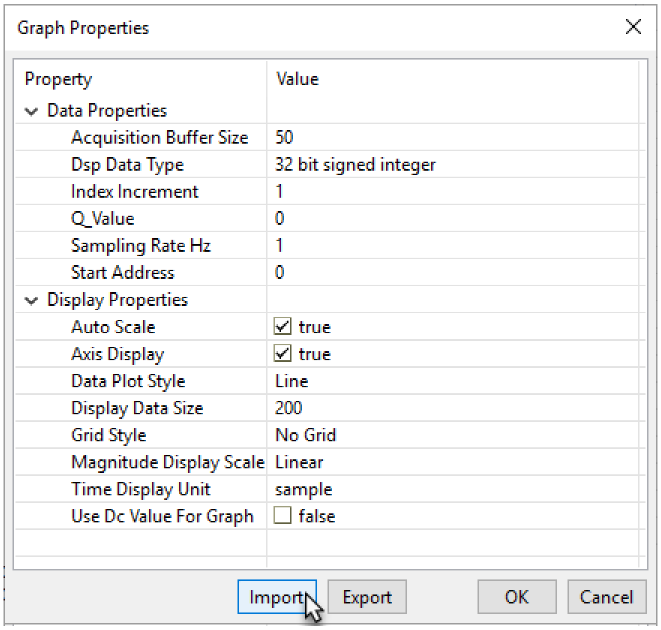
Size: [redacted]

Last modified: [redacted]

d. Tools » Graph » Single Time



e. Click on the *Import* button to load saved settings


Graph Properties

Property	Value
Data Properties	
Acquisition Buffer Size	50
Dsp Data Type	32 bit signed integer
Index Increment	1
Q_Value	0
Sampling Rate Hz	1
Start Address	0
Display Properties	
Auto Scale	☑ true
Axis Display	☑ true
Data Plot Style	Line
Display Data Size	200
Grid Style	No Grid
Magnitude Display Scale	Linear
Time Display Unit	sample
Use Dc Value For Graph	□ false

f. Select *graph1.graphProp* from the location found in Step 3c

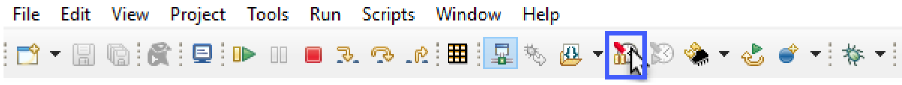
g. Click the *Open* button

h. Click the *Ok* button to create the graph window

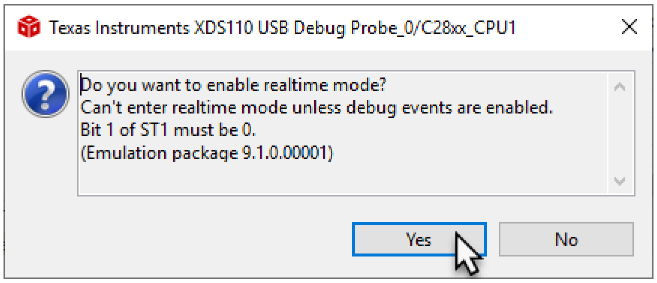
4.3.5 Enable Real-time Debug

Real-time emulation is a feature that allows Code Composer Studio to support debug activities with less intrusiveness while the MCU is running. This capability allows variables and memory locations to be modified interactively to change run-time behavior like tuning control law parameters on-the-fly. ISR servicing is allowed to continue while the CPU appears to be halted.

1. Enable Real-time Mode:
   a. Click on the *Enable Silicon Real-time Mode* shortcut



   b. If prompted, click on *Yes* to enable debug events

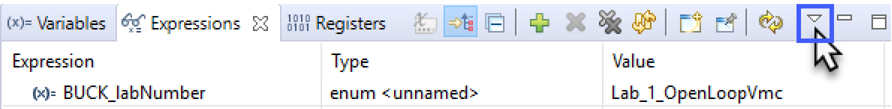


Texas Instruments XDS110 USB Debug Probe_0/C28xx_CPU1

Do you want to enable realtime mode?
Can't enter realtime mode unless debug events are enabled.
Bit 1 of ST1 must be 0.
(Emulation package 9.1.0.00001)

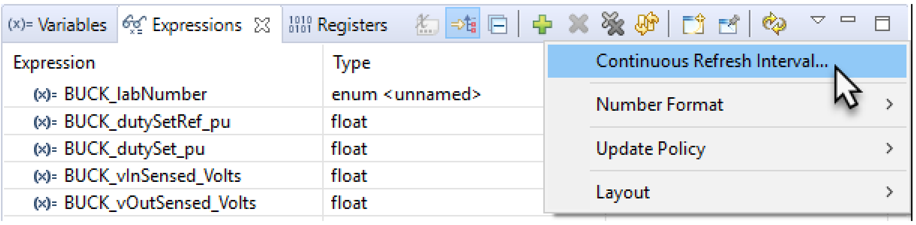
NOTE: Selecting *Yes* will clear the Debug Enable Mask (DBGM) bit of Status Register 1 (ST1). When DGBM is cleared to 0, memory and register values can be passed to the host processor for updating the debugger windows.

2. Enable Continuous Refresh:
   a. Click on the drop-down menu expansion shortcut in the Expressions window



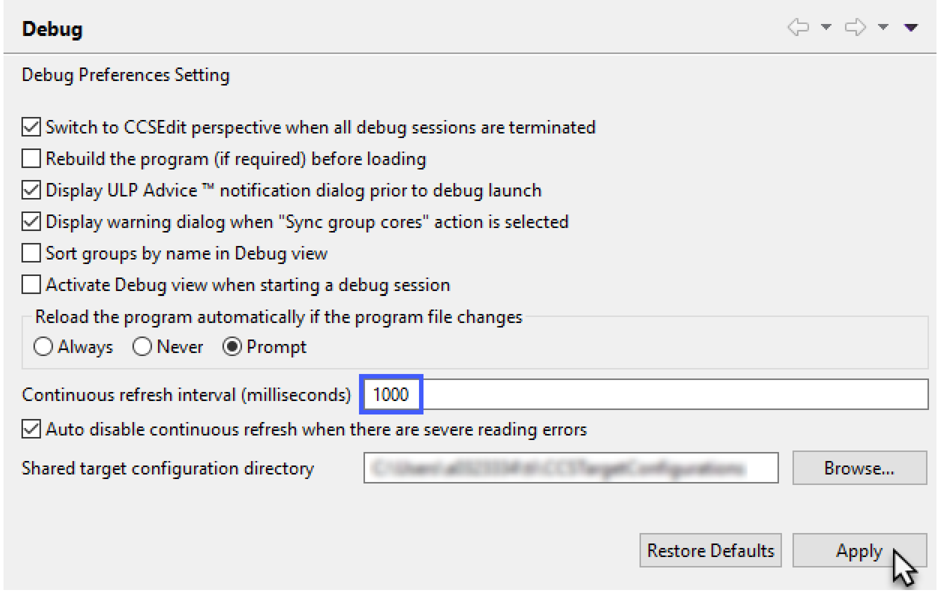
Expression	Type	Value
(x)= BUCK_labNumber	enum <unnamed>	Lab_1_OpenLoopVmc

   b. Select *Continuous Refresh Interval...*



Expression	Type
(x)= BUCK_labNumber	enum <unnamed>
(x)= BUCK_dutySetRef_pu	float
(x)= BUCK_dutySet_pu	float
(x)= BUCK_vInSensed_Volts	float
(x)= BUCK_vOutSensed_Volts	float

c. Increase the *Continuous refresh interval* value to 1000 ms and click on *Apply*



Debug

Debug Preferences Setting

- ☑ Switch to CCSEdit perspective when all debug sessions are terminated
- □ Rebuild the program (if required) before loading
- ☑ Display ULP Advice ™ notification dialog prior to debug launch
- ☑ Display warning dialog when "Sync group cores" action is selected
- □ Sort groups by name in Debug view
- □ Activate Debug view when starting a debug session

Reload the program automatically if the program file changes

○ Always ○ Never ● Prompt

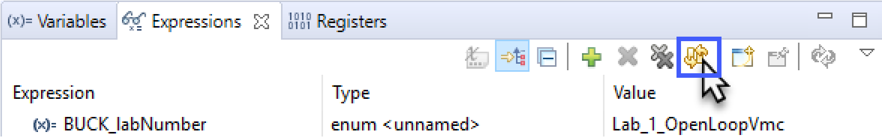
Continuous refresh interval (milliseconds) 1000

☑ Auto disable continuous refresh when there are severe reading errors

Shared target configuration directory [redacted]

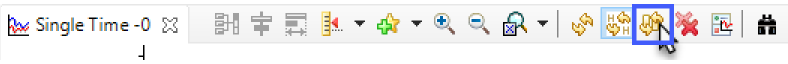
NOTE: The refresh operation introduces overhead to system debug and may degrade the reliability of the debug link if the refresh intervals are too short. 1000-ms is a safe starting value to use when experimenting with a new system. The refresh interval may be changed at the discretion of the user.

d. Click on the *Continuous Refresh* shortcut in the Expressions window to enable periodic updates of the variable values



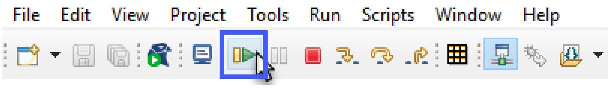
Expression	Type	Value
(x)= BUCK_labNumber	enum <unnamed>	Lab_1_OpenLoopVmc

e. Click on the *Enable Continuous Refresh* shortcut in the Single Time graph window to enable periodic updates of the graph data



4.3.6 Run the Program

1. Click on the *Resume* shortcut to run the program



2. Red light LED4 on LAUNCHXL-F280049C will blink to indicate that the program is running
3. The *BUCK_vInSensed_Volts* value in the Expressions window should be close to 0-V [6]
4. Turn on the 9-VDC power supply

[6] Noise in the 0-mV to 20-mV range is normal.

5. The *BUCK_vInSensed_Volts* value should now resemble the 9-VDC supply voltage

   NOTE: The feedback voltage from the 9-VDC supply is always present on the ADC input pin of the MCU, even when SW1 is turned off. To prevent damage to the MCU, avoid turning on the 9-VDC supply while the LAUNCHXL-F280049C is powered off.

6. Turn on toggle switch SW1 on the BOOSTXL-BUCKCONV
7. Confirm that the default 0% duty cycle state is operating correctly as Table 5 shows:

Table 5. Lab 1: Expected Values for 0% Duty Cycle

Variable	Expected Value
BUCK_labNumber	Lab_1_OpenLoopVmc
BUCK_dutySetRef_pu	0.0
BUCK_dutySet_pu	0.0
BUCK_vOutSensed_Volts	Close to 0 (less than 0.02)
BUCK_iOutSensed_Amps	Close to 0 (less than 0.06)
BUCK_iOutTripFlag	0

8. Increase *BUCK_dutySetRef_pu* to 0.3 pu:
   a. Click on the Value field of *BUCK_dutySetRef_pu* to enable editing of its value
   b. Type *0.3* into the field
   c. Press Enter to activate the new value

   NOTE: A software-based slew mechanism is used to moderate instantaneous duty cycle changes to minimize current spikes.

   NOTE: The maximum allowable duty cycle is clamped to 45% in software.

9. Confirm that the updated 30% duty cycle state is operating correctly, as Table 6 details:

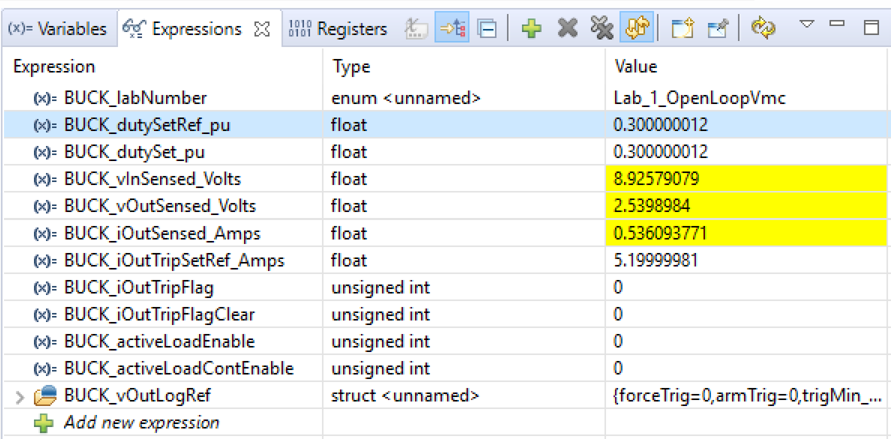
Table 6. Lab 1: Expected Values for 30% Duty Cycle

Variable	Expected Value [1]
BUCK_dutySetRef_pu	0.3
BUCK_dutySet_pu	0.3
BUCK_vOutSensed_Volts	Close to 2.5 [2]
BUCK_iOutSensed_Amps	Close to 0.5 [3]
BUCK_iOutTripFlag	0

[1] Floating-point values may differ slightly due to precision limitations.

[2] The output voltage will vary with the input voltage, and may range from 2.3 V to 3.3 V. For 9-V input voltage, the output voltage should be near 2.5 V in open loop.

[3] The output current will vary with the input voltage, and may range from 0.45 A to 0.75 A. For 9-V input voltage, the output current should be near 0.5 A in open loop.



Expression	Type	Value
BUCK_labNumber	enum <unnamed>	Lab_1_OpenLoopVmc
BUCK_dutySetRef_pu	float	0.300000012
BUCK_dutySet_pu	float	0.300000012
BUCK_vInSensed_Volts	float	8.92579079
BUCK_vOutSensed_Volts	float	2.5398984
BUCK_iOutSensed_Amps	float	0.536093771
BUCK_iOutTripSetRef_Amps	float	5.19999981
BUCK_iOutTripFlag	unsigned int	0
BUCK_iOutTripFlagClear	unsigned int	0
BUCK_activeLoadEnable	unsigned int	0
BUCK_activeLoadContEnable	unsigned int	0
BUCK_vOutLogRef	struct <unnamed>	{forceTrig=0,armTrig=0,trigMin_...
Add new expression		

Figure 7. Expressions Window for 0.3 pu Duty Cycle in Lab 1

4.3.7 Test the Overcurrent Trip Protection

The overcurrent trip protection is triggered whenever the instantaneous inductor current exceeds the trip threshold that is set using *BUCK_iOutTripSetRef_Amps*. Trip conditions are detected using an analog CMPSS comparator, which then signals the EPWM to immediately drive the PWM-HI control signal low.

To avoid false trips from switching noise, the example uses filtering hardware from both the CMPSS and EPWM modules. The CMPSS digital filter is enabled so that the trip condition must be present for a minimum number of MCU clock cycles before a trip signal is generated to the EPWM. The EPWM blanking filter is configured to ignore CMPSS trip signals near the beginning of the PWM cycle when the inductor current is expected to spike momentarily.

1. Reduce the *BUCK_iOutTripSetRef_Amps* trip level value to 0.2 A
2. Confirm that the updated 0.2-A trip level triggered the overcurrent protection state as Table 7 shows:

Table 7. Lab 1: Expected Values in Tripped State

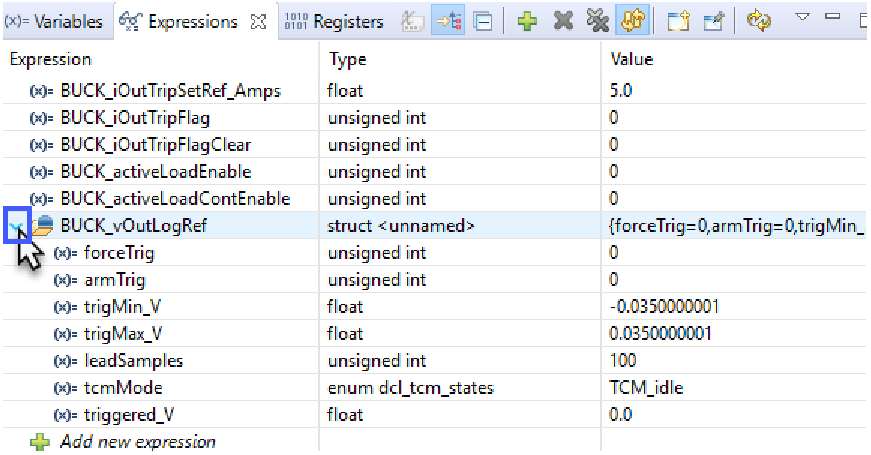
Variable	Expected Value
BUCK_vOutSensed_Volts	Close to 0.0
BUCK_iOutTripFlag	1

3. Clear the overcurrent trip condition:
   a. Reduce *BUCK_dutySetRef_pu* to 0.0 pu
   b. Increase *BUCK_iOutTripSetRef_Amps* to 5 A
   c. Write 1 to *BUCK_iOutTripFlagClear* (the value will self-clear back to 0)
   d. Confirm that *BUCK_iOutTripFlag* has been cleared back to 0
4. Increase *BUCK_dutySetRef_pu* back to 0.3 pu
5. Confirm that the system is operating as expected (same as Table 6)

4.3.8 Graph the Output Voltage

The program includes a 400-element array (*BUCK_vOutLog_Volts*) that is used for capturing a sequential series of output voltage ADC samples. The Transient Capture Module (TCM) framework from the DCL is used to control the data logging behavior.

1. Expand the *BUCK_vOutLogRef* structure in the Expressions window to access the member variables:

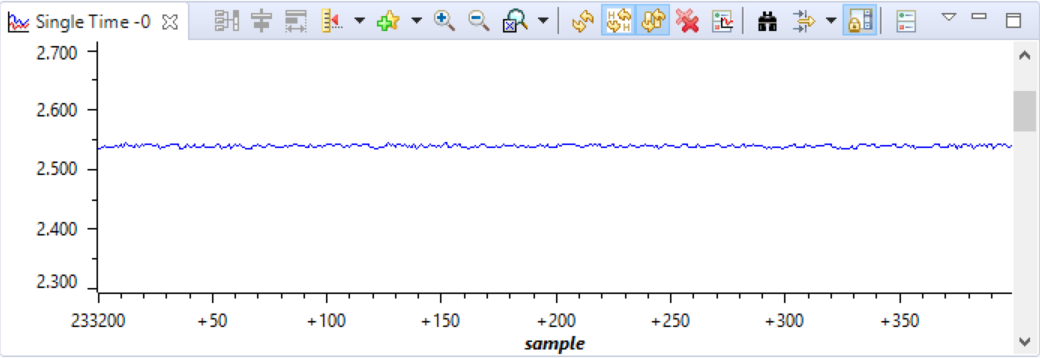


Expression	Type	Value
BUCK_iOutTripSetRef_Amps	float	5.0
BUCK_iOutTripFlag	unsigned int	0
BUCK_iOutTripFlagClear	unsigned int	0
BUCK_activeLoadEnable	unsigned int	0
BUCK_activeLoadContEnable	unsigned int	0
BUCK_vOutLogRef	struct <unnamed>	{forceTrig=0,armTrig=0,trigMin_
forceTrig	unsigned int	0
armTrig	unsigned int	0
trigMin_V	float	-0.0350000001
trigMax_V	float	0.0350000001
leadSamples	unsigned int	100
tcmMode	enum dcl_tcm_states	TCM_idle
triggered_V	float	0.0
Add new expression		

Table 8. BUCK_vOutLogRef Member Variables

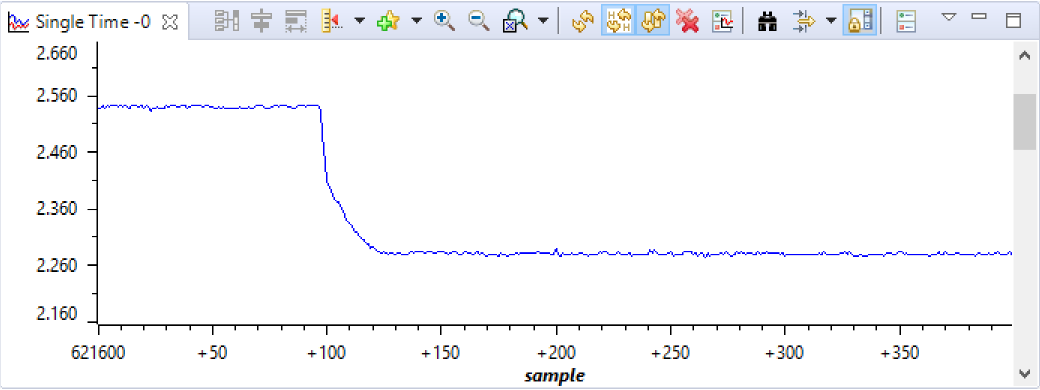
Variable	User Access	Description
forceTrig	Write	Immediately trigger a data series capture Write 0 = No effect Write 1 = Immediately capture a series of data (variable will self-clear to 0)
armTrig	Write	Arm capture trigger Write 0 = No effect Write 1 = Capture a series of data when the output voltage is either less than *trigMin_V* or greater than *trigMax_V* (variable will self-clear to 0)
trigMin_V	Read and Write	Lower-bound threshold of capture trigger
trigMax_V	Read and Write	Upper-bound threshold of capture trigger
leadSamples	Read and Write	Number of samples to capture before either trigger threshold is exceeded
tcmMode	Read	TCM capture state
triggered_V	Read	Value of the first output voltage sample that exceeded a trigger threshold

2. Write 1 to *BUCK_vOutLogRef.forceTrig* (the value will self-clear back to 0)
3. Observe the capture results:
   - *BUCK_vOutLogRef.tcmMode* will update to *TCM_complete* when the capture is complete
   - The Single Time graph display will update to show the captured output voltage, where each sample is collected in the ISR (5-us period)



NOTE: Zoom-in on the graph by using the left-click selection on the label of either axis to determine the viewing area. Use the *Reset Zoom* shortcut in the Single Time graph window to return to the default zoom level.

4. Capture the buck converter response to a transient load:
   a. Change *BUCK_vOutLogRef.trigMin_V* to be 100 mV below the *BUCK_vOutLogRef.triggered_V* voltage
   b. Change *BUCK_vOutLogRef.trigMax_V* to be 100 mV above the *BUCK_vOutLogRef.triggered_V* voltage
   c. Write 1 to *BUCK_vOutLogRef.armTrig*
   d. *BUCK_vOutLogRef.tcmMode* will update to *TCM_armed* to indicate that the TCM logger is ready to capture data
   e. Set *BUCK_activeLoadEnable* to 1 to introduce a periodic load to the buck converter output
   f. *BUCK_vOutLogRef.tcmMode* will update to *TCM_complete* when the capture is complete
   g. The Single Time graph window will update with the transient load response data



NOTE: If the TCM does not trigger, try to adjust the *BUCK_vOutLogRef.trigMin_V* trigger level:

1. Clear *BUCK_activeLoadEnable* to 0
2. Change *BUCK_vOutLogRef.trigMin_V* to a value closer to *BUCK_vOutLogRef.triggered_V*
3. Write 1 to *BUCK_vOutLogRef.armTrig*
4. Set *BUCK_activeLoadEnable* to 1

4.3.9 Run SFRA

SFRA can sweep and capture the frequency response of the buck converter without the use of external instrumentation. Frequency injection and response measurements take place in the control ISR. Background data transfers between the MCU and host computer is supported across the virtual serial (COM) port that is embedded in the XDS110 debug probe on the LAUNCHXL-F280049C.

1. Enable the continuous active load mode to ensure that the buck converter operates in continuous conduction mode:
   - Set *BUCK_activeLoadEnable* to 1
   - Set *BUCK_activeLoadContEnable* to 1
   - Confirm the variable values as Table 9 shows:

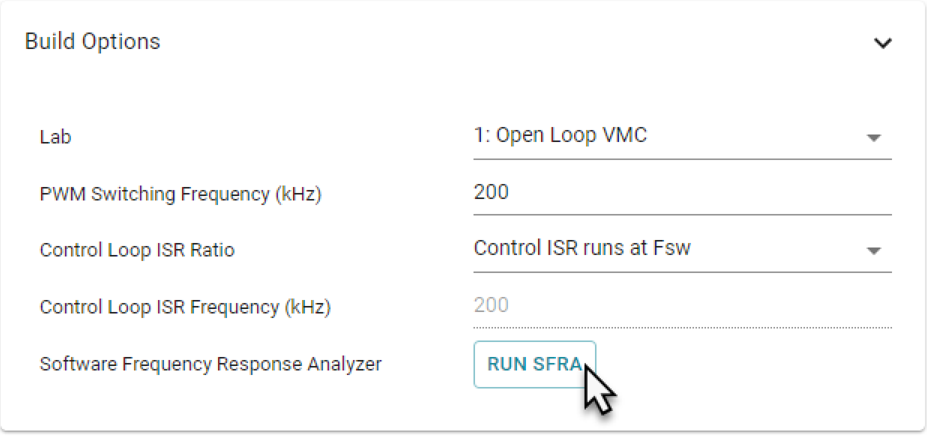
Table 9. Lab 1: Expected Values for SFRA Measurement

Variable	Expected Value
BUCK_labNumber	Lab_1_OpenLoopVmc
BUCK_dutySetRef_pu	0.3
BUCK_dutySet_pu	0.3
BUCK_vOutSensed_Volts	Close to 2.2 [1]
BUCK_iOutSensed_Amps	Close to 1.5 [2]
BUCK_iOutTripFlag	0
BUCK_activeLoadEnable	1
BUCK_activeLoadContEnable	1

[1] The output voltage will vary with the input voltage, and may range from 2.1 V to 3.0 V. For 9-V input voltage, the output voltage should be near 2.2 V in open loop.

[2] The output current will vary with the input voltage, and may range from 1.3 A to 2.1 A. For 9-V input voltage, the output current should be near 1.5 A in open loop.

2. Return to *main.syscfg* either by clicking on the open tab or by double-clicking on the file in the Project Explorer window
3. Click the *RUN SFRA* button to launch the SFRA GUI window



Build Options

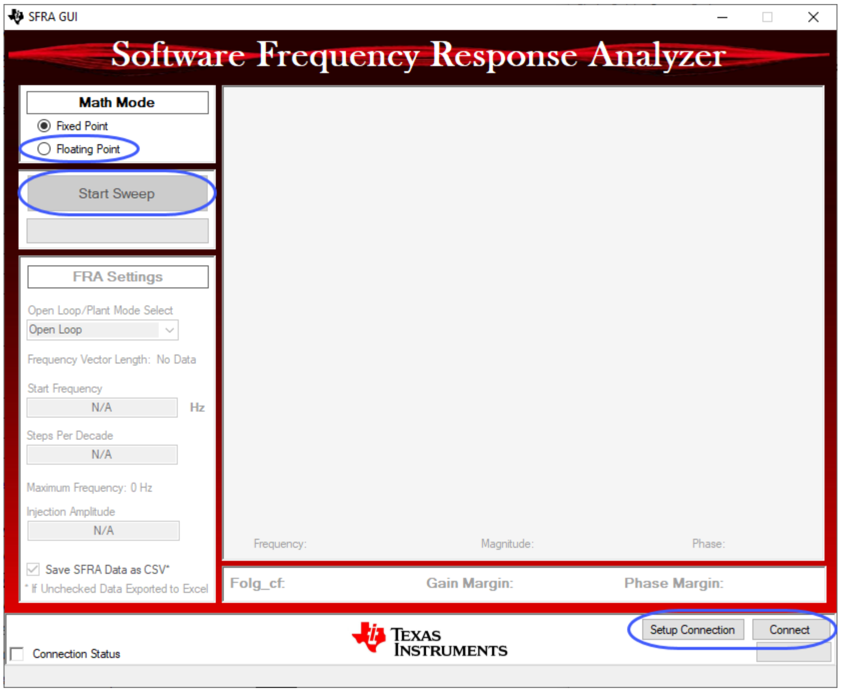
Lab: 1: Open Loop VMC

PWM Switching Frequency (kHz): 200

Control Loop ISR Ratio: Control ISR runs at Fsw

Control Loop ISR Frequency (kHz): 200

Software Frequency Response Analyzer



● Fixed Point
○ Floating Point

Open Loop/Plant Mode Select: Open Loop

Frequency Vector Length: No Data

Start Frequency: N/A Hz

Steps Per Decade: N/A

Maximum Frequency: 0 Hz

Injection Amplitude: N/A

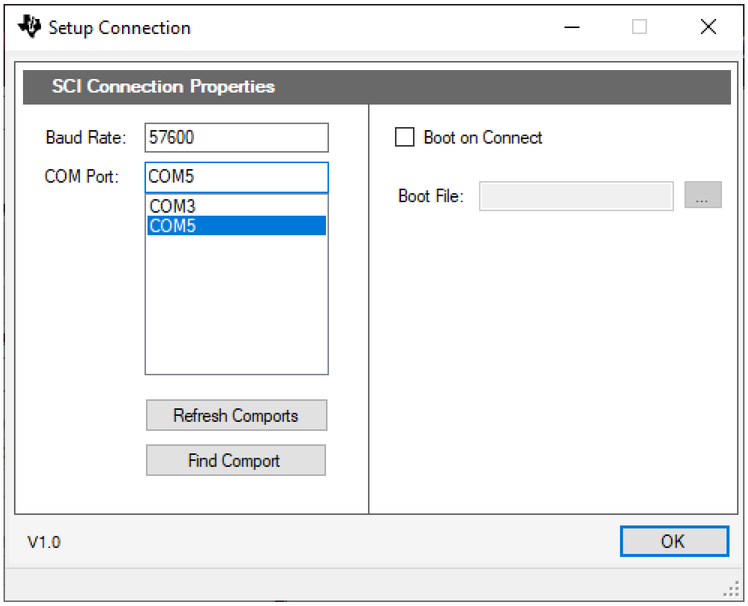
☑ Save SFRA Data as CSV*

* If Unchecked Data Exported to Excel

Folg_cf: Gain Margin: Phase Margin:

□ Connection Status

4. In SFRA, select the *Floating Point* Math Mode
5. Click the *Setup Connection* button



Baud Rate: 57600

COM Port: COM5

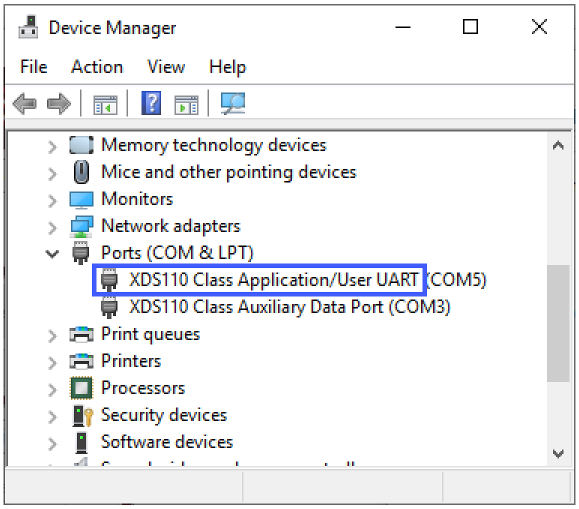
COM3
COM5

□ Boot on Connect

Boot File:

6. Uncheck the *Boot on Connect* checkbox
7. Set the Baud Rate to 57600
8. Select the LAUNCHXL-F280049C user COM port

NOTE: The COM port device description is *XDS110 Class Application/User UART*.

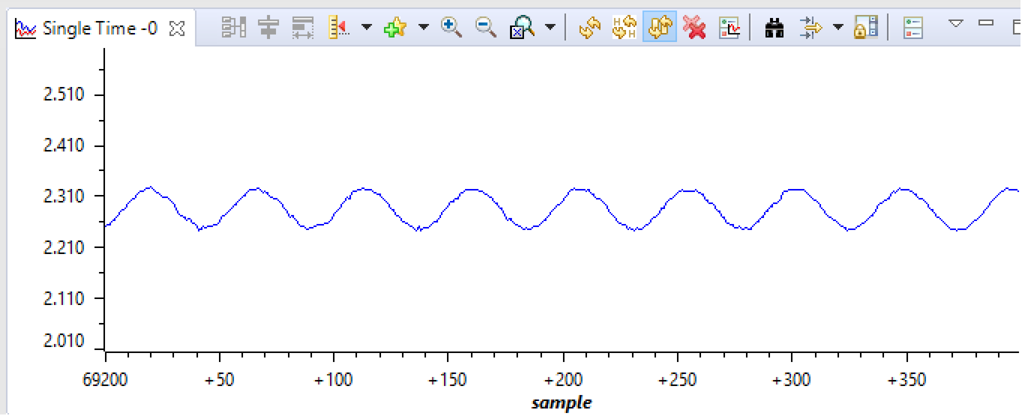


Device Manager

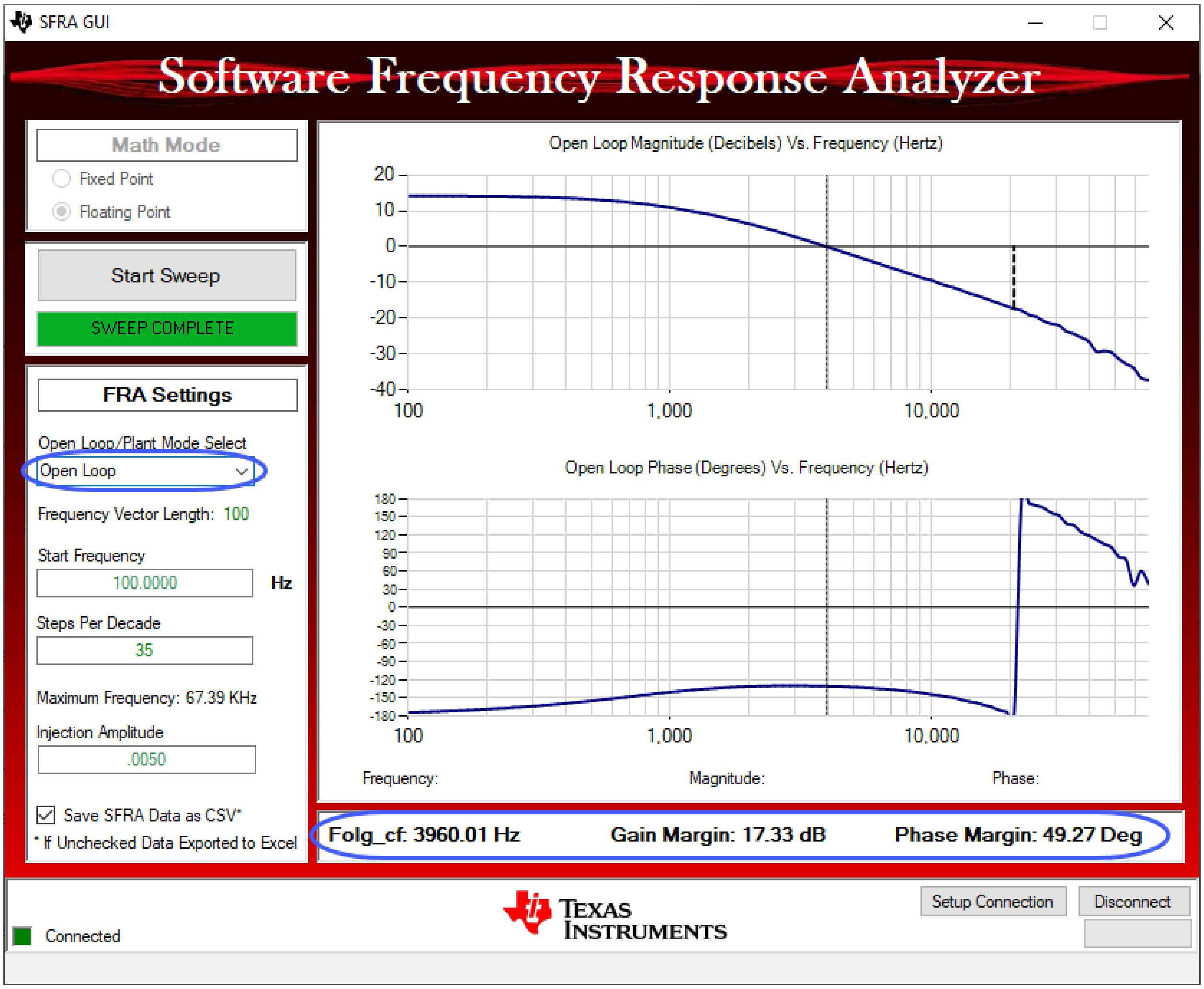
File Action View Help

- Memory technology devices
- Mice and other pointing devices
- Monitors
- Network adapters
- Ports (COM & LPT)
  - XDS110 Class Application/User UART (COM5)
  - XDS110 Class Auxiliary Data Port (COM3)
- Print queues
- Printers
- Processors
- Security devices
- Software devices

9. Click the *OK* button
10. Click the *Connect* button
11. Click the *Start Sweep* button:
    - The progress bar below the *Start Sweep* button will show the data collection status
    - Green light LED5 on LAUNCHXL-F280049C will blink when the COM port is active
    - SFRA will inject small signals on top of the PWM duty cycle during the sweep



12. The SFRA chart will update when the sweep is complete



Open Loop/Plant Mode Select: Open Loop

Frequency Vector Length: 100

Start Frequency: 100.0000 Hz

Steps Per Decade: 35

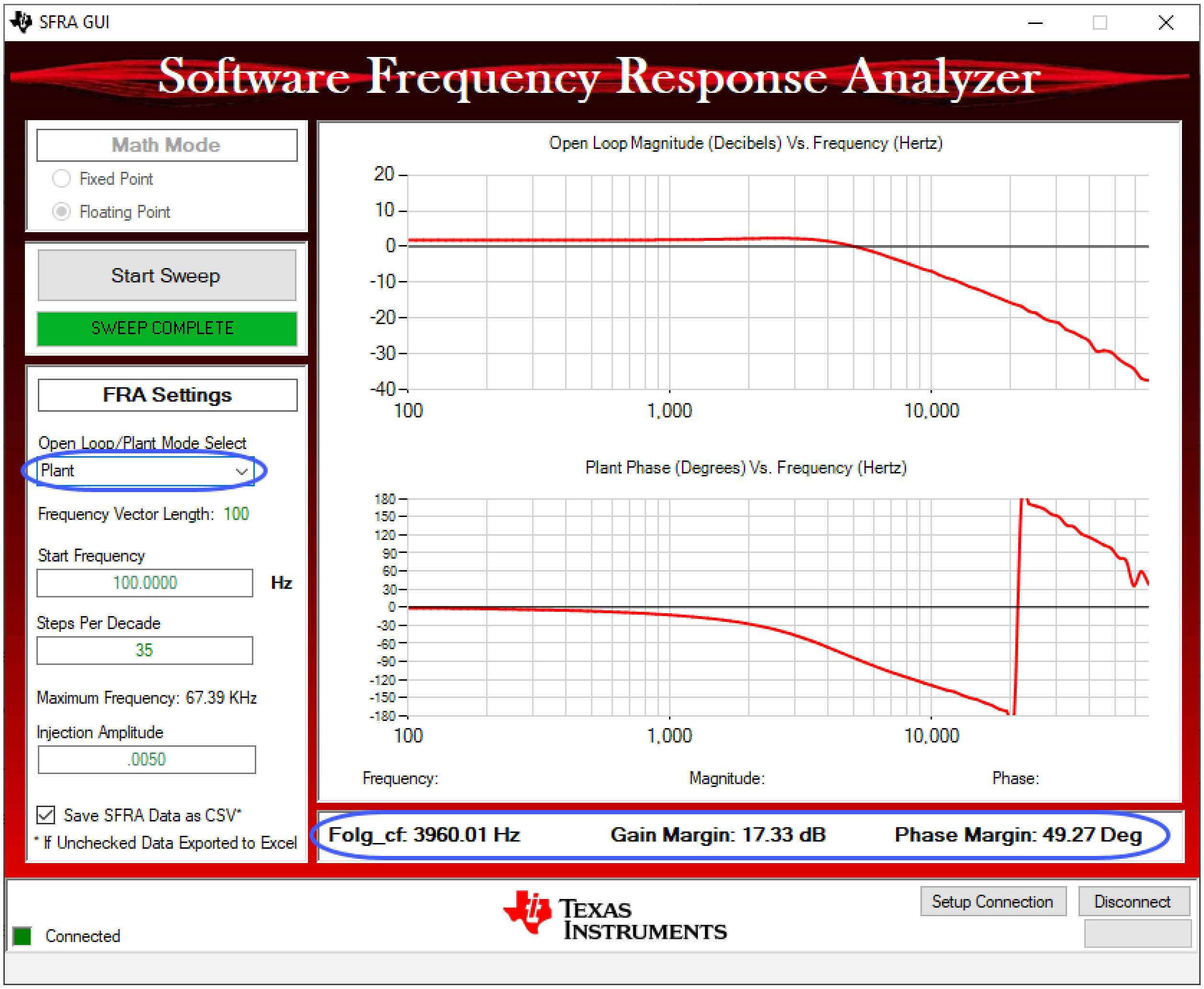
Maximum Frequency: 67.39 KHz

Injection Amplitude: .0050

☑ Save SFRA Data as CSV*

* If Unchecked Data Exported to Excel

Folg_cf: 3960.01 Hz Gain Margin: 17.33 dB Phase Margin: 49.27 Deg



Math Mode

○ Fixed Point

● Floating Point

Start Sweep

SWEEP COMPLETE

FRA Settings

Open Loop/Plant Mode Select: Plant

Frequency Vector Length: 100

Start Frequency: 100.0000 Hz

Steps Per Decade: 35

Maximum Frequency: 67.39 KHz

Injection Amplitude: .0050

☑ Save SFRA Data as CSV*

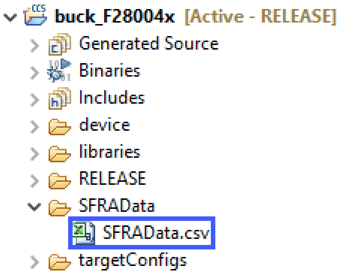
* If Unchecked Data Exported to Excel

Folg_cf: 3960.01 Hz Gain Margin: 17.33 dB Phase Margin: 49.27 Deg

Connected

Setup Connection Disconnect

13. Record the results under the SFRA chart for comparison with the compensated VMC results in Lab 2 (Section 5.3.6)
14. The raw SFRA data is saved to the project workspace for offline inspection and for use with Compensation Designer in Lab 2 (Section 5.3.2)



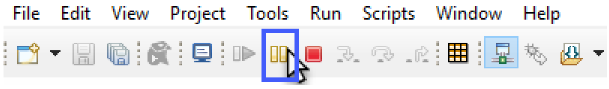
```
buck_F28004x [Active - RELEASE]
    Generated Source
    Binaries
    Includes
    device
    libraries
    RELEASE
    SFRAData
        SFRAData.csv
    targetConfigs
```

15. Click the *Disconnect* button
16. Close the SFRA GUI window
17. Clear *BUCK_activeLoadEnable* to 0
18. Clear *BUCK_activeLoadContEnable* to 0

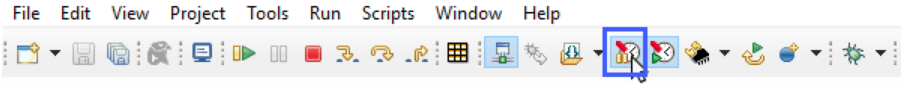
4.3.10 Halt the MCU

Put the system into a safe state and halt the MCU.

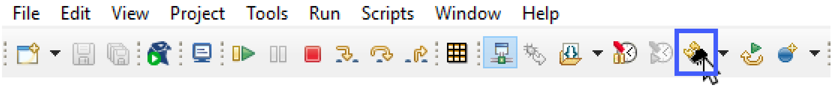
1. Reduce *BUCK_dutySetRef_pu* to 0.0 pu
2. Turn off toggle switch SW1 on the BOOSTXL-BUCKCONV
3. Turn off the 9-VDC power supply
4. Click the *Suspend* shortcut to halt the processor



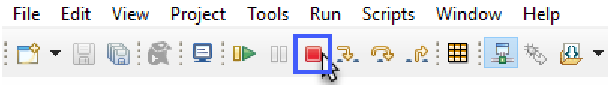
5. Disable Real-time Mode:
   a. Click the *Silicon Real-time Mode* shortcut to toggle off Real-time Mode



   b. Click the *CPU Reset* shortcut to clear pending activity



6. Continue to the next lab, or disconnect LAUNCHXL-F280049C with the following steps:
   a. Click the *Terminate* shortcut to disconnect the debug connection



   b. Close CCS
   c. Disconnect the LAUNCHXL-F280049C micro-USB cable to power off the board

5 Lab 2: Closed-Loop Control with VMC

5.1 *Objective*

The objective of this lab is to demonstrate the regulation of the buck converter output voltage using a VMC feedback control-loop that is implemented in software. Compensation Designer will be used to configure the performance of the control loop, and SFRA will be used to validate the results at run-time. An active-load circuit that is controlled by the software will provide a switched load to test the transient performance of the system. A method for fine-tuning at run-time will also be used.

The control loop in this lab was manually tuned to meet generic stability and performance targets for demonstration purposes. Custom tuning can be achieved using the same tools and procedures described in this lab.

NOTE: Lab 2 should only be attempted after the Lab 1 operational checks have been completed (Section 4.3.6 and Section 4.3.7).

5.2 *Overview*

Lab 2 uses the same code base as Lab 1 except for the following build-time changes:

- Lab 2 responds to user requests for a regulated output voltage level using *BUCK_vOutSetRef_Volts* instead of a PWM duty-cycle using *BUCK_dutySetRef_pu*
- Lab 2 introduces a compensated voltage-loop to dynamically adjust the PWM duty-cycle to follow the requested output voltage

The software has been configured to provide closed-loop VMC control for the buck power stage. A 2-pole 2-zero DCL compensator (DF22) implements the control law for the control loop. The DF22 compensator accepts the output voltage error as its input and returns the compensated control effort.

The output voltage error (ek) is calculated as the difference between the user-requested output voltage versus the ADC-sensed output voltage.

The control effort (uk) is interpreted as a per-unit duty cycle that is converted to a proportionate EPWM[CMPA] value to realize the approximate duty cycle for driving the FET switches on the buck converter.

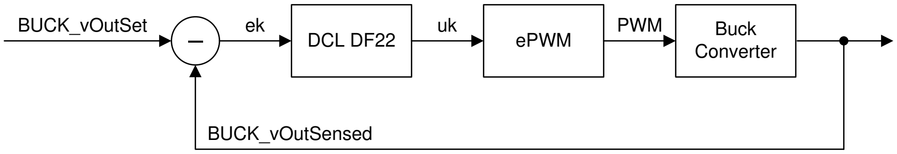


Figure 8. VMC System Diagram

5.3 *Procedure*

5.3.1 Prepare the Hardware and Software

Repeat the hardware and software setup procedures from Lab 1 (Section 4.3.1 and Section 4.3.2).

5.3.2 Run Compensation Designer

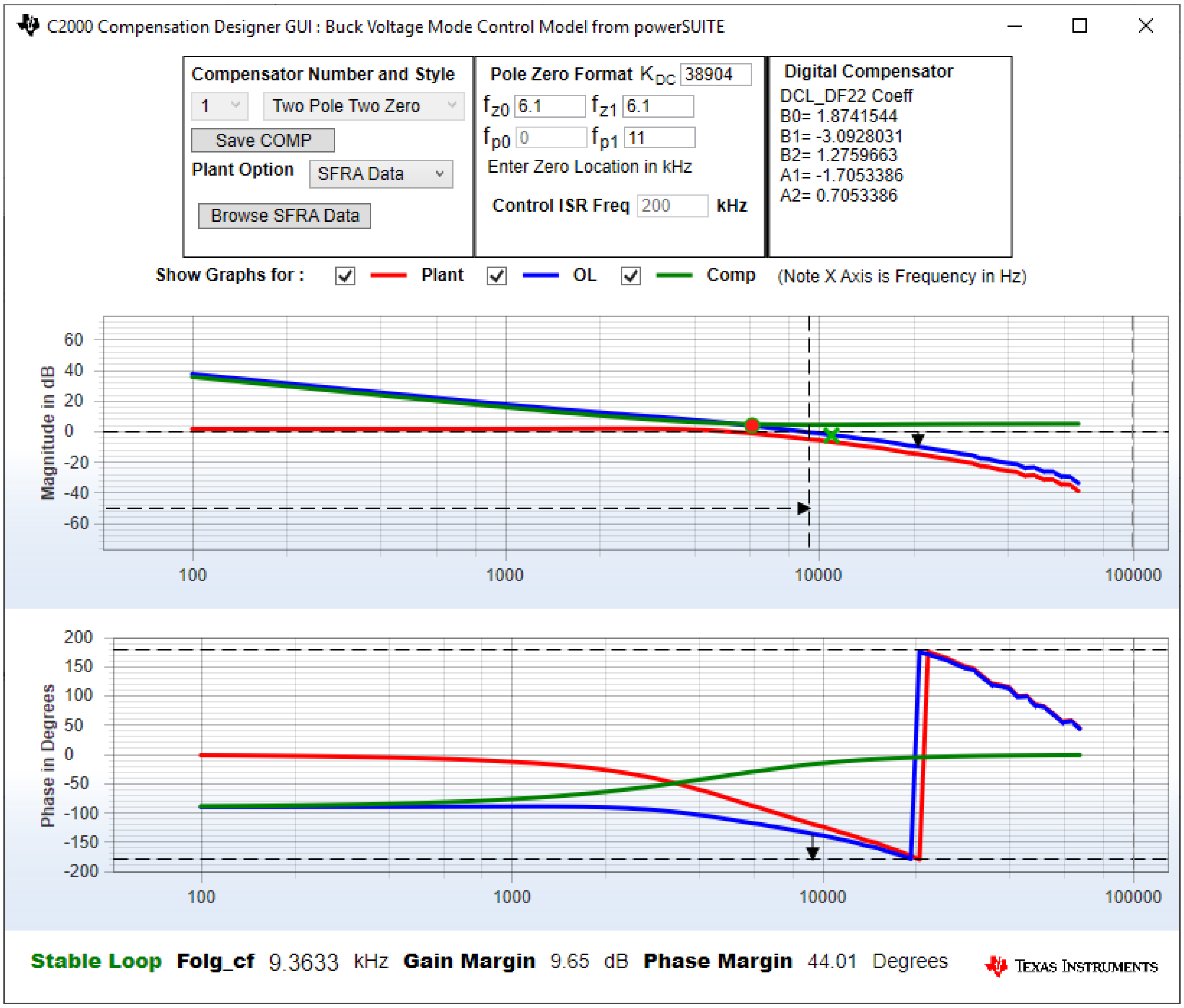
1. In *main.syscfg*, configure the project settings as Table 10 lists:

Table 10. Lab 2 Project Settings

Parameter	Value
Lab	2: Closed Loop VMC
PWM Switching Frequency (kHz)	200
Control Loop ISR Ratio	Control ISR runs at Fsw
Plant	BOOSTXL-BUCKCONV
LaunchPad Site	Site 2 [1]

[1] Select *Site 1* if using headers J1-J4 instead of J5-J8.

2. Click *RUN COMPENSATION DESIGNER* to launch the Compensation Designer GUI
3. In Compensation Designer, load the SFRA data from Lab 1 (Section 4.3.9):



C2000 Compensation Designer GUI : Buck Voltage Mode Control Model from powerSUITE

Compensator Number and Style: 1, Two Pole Two Zero

Plant Option: SFRA Data

Pole Zero Format K_{DC} 38904

f_{z0} 6.1 f_{z1} 6.1

f_{p0} 0 f_{p1} 11

Enter Zero Location in kHz

Control ISR Freq 200 kHz

Digital Compensator
DCL_DF22 Coeff
B0= 1.8741544
B1= -3.0928031
B2= 1.2759663
A1= -1.7053386
A2= 0.7053386

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

Stable Loop Folg_cf 9.3633 kHz Gain Margin 9.65 dB Phase Margin 44.01 Degrees

a. Click *Browse SFRA Data*

b. Select *SFRAData.csv* from the project workspace

NOTE: The path of SFRAData.csv can be found by inspecting its file properties in the Project Explorer tree (buck_F28004x » SFRAData » SFRAData.csv).

c. Click *Open*

d. Change the Plant Option from *Modelled* to *SFRA Data*

4. Fill in the Pole Zero parameter values as Table 11 shows:

Table 11. Lab 2 Compensation Designer Initial Pole Zero Values

Parameter	Value
K_{DC}	38904
f_{z0}	6.1
f_{z1}	6.1
f_{p1}	11

NOTE: Compensation Designer will not act upon typed values until the user presses the *Enter* key in the input field.

5. Confirm that the Digital Compensator results match the values in Table 12:

Table 12. Lab 2 Compensation Designer Initial Coefficient Values

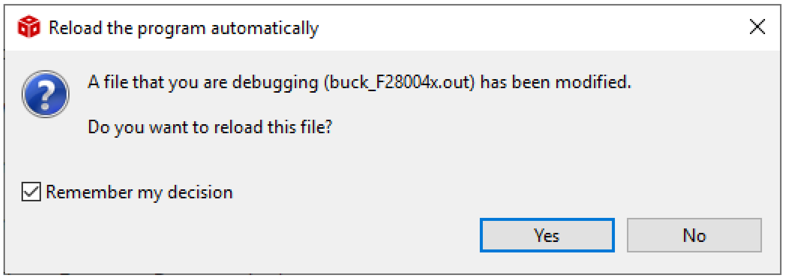
Coefficient	Value
B0	1.8741544
B1	-3.0928031
B2	1.2759663
A1	-1.7053386
A2	0.7053386

6. Review the analysis at the bottom of the Compensation Designer window:
   - Confirm that Compensation Designer reports *Stable Loop*
   - Record the estimated crossover frequency, gain margin, and phase margin for comparison with the SFRA measurement results (to be taken in Section 5.3.6)
   - **Optional**: Uncheck the *Plant* and *Comp* graphs for a clearer view of the *OL* behavior
7. Reduce the K_{DC} gain value to 4000 and observe the changes in the loop performance:
   - Decrease in crossover frequency
   - Increase in gain margin and phase margin
8. Experiment with additional pole zero values to see how the loop behavior changes
9. When ready, restore the pole zero values to match those from Table 11
10. Click *Save COMP* to export the compensator settings to Solution Adapter
11. Close the Compensation Designer window

5.3.3 Build and Load Lab 2

NOTE: If needed, review Lab 1 (Section 4.3.3) for the detailed steps to build and load projects with CCS.

1. Confirm that the *main.syscfg* selections still match the project settings from Table 10
2. Rebuild the project
3. Load the program
   - If starting from a new CCS session, use the *Debug* shortcut to launch the target configuration and load the program
   - If continuing uninterrupted from Lab 1, CCS may prompt to reload the program automatically after rebuilding the project



Reload the program automatically

A file that you are debugging (buck_F28004x.out) has been modified.

Do you want to reload this file?

☑ Remember my decision

   a. Check *Remember my decision*
   b. Click *Yes*
   c. CCS will automatically load the program after this and each subsequent rebuild

5.3.4 Prepare the Debug Environment

NOTE: If needed, review Lab 1 (Section 4.3.4 and Section 4.3.4) for the detailed steps to prepare the debug windows and enable the Real-time Debug mode.

1. Run the *setupdebugenv_lab2.js* script
2. Add a Single Time graph using the *graph1.graphProp* properties
3. Enable Continuous Refresh for the Expressions window and Single Time graph
4. Enable Real-time Debug mode

Table 13. Lab 2 Expressions Window Variables

Variable	User Access	Description
BUCK_labNumber	Read	Lab build that is running on the MCU
BUCK_vOutSetRef_Volts	Read and Write	Output voltage requested by the user
BUCK_dutySet_pu	Read	PWM duty cycle used for programming the ePWM
BUCK_vInSensed_Volts	Read	Input voltage sensed from JP1
BUCK_vOutSensed_Volts	Read	Output voltage sensed from the plant
BUCK_iOutSensed_Amps	Read	Low-pass filtered (average) inductor current
BUCK_iOutTripSetRef_Amps	Read and Write	Overcurrent trip (shut-down) threshold for monitoring the unfiltered inductor current
BUCK_iOutTripFlag	Read	Overcurrent trip status 0 = Normal Operation 1 = Tripped
BUCK_iOutTripFlagClear	Write	Request to clear overcurrent trip status Write 0 = No effect Write 1 = Clear trip condition (variable will self-clear to 0)
BUCK_activeLoadEnable	Read and Write	Enable the active load 0 = Switched resistor isolated 1 = Switched resistor connected as determined by BUCK_activeLoadContEnable
BUCK_activeLoadContEnable	Read and Write	Enable continuous load (when BUCK_activeLoadEnable = 1) 0 = Switched resistor is connected and disconnected periodically 1 = Switched resistor is connected continuously
BUCK_vOutLogRef	Read and Write	Structure to configure sampling of the output voltage for graphing
BUCK_ctrlRef	Read and Write	Structure to configure the DF22 compensator coefficients

5.3.5 Run the Program

1. Turn on the 9-VDC supply
2. Turn on SW1 (BOOSTXL-BUCKCONV)
3. Click the *Resume* shortcut to run the program
4. Confirm that the default 2-V output state is operating correctly as Table 14 shows:

Table 14. Lab 2: Expected Values for 2-V Output

Variable	Expected Value
BUCK_labNumber	Lab_2_ClosedLoopVmc
BUCK_vOutSetRef_Volts	2.0
BUCK_dutySet_pu	Close to 0.24 [(1)]
BUCK_vOutSensed_Volts	Close to 2.0
BUCK_iOutTripFlag	0

[(1)] The duty cycle will vary with the input voltage, and may range from 0.27 pu to 0.18 pu. For 9-V input voltage, the duty cycle should be near 0.24 pu in open loop.

NOTE: If the program is allowed to run before the 9-VDC supply and SW1 are turned on, the system may enter into a tripped state (*BUCK_iOutTripFlag* = 1). This happens because the control loop will attempt to regulate the output voltage while the input voltage is off, which results in a very large PWM duty cycle that will trigger the overcurrent protection when the supply is eventually turned on. To clear the fault condition, perform these steps:

1. Reduce *BUCK_vOutSetRef_Volts* to 0.0 V
2. Write 1 to *BUCK_iOutTripFlagClear*
3. Confirm that *BUCK_iOutTripFlag* self-clears to 0
4. Restore *BUCK_vOutSetRef_Volts* back to 2.0 V

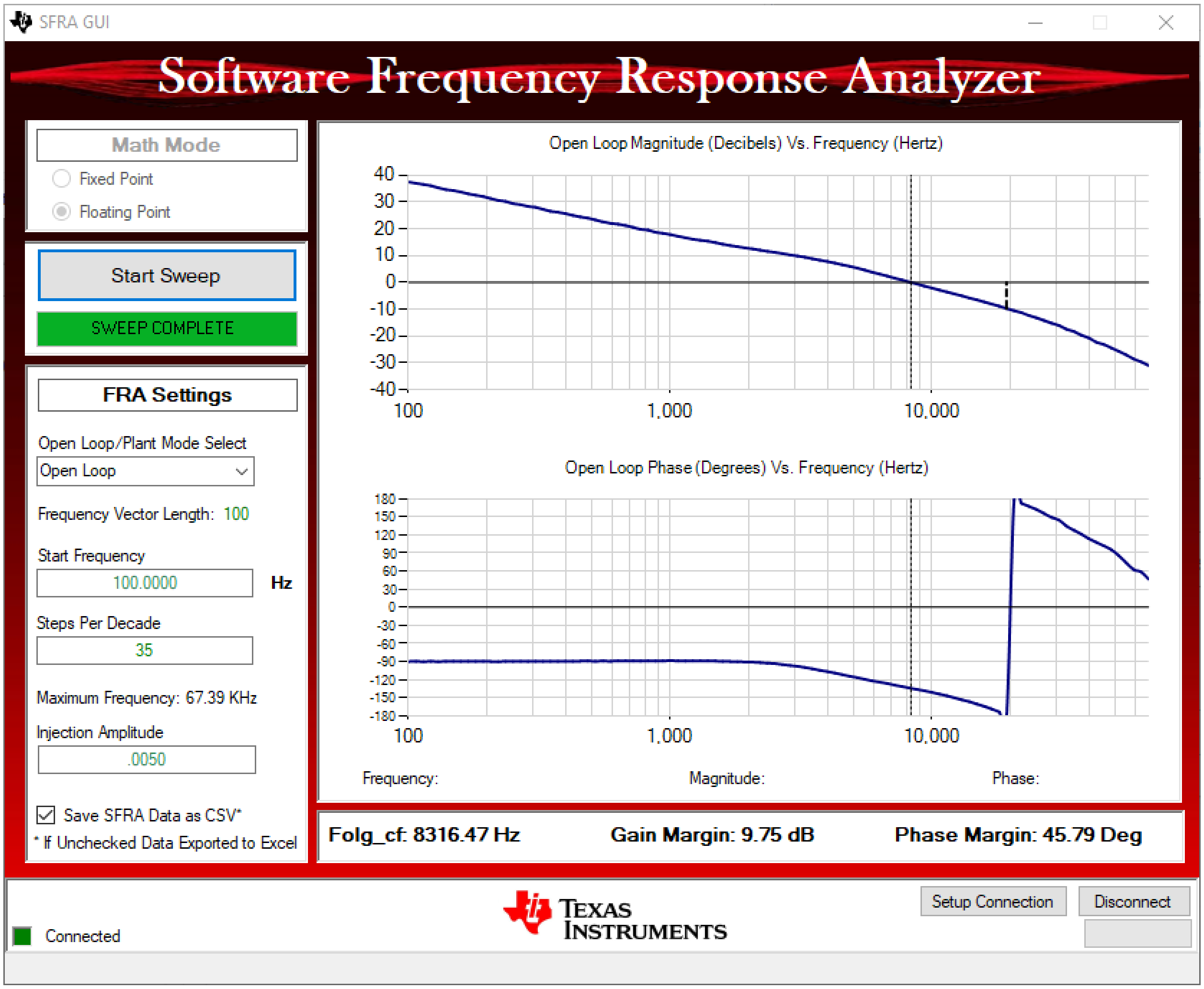
5. Reduce *BUCK_vOutSetRef_Volts* to 1.0 V
6. Observe the changes in *BUCK_dutySet_pu* and *BUCK_vOutSensed_Volts* as the system seeks the new voltage set point
7. Restore *BUCK_vOutSetRef_Volts* back to 2.0 V

5.3.6 Run SFRA

NOTE: If needed, review Lab 1 (Section 4.3.9) for the detailed steps to run a frequency sweep with SFRA.

1. Confirm that the system is still in the state matching Table 14
2. Set *BUCK_activeLoadEnable* to 1
3. Set *BUCK_activeLoadContEnable* to 1
4. In *main.syscfg*, click *RUN SFRA*
5. Perform a frequency sweep with the same SFRA GUI configuration used in Lab 1:
   - Math Mode = Floating Point
   - Boot on Connect = Disabled
   - Baud Rate = 57600
   - COM Port = Selected

6. Review the results:



SFRA GUI

Software Frequency Response Analyzer

Math Mode

○ Fixed Point

● Floating Point

Start Sweep

SWEEP COMPLETE

FRA Settings

Open Loop/Plant Mode Select: Open Loop

Frequency Vector Length: 100

Start Frequency: 100.0000 Hz

Steps Per Decade: 35

Maximum Frequency: 67.39 KHz

Injection Amplitude: .0050

☑ Save SFRA Data as CSV*

* If Unchecked Data Exported to Excel

Folg_cf: 8316.47 Hz

Gain Margin: 9.75 dB

Phase Margin: 45.79 Deg

🟩 Connected

- The crossover frequency, gain margin, and phase margin values should be similar to the values estimated by Compensation Designer (Section 5.3.2)
- Compare the performance to the open-loop measurements from Lab 1 (Section 4.3.9)
- Record the results for comparison with a modified compensator (Section 5.3.8) and PCMC in Lab 4 (Section 7.3.4)

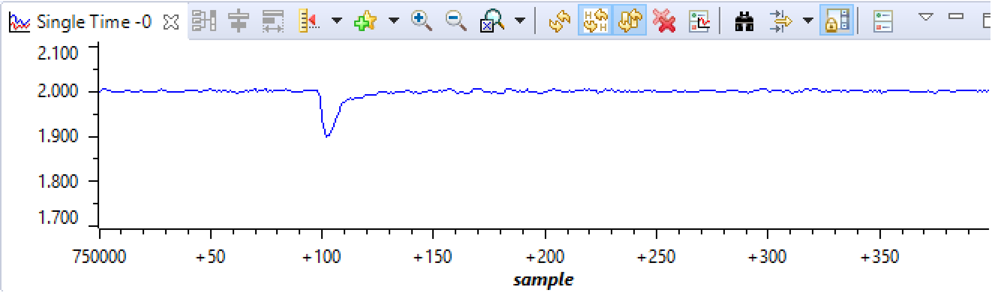
7. Click *Disconnect*
8. Close the SFRA window
9. Clear *BUCK_activeLoadEnable* to 0
10. Clear *BUCK_activeLoadContEnable* to 0

5.3.7 Graph the Load Transient Response

NOTE: If needed, review Lab 1 (Section 4.3.8) for the detailed steps to graph the output voltage.

1. Expand the *BUCK_vOutLogRef* structure and set:
   - *BUCK_vOutLogRef.trigMin_V* to 1.965 V
   - *BUCK_vOutLogRef.trigMax_V* to 2.035 V
2. Write 1 to *BUCK_vOutLogRef.armTrig*

3. Set *BUCK_activeLoadEnable* to 1



4. Record the voltage drop and recovery time shown in the graph for comparison with a modified compensator (Section 5.3.8)
5. Clear *BUCK_activeLoadEnable* to 0

5.3.8 Modify the DCL Compensator

Compensation Designer is a useful tool for tuning the control loop because of its visual feedback and ability to export the compensator coefficients directly into the project. However, the design flow of running Compensation Designer and rebuilding the project for each trial of coefficients may be cumbersome when the loop needs to be fine-tuned manually with iterative steps.

This example supports run-time tuning of the compensator by using the DCL safe parameter update functions to change the active coefficients in RAM. In a custom system, the fine-tuned coefficients can be built into the project either through manual software edits, or by porting the Pole Zero run-time parameters through the Compensation Designer tool flow. The following steps use the Pole Zero format to describe the compensator behavior, but any other format that is supported by DCL can be used instead.

1. Expand the *BUCK_ctrlRef* structure to access the member variables shown in Table 15:

Table 15. BUCK_ctrlRef Member Variables

Variable	User Access	Description

ctrlFormat	Read and Write	Select the compensator update format
		Format_PID = Update using the PID format (*ctrlPid*)
		Format_ZPK = Update using the Pole Zero format (*ctrlZpk*)
		Format_Coeff = Update the coefficients directly (*ctrlCoeff*)

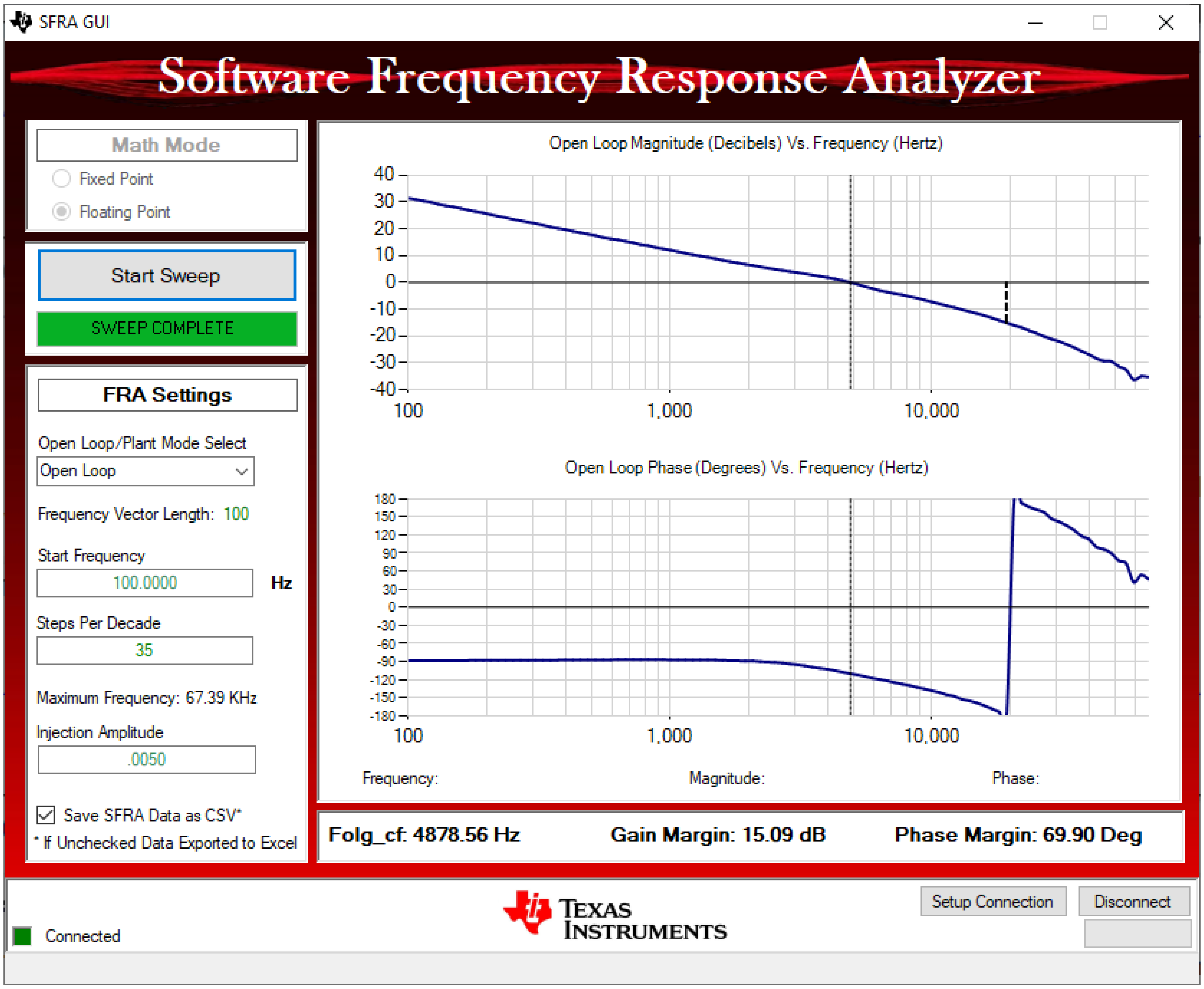
ctrlUpdate	Write	Perform a DCL safe parameter update using the format selected with *ctrlFormat* Write 0 = No effect Write 1 = Update the coefficients (variable will self-clear to 0)

ctrlPid	Read and Write	PID format update structure
		Scale = Multiplication factor applied to the *Kp*, *Ki*, and *Kd* variables to improve user readability
		Kp = Proportional gain (effective value is *Kp* / *Scale*)
		Ki = Integral gain (effective value is *Ki* / *Scale*)
		Kd = Derivative gain (effective value is *Kd* / *Scale*)
		fc_Hz = Cut-off frequency

ctrlZpk	Read and Write	Pole Zero format update structure
		Kdc = Gain
		Z0_kHz = f_{z0}
		Z1_kHz = f_{z1}
		P1_kHz = f_{p1}

ctrlCoeff	Read and Write	Coefficient format update structure B0, B1, B2, A1, A2
ctrlActiveCoeff	Read	Coefficients that are actively used by the compensator B0, B1, B2, A1, A2

2. Change the *BUCK_ctrlRef.ctrlFormat* value to *Format_ZPK*
3. Reduce the Pole Zero gain:
   a. Expand *BUCK_ctrlRef.ctrlZpk*
   b. Confirm that the default values are the same as those imported from Compensation Designer (Section 5.3.2)
   c. Reduce *BUCK_ctrlRef.ctrlZpk.Kdc* to 4000 [7]
4. Update the compensator
   a. Expand *BUCK_ctrlRef.ctrlActiveCoeff*
   b. Confirm that the active coefficients are the same as those imported from Compensation Designer (Section 5.3.2)
   c. Write 1 to *BUCK_ctrlRef.ctrlUpdate* (the value will self-clear back to 0)
   d. Confirm that the active coefficient values are now changed
5. Repeat the SFRA procedure from Section 5.3.6 to measure the modified frequency response
6. Review the SFRA results:



SWEEP COMPLETE

Open Loop/Plant Mode Select: Open Loop

Frequency Vector Length: 100

Start Frequency: 100.0000 Hz

Steps Per Decade: 35

Maximum Frequency: 67.39 KHz

Injection Amplitude: .0050

☑ Save SFRA Data as CSV*

* If Unchecked Data Exported to Excel

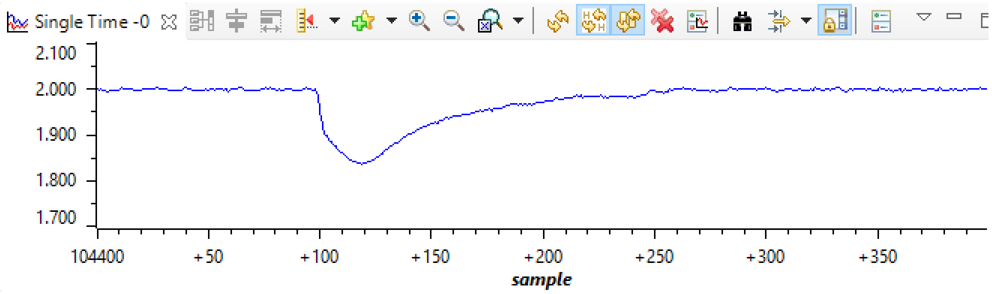
Folg_cf: 4878.56 Hz   Gain Margin: 15.09 dB   Phase Margin: 69.90 Deg

🟩 Connected

- The crossover frequency is expected to be significantly lower
- The gain margin and phase margin are expected to be significantly higher

[7] Kdc = 4000 is based on the initial starting values from Table 11. If custom values were used, select a Kdc value that is much smaller than the initial value.

7. Repeat the load transient response graphing procedure from Section 5.3.7
8. Compare the initial response to this modified response



9. **Optional** - Rebuild the program to use the modified coefficients:
   a. Record the active coefficients for comparison in Step 9c
   b. Follow the procedure in Section 5.3.2 to rebuild the project with Kdc = 4000
   c. Confirm that the coefficients from Step 9a are now built into the program
   d. Follow the procedure in Section 5.3.6 to measure the updated frequency response

5.3.9 Halt the MCU

NOTE: If needed, review Lab 1 (Section 4.3.10) for the detailed steps to disable Real-time Mode and disconnect the LAUNCHXL-F280049C.

1. Reduce *BUCK_vOutSetRef_Volts* to 0.0 V
2. Turn off SW1 (BOOSTXL-BUCKCONV)
3. Turn off the 9-VDC supply
4. Click the *Suspend* shortcut
5. Disable Real-time Mode
6. Continue to the next lab or disconnect the LAUNCHXL-F280049C

6 Lab 3: Open-Loop Check for PCMC

6.1 Objective

The objective of this lab is to verify the operation of the PCMC hardware using an open voltage-loop system. The user will be introduced to the concept of indirectly influencing the PWM duty cycle by setting a peak current limit.

Without the ability to control the duty cycle directly, there is an increased potential for the system to encounter unfavorable boundary conditions and become unstable. Therefore, Lab 3 will have fewer exercises than the other labs.

6.2 Overview

Lab 3 is configured to accept user input from the CCS Expressions window to vary the peak ILFB current limit that applies to each PWM cycle. Each PWM cycle will start by driving PWM-HI high until the peak current threshold is reached, at which time, the PWM-HI signal will be driven low for the rest of the PWM cycle. As with VMC, the ePWM Dead-Band Generator Submodule will produce a complementary PWM-LO signal with inserted dead-band.

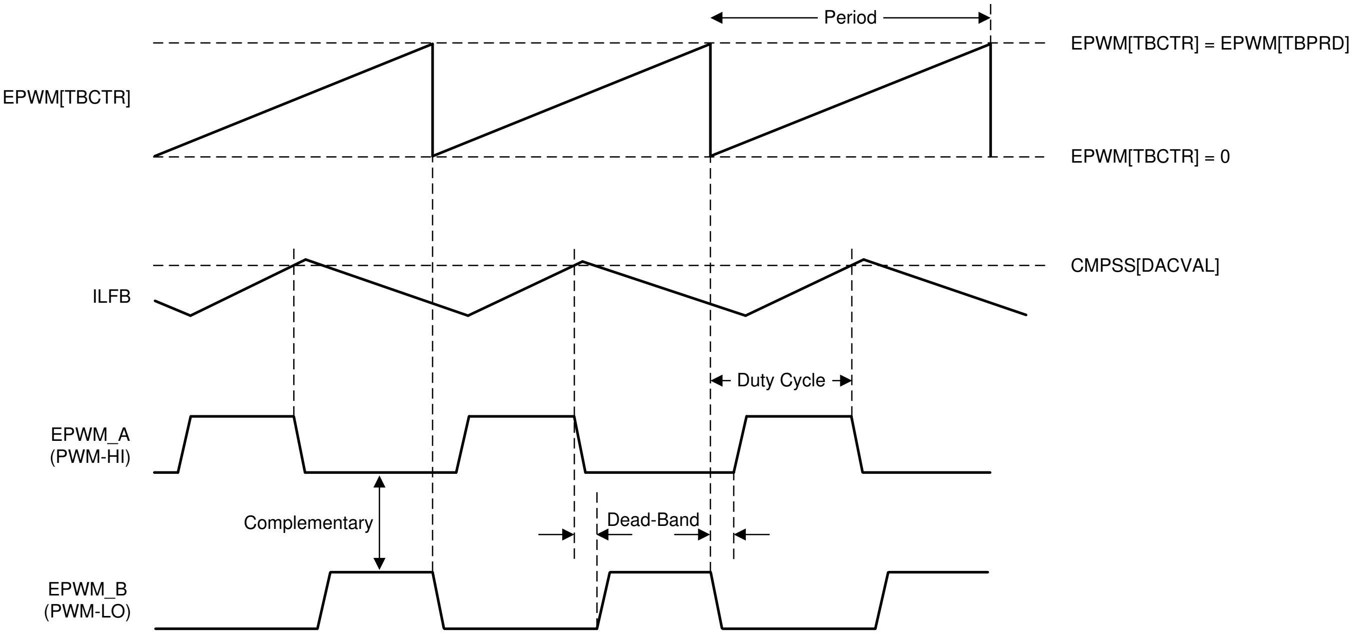


Figure 9. PWM Signal Generation for PCMC

The peak current limit is enforced using a CMPSS module with the following embedded components:

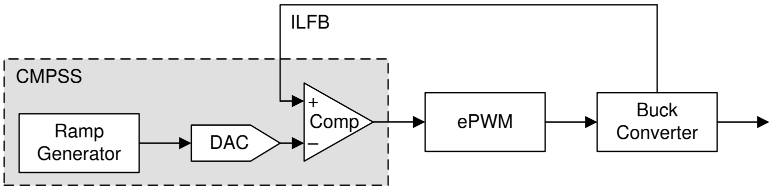


Figure 10. CMPSS Components for PCMC

- Comparator — Continuously monitors the voltages on its inputs for level crossing. For PCMC, the comparator monitors the inductor feedback current signal against the peak current threshold that is approximated by the internal reference DAC voltage. The comparator generates a trip signal to the driving ePWM when the current limit is met, which then responds by driving PWM-HI low.
- Reference DAC — Provides an internal reference voltage to the inverting input of the comparator. For PCMC, the DAC value represents the peak current threshold. The Ramp Generator is leveraged to control the DAC with value updates that are synchronized with the driving ePWM and the comparator state.
- Ramp Generator — Controls the reference DAC value using a state machine that is designed to generate a series of decreasing values for PCMC slope compensation.

Although overcurrent protection is inherent in the peak current loop, a redundant protection path is implemented in the program using spare comparator resources on the MCU.

A general overview of implementing PCMC using a digital controller is found in the *Digital Peak Current Mode Control With Slope Compensation Using the TMS320F2803x Application Report* . [8]

6.3 Procedure

6.3.1 Prepare the Hardware and Software

Repeat the hardware and software setup procedures from Lab 1 (Section 4.3.1 and Section 4.3.2).

6.3.2 Build and Load Lab 3

NOTE: If needed, review Lab 1 (Section 4.3.3) for the detailed steps to build and load projects with CCS.

1. In *main.syscfg*, configure the project settings as Table 16 shows:

Table 16. Lab 3 Project Settings

Parameter	Value
Lab	3: Open Loop PCMC
PWM Switching Frequency (kHz)	200
Control Loop ISR Ratio	Control ISR runs at Fsw
Slope Compensation (16b RAMPDECVAL)	7
Plant	BOOSTXL-BUCKCONV
LaunchPad Site	Site 2 [1]

[1] Select *Site 1* if using headers J1-J4 instead of J5-J8.

2. Rebuild the project
3. Load the program

[8] TMS320F28004x devices include hardware enhancements that simplify the implementation of PCMC systems when compared to TMS320F2803x.

6.3.3 Prepare the Debug Environment

NOTE: If needed, review Lab 1 (Section 4.3.4 and Section 4.3.4) for the detailed steps to prepare the debug windows and enable the Real-time Debug mode.

1. Run the *setupdebugenv_lab3.js* script
2. Enable Continuous Refresh for the Expressions window
3. Enable Real-time Debug mode

Table 17. Lab 3 Expressions Window Variables

Variable	User Access	Description
BUCK_labNumber	Read	Lab build that is running on the MCU
BUCK_dacSetRef_pu	Read and Write	Reference DAC value requested by the user
BUCK_dacSet_pu	Read	Reference DAC value used for setting the peak current threshold
BUCK_vInSensed_Volts	Read	Input voltage sensed from JP1
BUCK_vOutSensed_Volts	Read	Output voltage sensed from the plant
BUCK_iOutSensed_Amps	Read	Low-pass filtered (average) inductor current
BUCK_iOutTripSetRef_Amps	Read and Write	Overcurrent trip (shut-down) threshold for monitoring the unfiltered inductor current
BUCK_iOutTripFlag	Read	Overcurrent trip status 0 = Normal Operation 1 = Tripped
BUCK_iOutTripFlagClear	Write	Request to clear overcurrent trip status Write 0 = No effect Write 1 = Clear trip condition (variable will self-clear to 0)
BUCK_activeLoadEnable	Read and Write	Enable the active load 0 = Switched resistor isolated 1 = Switched resistor connected as determined by BUCK_activeLoadContEnable
BUCK_activeLoadContEnable	Read and Write	Enable continuous load (when BUCK_activeLoadEnable = 1) 0 = Switched resistor is connected and disconnected periodically 1 = Switched resistor is connected continuously
BUCK_vOutLogRef	Read and Write	Structure to configure sampling of the output voltage for graphing

6.3.4 Run the Program

1. Click the *Resume* shortcut to run the program
2. Confirm that *BUCK_dacSetRef_pu* = 0.0 pu
3. Confirm that *BUCK_vOutSensed_Volts* is near 0 V
4. Turn on the 9-VDC supply
5. Confirm that *BUCK_vInSensed_Volts* resembles the 9-VDC supply voltage
6. Turn on SW1 (BOOSTXL-BUCKCONV)
7. Enable the continuous active load mode to ensure that the buck converter operates in continuous conduction mode:
   - Set *BUCK_activeLoadEnable* to 1
   - Set *BUCK_activeLoadContEnable* to 1
8. Increase *BUCK_dacSetRef_pu* to 0.1 pu
9. Confirm that *BUCK_vOutSensed_Volts* is close to 0.9 V
10. Increase *BUCK_dacSetRef_pu* to 0.3 pu
11. Confirm that *BUCK_vOutSensed_Volts* is close to 2.4 V
12. Confirm that *BUCK_iOutSensed_Amps* is close to 1.6 A

6.3.5 Halt the MCU

NOTE: If needed, review Lab 1 (Section 4.3.10) for the detailed steps to disable Real-time Mode and disconnect the LAUNCHXL-F280049C.

1. Reduce *BUCK_dacSetRef_pu* to 0.0 pu
2. Clear *BUCK_activeLoadEnable* to 0
3. Turn off SW1 (BOOSTXL-BUCKCONV)
4. Turn off the 9-VDC supply
5. Click the *Suspend* shortcut
6. Disable Real-time Mode
7. Continue to the next lab or disconnect the LAUNCHXL-F280049C

7 Lab 4: Closed-Loop Control with PCMC

7.1 Objective

The objective of this lab is to demonstrate the regulation of the buck converter output voltage using a PCMC feedback control-loop that is implemented on the LAUNCHXL-F280049C MCU. Loop tuning will rely on the run-time method that was introduced in Lab 2 (Section 5.3.8) because Compensation Designer does not support PCMC-based buck converters. SFRA and load transient response measurements will be taken for comparison against VMC.

The control loop in this lab was manually tuned to meet generic stability and performance targets for demonstration purposes. Custom tuning can be achieved using the same tools and procedures described in this lab.

7.2 Overview

Lab 4 uses the same code base as Lab 3 except for the following build-time changes:

- Lab 4 responds to user requests for a regulated output voltage level using *BUCK_vOutSetRef_Volts* instead of a current-loop limit using *BUCK_dacSetRef_pu*
- Lab 4 introduces a compensated voltage-loop to dynamically adjust the current-loop threshold to follow the requested output voltage

The PCMC solution shares the same DF22 compensator and output voltage error calculations from the VMC solution, but the control effort is converted to a proportionate CMPSS[DACVAL] value to set the current threshold for PCMC instead of a PWM duty cycle for VMC.

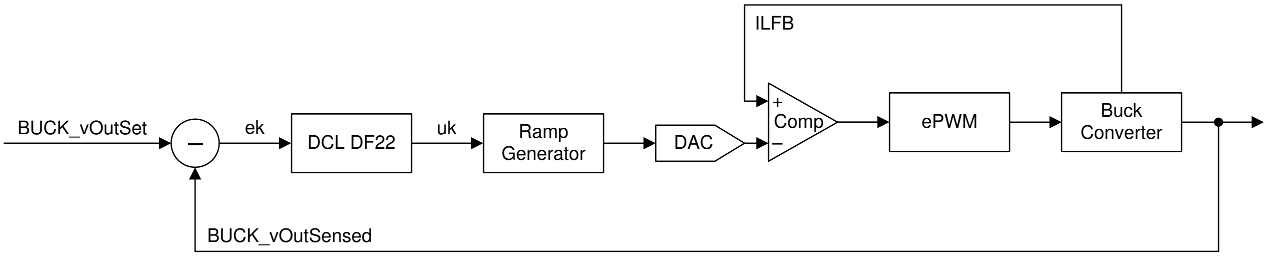


Figure 11. PCMC System Diagram

7.3 Procedure

7.3.1 Build and Load Lab 4

NOTE: If needed, review Lab 1 (Section 4.3.3) for the detailed steps to build and load projects with CCS.

1. In *main.syscfg*, configure the project settings as Table 18 shows:

Table 18. Lab 4 Project Settings

Parameter	Value
Lab	4: Closed Loop PCMC
PWM Switching Frequency (kHz)	200
Control Loop ISR Ratio	Control ISR runs at Fsw
Slope Compensation (16b RAMPDECVAL)	7
Plant	BOOSTXL-BUCKCONV
LaunchPad Site	Site 2 [1]

[1] Select *Site 1* if using headers J1-J4 instead of J5-J8.

2. Rebuild the project
3. Load the program

7.3.2 Prepare the Debug Environment

> **NOTE:** If needed, review Lab 1 (Section 4.3.4 and Section 4.3.4) for the detailed steps to prepare the debug windows and enable the Real-time Debug mode.

1. Run the *setupdebugenv_lab4.js* script
2. Add a Single Time graph using the *graph1.graphProp* properties
3. Enable Continuous Refresh for the Expressions window and Single Time graph
4. Enable Real-time Debug mode

Table 19. Lab 4 Expressions Window Variables

Variable	User Access	Description
BUCK_labNumber	Read	Lab build that is running on the MCU
BUCK_vOutSetRef_Volts	Read and Write	Output voltage requested by the user
BUCK_dacSet_pu	Read	Reference DAC value used for setting the peak current threshold
BUCK_vInSensed_Volts	Read	Input voltage sensed from JP1
BUCK_vOutSensed_Volts	Read	Output voltage sensed from the plant
BUCK_iOutSensed_Amps	Read	Low-pass filtered (average) inductor current
BUCK_iOutTripSetRef_Amps	Read and Write	Overcurrent trip (shut-down) threshold for monitoring the unfiltered inductor current
BUCK_iOutTripFlag	Read	Overcurrent trip status 0 = Normal Operation 1 = Tripped
BUCK_iOutTripFlagClear	Write	Request to clear overcurrent trip status Write 0 = No effect Write 1 = Clear trip condition (variable will self-clear to 0)
BUCK_activeLoadEnable	Read and Write	Enable the active load 0 = Switched resistor isolated 1 = Switched resistor connected as determined by BUCK_activeLoadContEnable
BUCK_activeLoadContEnable	Read and Write	Enable continuous load (when BUCK_activeLoadEnable = 1) 0 = Switched resistor is connected and disconnected periodically 1 = Switched resistor is connected continuously
BUCK_vOutLogRef	Read and Write	Structure to configure sampling of the output voltage for graphing
BUCK_ctrlRef	Read and Write	Structure to configure the DF22 compensator coefficients

7.3.3 Run the Program

1. Turn on the 9-VDC supply
2. Turn on SW1 (BOOSTXL-BUCKCONV)
3. Click the *Resume* shortcut to run the program
4. Enable the continuous active load mode to ensure that the buck converter operates in continuous conduction mode:
   - Set *BUCK_activeLoadEnable* to 1
   - Set *BUCK_activeLoadContEnable* to 1

5. Confirm that the default 2-V output state is operating correctly as Table 20 shows:

Table 20. Lab 4: Expected Values for 2-V Output

Variable	Expected Value
BUCK_labNumber	Lab_4_ClosedLoopPcmc
BUCK_vOutSetRef_Volts	2.0
BUCK_dacSet_pu	Close to 0.25
BUCK_vOutSensed_Volts	Close to 2.0
BUCK_iOutTripFlag	0
BUCK_activeLoadEnable	1
BUCK_activeLoadContEnable	1

NOTE: If the program is allowed to run before the 9-VDC supply and SW1 are turned on, the system may enter into a tripped state (*BUCK_iOutTripFlag* = 1). This happens because the control loop will attempt to regulate the output voltage while the input voltage is off, which results in a very large peak current threshold that will trigger the overcurrent protection when the supply is eventually turned on. To clear the fault condition, perform these steps:

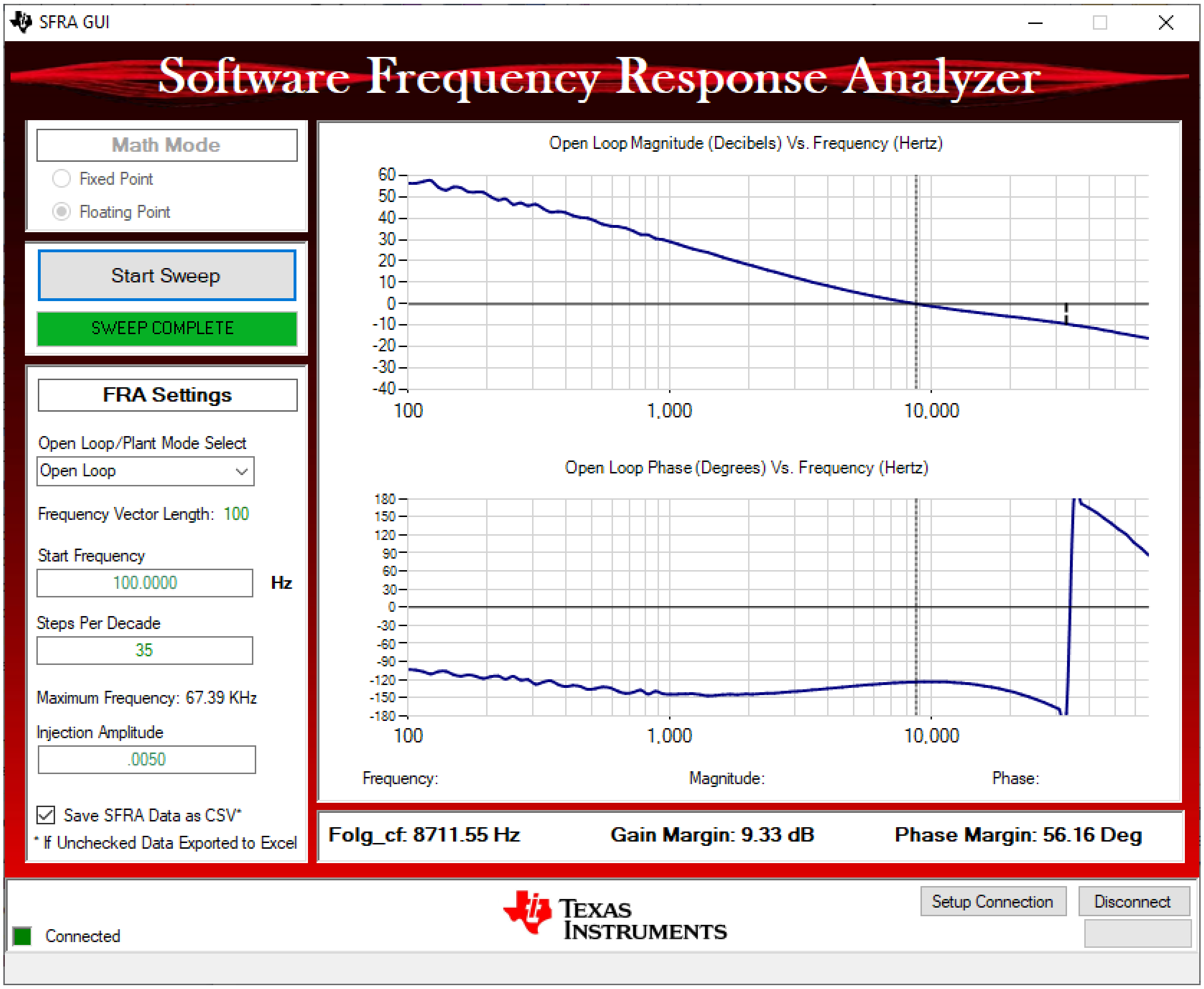
1. Reduce *BUCK_vOutSetRef_Volts* to 0.0 V
2. Write 1 to *BUCK_iOutTripFlagClear*
3. Confirm that *BUCK_iOutTripFlag* self-clears to 0
4. Restore *BUCK_vOutSetRef_Volts* back to 2.0 V

7.3.4 Run SFRA

NOTE: If needed, review Lab 1 (Section 4.3.9) for the detailed steps to run a frequency sweep with SFRA.

1. Confirm that the system is still in the state matching Table 20
2. In *main.syscfg*, click *RUN SFRA*
3. Perform a frequency sweep with the same SFRA GUI configuration used in Lab 1:
   - Math Mode = Floating Point
   - Boot on Connect = Disabled
   - Baud Rate = 57600
   - COM Port = Selected

4. Review the SFRA measurement results



Math Mode

○ Fixed Point

● Floating Point

SWEEP COMPLETE

FRA Settings

Open Loop/Plant Mode Select: Open Loop

Frequency Vector Length: 100

Start Frequency: 100.0000 Hz

Steps Per Decade: 35

Maximum Frequency: 67.39 KHz

Injection Amplitude: .0050

☑ Save SFRA Data as CSV*

* If Unchecked Data Exported to Excel

Folg_cf: 8711.55 Hz Gain Margin: 9.33 dB Phase Margin: 56.16 Deg

🟩 Connected

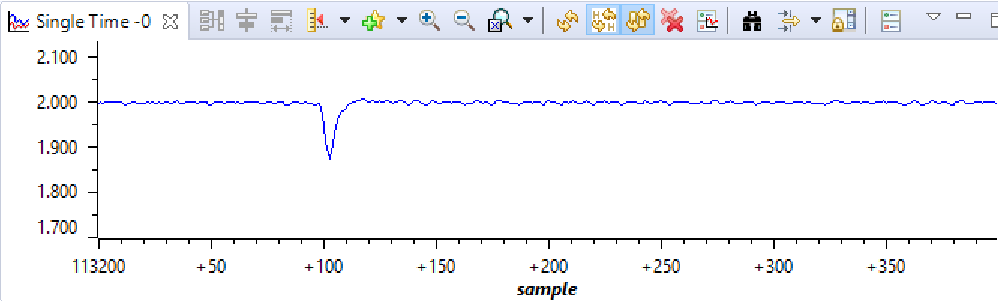
- The phase margin for PCMC is expected to be considerably better than VMC from Lab 2 (Section 5.3.6)
- The crossover frequency and gain margin are expected to be similar between PCMC and VMC
- Record the results for comparison with a modified compensator (Section 7.3.6)

7.3.5 Graph the Load Transient Response

NOTE: If needed, review Lab 1 (Section 4.3.8) for the detailed steps to graph the output voltage.

1. Expand the *BUCK_vOutLogRef* structure and set:
   - *BUCK_vOutLogRef.trigMin_V* to 1.965 V
   - *BUCK_vOutLogRef.trigMax_V* to 2.035 V
2. Clear *BUCK_activeLoadEnable* to 0
3. Write 1 to *BUCK_vOutLogRef.armTrig*

4. Set *BUCK_activeLoadEnable* to 1



5. Record the voltage drop and recovery time for comparison with a modified compensator (Section 7.3.6)

7.3.6 Modify the DCL Compensator

NOTE: If needed, review Lab 2 (Section 5.3.8) for details regarding compensator updates at run-time.

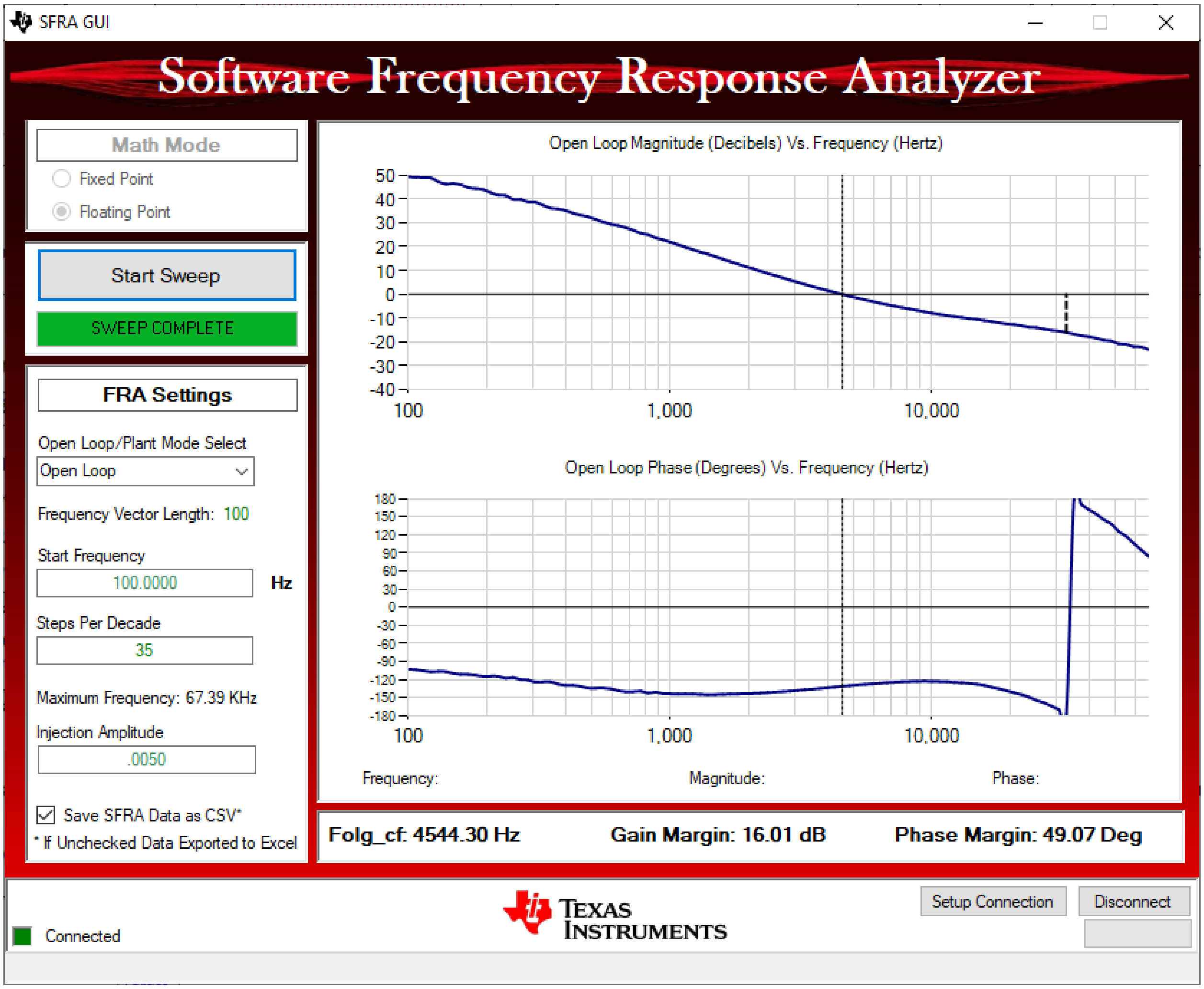
1. Expand *BUCK_ctrlRef*
2. Set *BUCK_ctrlRef.ctrlFormat* to *Format_PID*
3. Expand *ctrlPid*
4. Confirm the default PID values:

Table 21. Lab 4 Default PID Values

Parameter	Value
Scale	10.0
Kp	45.0
Ki	4.5
Kd	0.0
fc_Hz	10000.0

5. Reduce *Kp* to 20.0
6. Write 1 to *BUCK_ctrlRef.ctrlUpdate*
7. Repeat the SFRA procedure from Section 7.3.4 to measure the modified frequency response

8. Review the SFRA results:



Math Mode

○ Fixed Point

● Floating Point

SWEEP COMPLETE

FRA Settings

Open Loop/Plant Mode Select: Open Loop

Frequency Vector Length: 100

Start Frequency: 100.0000 Hz

Steps Per Decade: 35

Maximum Frequency: 67.39 KHz

Injection Amplitude: .0050

☑ Save SFRA Data as CSV*

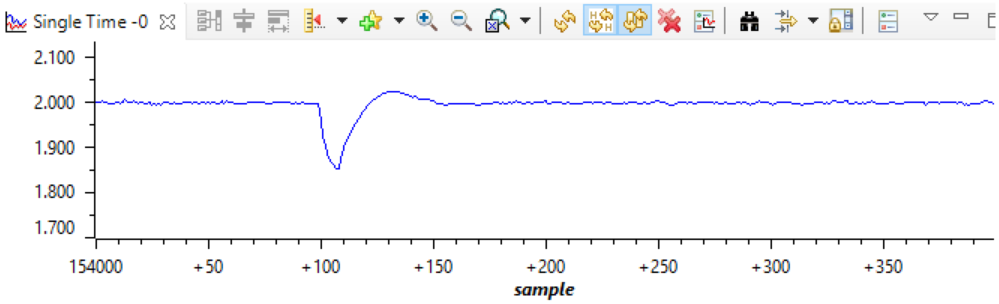
* If Unchecked Data Exported to Excel

Folg_cf: 4544.30 Hz   Gain Margin: 16.01 dB   Phase Margin: 49.07 Deg

🟩 Connected

- The crossover frequency is expected to be significantly lower
- The gain margin is expected to be significantly higher
- The phase margin is expected to vary

9. Repeat the load transient response graphing procedure from Section 7.3.5



10. Compare the initial response to this modified response

7.3.7 Halt the MCU

NOTE: If needed, review Lab 1 (Section 4.3.10) for the detailed steps to disable Real-time Mode and disconnect the LAUNCHXL-F280049C.

1. Reduce *BUCK_vOutSetRef_Volts* to 0.0 V
2. Turn off SW1 (BOOSTXL-BUCKCONV)
3. Turn off the 9-VDC supply
4. Click the *Suspend* shortcut
5. Disable Real-time Mode
6. Disconnect the LAUNCHXL-F280049C

8 Adapting This Solution

The code for this solution is organized into a framework of modular source files that is intended to minimize the effort of adapting the software to custom systems. In the hierarchy in Figure 12, the higher-level components call upon the lower-level components to execute the desired procedures.

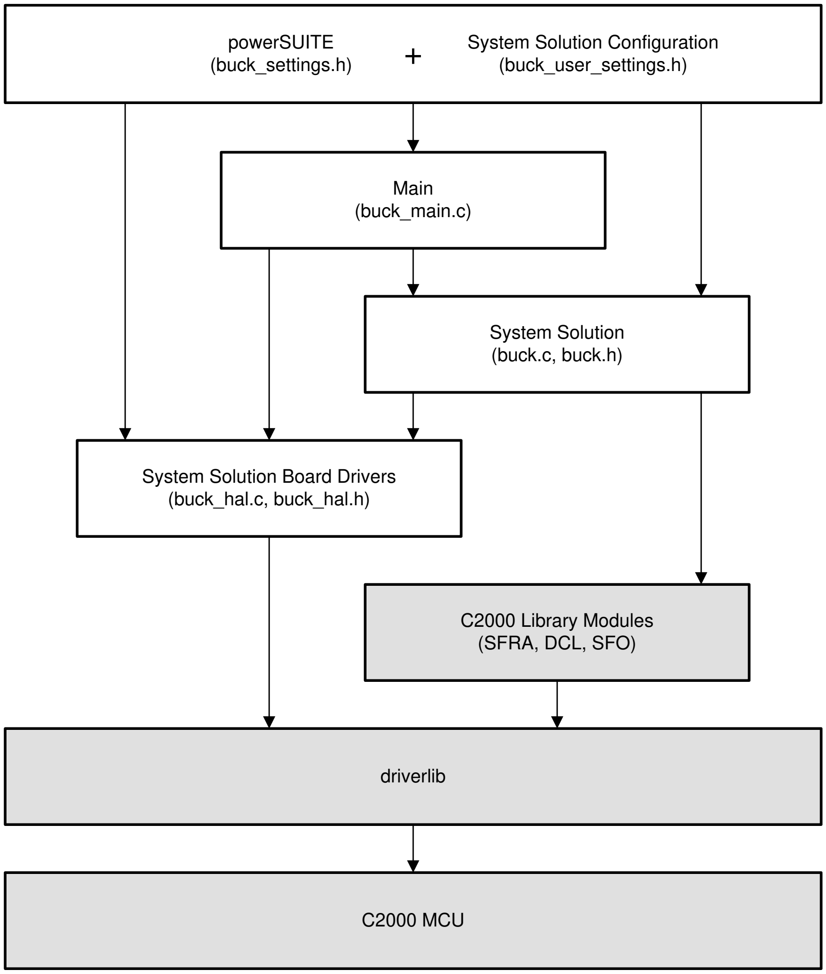


Figure 12. Software Project Hierarchy

The role of each file is described in Table 22:

Table 22. Software Project Source Files

File	Description
buck_setings.h	Defines global configuration settings for the hardware and DC-DC buck solution. For example, LaunchPad Site selection and PWM switching frequency. For powerSUITE projects, *buck_settings.h* is automatically generated at build-time based on the *main.syscfg* selections. For non-powerSUITE projects, *buck_settings.h* is treated as a normal project file.
buck_user_settings.h	Defines global configuration settings for the hardware and DC-DC buck solution. For example, ADC channel selection and ePWM dead-band duration.
buck_main.c	Makes calls to system initialization functions and schedules background tasks.
buck.c, buck.h	Defines variables and functions that are common for implementing a DC-DC buck solution. For example, DCL compensator support variables and PWM duty-cycle update functions.
buck_hal.c, buck_hal.h	Defines abstracted functions to access device-specific settings and operations. For example, ADC initialization and LED blinking.

Common configuration settings, such as pin selection, PWM frequency, and ADC conversion scaling, can often be customized through *buck_settings.h* and *buck_user_settings.h*, with no additional files changes required when running the solution on an MCU from the F28004x device family. The non-powerSUITE version of the project is recommended for such customization because the *buck_settings.h* file will be overwritten by powerSUITE at build-time.

To import the *nonpowerSUITE* version of the project into CCS, use the *CCS Edit Perspective » Project Menu » Import CCS Projects...* function.

Updates to the *buck_hal.c* and *buck_hal.h* source files may be required when modifying the solution to use different hardware capabilities, or to run on another C2000 device family. The register-level abstraction provided by *driverlib* helps significantly when porting code between devices that both have *driverlib* support.

Changes in software features, algorithm, and library usage are typically implemented in *buck_main.c*, *buck.c*, and *buck.h*.

9 Appendix

9.1 BOOSTXL-BUCKCONV Plug-in Options for LAUNCHXL-F280049C

The BOOSTXL-BUCKCONV Rev 2.1 board was originally assembled with single-sided terminal-strip connectors to interface with the LAUNCHXL-F28069M and its socket strip connectors J5–J8. When used together, the LAUNCHXL-F28069M is able to seat fully onto the BOOSTXL-BUCKCONV.

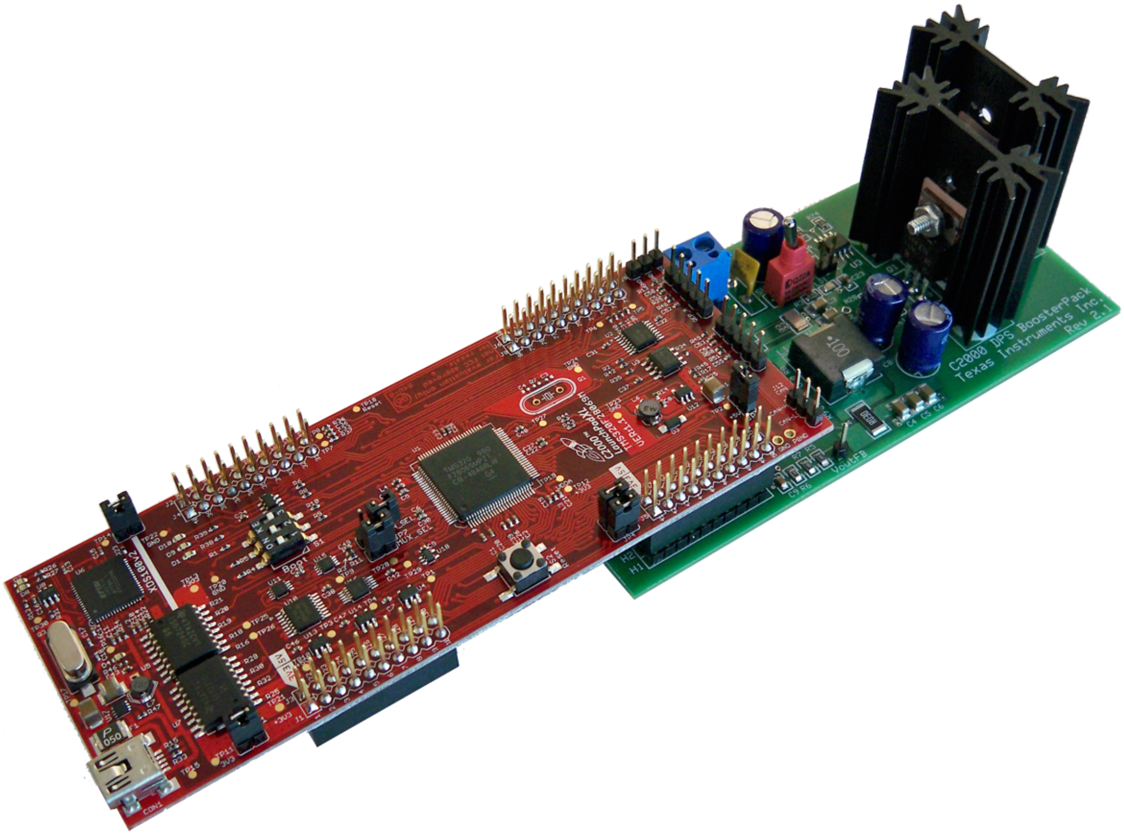


Figure 13. BOOSTXL-BUCKCONV and LAUNCHXL-F28069M

The equivalent J5–J8 socket strip connectors on the LAUNCHXL-F280049C are placed further away from the short-edge board side when compared to the LAUNCHXL-F28069M. As such, the LAUNCHXL-F280049C is not able to seat fully onto the BOOSTXL-BUCKCONV Rev 2.1 board because of mechanical interference between the LAUNCHXL-F280049C board edge and BOOSTXL-BUCKCONV JP1 terminal block. While not ideal, this partial seating is expected to suffice for running the labs in this guide.

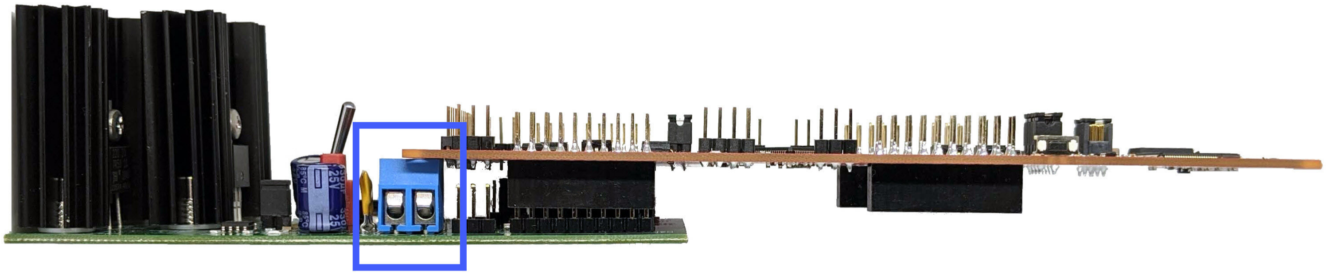


Figure 14. BOOSTXL-BUCKCONV JP1 Interference with LAUNCHXL-F280049C

The fit between the BOOSTXL-BUCKCONV Rev 2.1 and the LAUNCHXL-F280049C can be improved by press-fitting a LaunchPad style passthrough socket strip connector (such as a Samtec SSQ-110-03-T-D) between the two boards for the purpose of introducing additional clearance.

Similarly, the BOOSTXL-BUCKCONV Rev 2.1 board can be reworked to replace the terminal strip connectors with LaunchPad style socket strip connectors to enable top-side plug-in access to both Site 1 and Site 2 of the LAUNCHXL-F280049C.

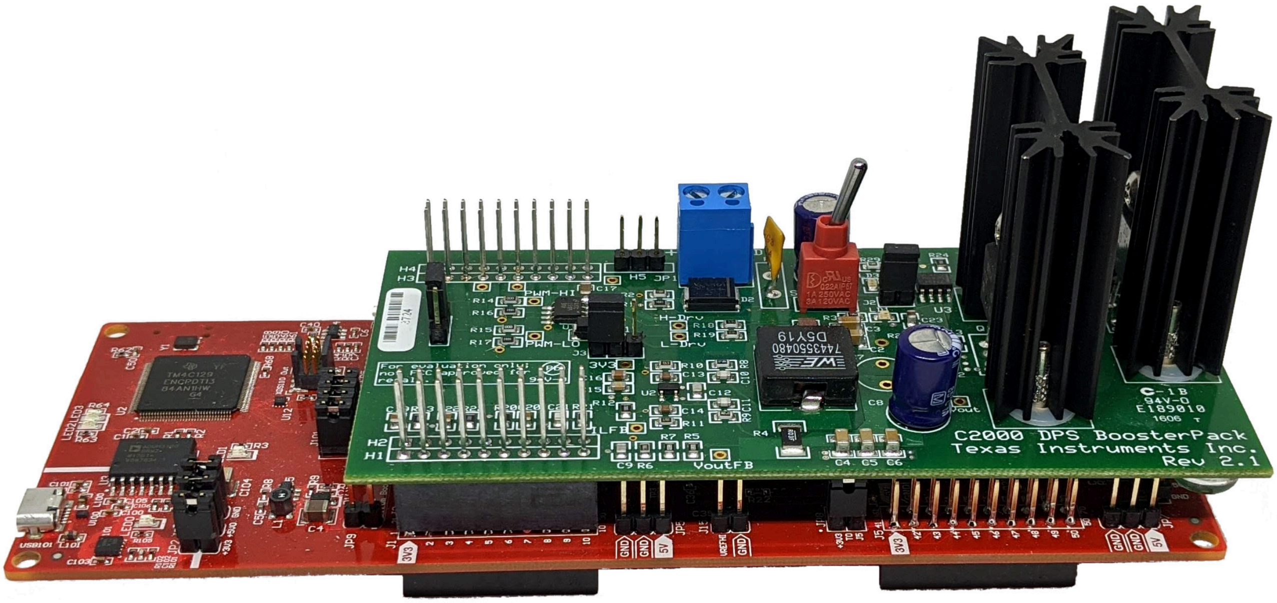


Figure 15. Modified BOOSTXL-BUCKCONV with LAUNCHXL-F280049C

Revision History

NOTE: Page numbers for previous revisions may differ from page numbers in the current version.

Changes from Original (September 2015) to A Revision	Page
• This reference design is completely updated to take advantage of newer C2000 hardware and software features.	1

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale (www.ti.com/legal/termsofsale.html) or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

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