**TI Designs**

**C2000™ Digital Power BoosterPack**

**TI Designs**

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.

**Design Resources**

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

**Design 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
- Offers an Onboard Active Load for Transient Performance Testing
- Offers Various powerSUITE Tools
  - Compensation Designer
  - Software Frequency Response Analyzer (SFRA)
  - Solution Adapter

**Featured 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.
Introduction

The C2000 Digital Power BoosterPack provides a quick and easy way to learn about digital power supply control and design using C2000 MCUs. This board consists of a DC-DC synchronous buck power stage controlled by a TMS320F28069M LaunchPad™ (LAUNCHXL-F28069M not included). The software accompanying this design at www.ti.com/controlsuite allows programming the controller and experimenting with control parameters to tune the control loop for optimal system performance. An onboard active load allows transient performance testing. This kit supports powerSUITE tools including the compensation designer, software frequency response analyzer, and solution adapter. The software project lets you evaluate the complete system with the help of these tools. This reference design shows how to connect a buck converter power stage to a C2000 digital power controller.

This guide presents how to use this board and the accompanying software with the powerSUITE tools. This guide offers a structured, step-by-step method that starts with simple open-loop excitation and proceeds to a complete, well-tuned, closed-loop system.
To get started, this kit requires the following:

- A TMS320F28069M LaunchPad (LAUNCHXL-F28069M)
- A mini-USB cable
- A 9-V, 2-A DC-bench supply
- A computer with Code Composer Studio™ software v6.1 (CCS) or later
- The latest version of controlSUITE™
- The CCS Graphical User Interface (GUI) composer

Figure 1 shows the DPS BoosterPack and its key components. The LaunchPad connects at the BoosterPack headers. The nominal input bus voltage is 9-V DC, which can be supplied from a DC-bench power supply at JP1. The DC-DC synchronous buck power stage can regulate the nominal output DC voltage at 2 V. The active load provides a software-controlled switched load, which helps test transient performance and tune a loop.
Table 1. Key Components

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>9 V In (JP1)</td>
<td>Vin - DC-power supply (Minimum 8.5 V, Maximum 12 V)</td>
</tr>
<tr>
<td>BoosterPack Headers</td>
<td>LaunchPad interface</td>
</tr>
<tr>
<td>Control Power Switch</td>
<td>SW1 - Power switch to connect or disconnect the DC bus to or from Vin.</td>
</tr>
<tr>
<td>Buck Converter</td>
<td>Synchronous buck power stage (includes TI NexFET™ Power Block)</td>
</tr>
<tr>
<td>Load</td>
<td>7.5-Ω resistive load permanently connected across the output</td>
</tr>
<tr>
<td>Active Load</td>
<td>Software-controlled switched load (2 Ω)</td>
</tr>
<tr>
<td>Gate Drive</td>
<td>Buffered PWM drive to ensure quick and efficient turn on or off of the</td>
</tr>
<tr>
<td></td>
<td>power FETs</td>
</tr>
<tr>
<td>Current Sensing</td>
<td>Inductor current feedback for current monitoring and overcurrent protection</td>
</tr>
</tbody>
</table>

The TMS320F28069M microcontroller on the F28069M LaunchPad 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. Sense resistor R4 and op-amp U2 sense, condition, and feed the current through the inductor L1 back to the MCU. This synchronous buck power stage operates at a 200-kHz switching frequency. A low-load PR1 is permanently connected between the synchronous buck output and ground. A high-load PR2 can be switched in and out of the circuit using the MOSFET switch Q1 by driving Q1 with a 50% duty cycle PWM signal. This capability provides an active load feature at run time and allows transient performance tests and tuning. PR2 can also be turned on continuously to increase output load which causes the inductor current to stay positive and nonzero, and providing better loop response measurements.

Table 2 lists the key signal connections between the F28069M LaunchPad and the DPS BoosterPack. For a portion of the schematic, see Figure 2.

Table 2. Key Signal Connections

<table>
<thead>
<tr>
<th>Signal Name</th>
<th>Description</th>
<th>Connection to LaunchPad</th>
</tr>
</thead>
<tbody>
<tr>
<td>EPWM-4A</td>
<td>High-side drive signal for synchronous buck (PWM-HI)</td>
<td>GPIO-06</td>
</tr>
<tr>
<td>EPWM-4B</td>
<td>Low-side drive signal for synchronous buck (PWM-LO)</td>
<td>GPIO-07</td>
</tr>
<tr>
<td>EPWM-5A</td>
<td>Alternate low side drive signal (PWM-LO)</td>
<td>GPIO-08</td>
</tr>
<tr>
<td>EPWM-6A</td>
<td>PWM duty control signal for active load</td>
<td>GPIO-10</td>
</tr>
<tr>
<td>VoutFB-1</td>
<td>Output voltage feedback</td>
<td>ADC-A3</td>
</tr>
<tr>
<td>ILFB</td>
<td>Inductor current feedback (overcurrent protection)</td>
<td>ADC-A4</td>
</tr>
<tr>
<td>ILFB_AVG</td>
<td>Inductor current feedback (heavily filtered)</td>
<td>ADC-B4</td>
</tr>
<tr>
<td>VinFB</td>
<td>Input voltage feedback</td>
<td>ADC-B3</td>
</tr>
</tbody>
</table>
To start experimenting with this board, follow the steps in this manual.

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.
1 Software Flow

The software project uses the C-background and C-ISR framework and C-callable assembly functions. This project uses C-code as the main supporting program for the application and manages all system tasks, decision making, intelligence, and host interaction. The assembly code is executed inside the interrupt service routine (ISR) and runs the critical control code. The ISR includes ADC reading, control calculations, and PWM updates. Figure 5 presents the general software flow for this project and the library functions of the software frequency response analyzer (SFRA).

TI designed the SFRA library to enable frequency response analysis on digitally controlled power converters using software. This library enables performing frequency response analysis of the power converter easily without requiring external connections or equipment. In high-frequency power conversion applications, the library can identify the plant and the open-loop characteristics of a closed-loop power converter. This library can help provide stability information such as bandwidth, gain margin, and phase margin to evaluate the control-loop performance. Refer to the C2000 Software Frequency Response Analyzer (SFRA) Library and Compensation Designer User’s Guide (SPRUHZ5) for more information.

This kit also supports other powerSUITE tools including the compensation designer and the solution adapter. These tools help you evaluate the complete system, adapt it for your application, and tune it for improved performance.
Figure 6 shows an overview of 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.

Figure 6. Designing and Tuning a System Process Flow Chart

The solution adapter lets you adapt code examples from TI digital power supply kits and configure them to run on your custom digital power supply board using the same topology and similar resources. The GUI guides you 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 your hardware design.

The compensation designer is a software utility that helps power engineers design multiple digital compensators that achieve the closed-loop performance of a closed-loop control system they desire. You can achieve this performance using the measured power stage or plant data from the SFRA tool or using the modeled power stage as part of the solution adapter. The solution adapter generates the coefficients that require programming on the device and can be copied into the code directly. The default project uses a 2-pole 2-zero control law and does not support 3-pole 3-zero compensators.

The key framework C files in this project are the following:
- Buck_VMC-Main.c – This file initializes, runs, and manages the application.
- Buck_VMC-DevInit_F2806x.c – This file initializes and configures the microcontroller once. This one-time initialization and configuration includes setting up the clocks, PLL, GPIO, and so forth.
- Buck_VMC-DPL.asm – This file contains all time, critical control-type code in a C-callable assembly function and has an initialization section and a runtime function from the ISR.

The digital power library modules are called from this framework.
Library modules may have both a C and an assembly component. Table 3 lists the C configure functions, their initialization macros, and runtime macros.

<table>
<thead>
<tr>
<th>C Configure Function</th>
<th>ASM Initialization Macro</th>
<th>ASM Runtime Macro</th>
</tr>
</thead>
<tbody>
<tr>
<td>PWM_1ch_UpCntDB_ActivHIC_Cnf.c</td>
<td>PWMDRV_1ch_INIT n</td>
<td>PWMDRV_1ch n</td>
</tr>
<tr>
<td>ADC_SOC_Cnf.c</td>
<td>ADCDRV_1ch_INIT n</td>
<td>ADCDRV_1ch n</td>
</tr>
<tr>
<td></td>
<td>CNTL_2P2Z_INIT n</td>
<td>CNTL_2P2Z n</td>
</tr>
</tbody>
</table>

Table 3. Library Modules

Figure 7 shows the control blocks.

The dark teal blocks represent hardware modules on the C2000 microcontroller. Light blue blocks are the software drivers for these modules. The yellow block is the controller block for the control loop. Although a 2-pole 2-zero controller is used in this design, you can use a PIPID, a 3-pole 3-zero, or any other controller that can be implemented for this application. This modular library structure lets you conveniently visualize and understand the complete system software flow and easily use and add or delete functions. By implementing an incremental build approach, this project demonstrates the these functions.
2 Incremental Builds

This project is divided into three incremental builds. Dividing this project into three builds makes it easier to learn and become familiar with the board and the software and benefits for debugging and testing boards.

The build options are the following.

Incremental Build Options

INCR_BUILD = 1—Open-loop check (check PWM drive circuit and sensing circuit) with SFRA.

INCR_BUILD = 2—Closed voltage loop (check full system functionality) with SFRA.

INCR_BUILD = 3—Closed voltage loop (check full system functionality) without SFRA. See Table 2.

To select a build option, do the following:
1. Select the corresponding project in main.cfg.
2. Select rebuild-all compiler to compile the complete project.
Running Incremental Builds

The main source files, ISR assembly file, and the project file for C framework to bring up the system are in the project directory. The projects included with this software are targeted for CCSv6.1 and higher.

The following sections present the steps to build and run the software.

1 Build 1: Open-Loop Control With SFRA

1.1 Objective
The objective of this build is to become familiar with the BoosterPack hardware and to control the buck output voltage using direct PWM duty cycle adjustments without feedback. Because this system is running as an open loop, the ADC measured values are used only for instrumentation in this build. You can adjust the PWM duty cycle using the expressions window. To get the frequency response of the plant, you can use SFRA GUI during run time.

1.2 Overview
The software is configured to adjust the duty cycle of the PWM output for the module selected in main.cfg (for the BoosterPack board, this is PWM4). Variable Duty1A is connected to the input of the PWMDRV_1ch macro. You can adjust the duty command from the expressions window by using the variable Duty1A_Set. Figure 8 presents the system diagram for this build.

![Figure 8. System Diagram](image)

The on-chip analog comparator (selected in the main.cfg file) and corresponding DAC mechanism provide overcurrent protection (for the BoosterPack board this is COMP2). You can use the Gui_ItripSet variable to set the reference trip level for the comparator. The comparator output is configured to generate a one-shot trip action on the PWM module (selected in the main.cfg file) whenever the sensed current is greater than the set limit. The flexibility of the trip mechanism on C2000 devices offers the possibility to assign which actions should be taken by the PWM pin (set hi, set low, set high-impedance) when specific trip events occur. In this project, EPWM4A and EPWM4B outputs are driven low immediately on a comparator event to protect the power stage.

The buck converter is driven by a 200-kHz PWM. The DPL_ISR_wFRA interrupt routine is triggered by the PWM module. This interrupt calls the C-callable assembly function DPL(Func). This function is where the PWMDRV_1ch and ADCDRV_1ch macros are executed.

A task state machine is part of the background code. Tasks are arranged in groups (A1, A2, A3…, B1, B2, B3…, C1, C2, C3…). Each group is executed according to three CPU timers which are configured with periods of 1 ms, 10 ms, and 100 ms, respectively. Within each group, each task is run in a round-robin manner. For example, group B executes every 10 ms and group B has four tasks. Each of the tasks B1, B2, B3 and B4 execute once every 40 ms.

1.3 Procedure

1.3.1 Hardware Setup

1. To get started with the BoosterPack, you will also need an F28069M LaunchPad, with a mini-USB cable, a 9-V, 2-A DC-bench power supply, and a computer with CCS version 6.1 or newer and the
latest version of controlSUITE installed on it. Ensure you also have CCS GUI composer installed.

2. Plug in the LaunchPad to BoosterPack headers. (See Figure 9.)

Figure 9. Connecting the LaunchPad to the BoosterPack

5. Turn on all three S1 switches on the LaunchPad.
6. Connect the LaunchPad using the included mini-USB cable to a computer with CCSv6.1 or later, the latest version of controlSUITE, and CCS GUI composer.
7. Turn off SW1 on the BoosterPack.
9. Set the DC-bench supply to output 9 V.
1.3.2 Software Setup

1. Open CCS.
2. Navigate to View.
4. Navigate to controlSUITE.
5. Navigate to English.
6. Navigate to powerSUITE.
7. Navigate to Development Kits.
8. Navigate to Digital Power BoosterPack (BOOSTXL-BUCKCONV.)
10. Navigate to Buck_VMC_F28069M.
11. Click Import to import the project into CCS.

**NOTE:** The Buck_VMC_F28069M project appears in the CCS Project Explorer window. This project invokes the tools (for example, the compiler, the assembler, and the linker) to build the project. A project contains all the files and build options to develop an executable output file (.out), which can run on the MCU hardware.

12. Click the arrow sign to the left of the project name.
13. Double-click main.cfg file (This file displays all the relevant device resources information, power stage parameters and project options.)
14. Under Project Options, select Open Loop. (See Figure 10.)
15. Click Save main.cfg.
Figure 10. Main.cfg File Selections for Build 1
### 1.3.3 Build and Load the Project

1. Click Clean Project. (If another build option exists, right-click on the project name.)
2. Click Project.
3. Click Build All.
4. Click 🚀. (The build 1 code should compile and load.)

**NOTE:**
The CCS Debug icon in the upper right-hand corner indicates the Debug Perspective view. The program will stop at the start of main().

### 1.3.4 Debug Environment Windows

Watching local and global variables while debugging code is standard debug practice. Code Composer Studio can create time and frequency domain plots. This capability lets you view waveforms using graph windows.

1. Click View.
2. Click scripting console on the menu bar.
3. Use the scripting console Open File command to open the AddWatchWindowVars.js file from the project directory. (To see the expressions window, see Figure 11.)

<table>
<thead>
<tr>
<th>Expression</th>
<th>Type</th>
<th>Value</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Duty1A</td>
<td>long</td>
<td>0.0 (Q-Value(24))</td>
<td>0x0000A06C@Data</td>
</tr>
<tr>
<td>Duty1A_Set</td>
<td>long</td>
<td>0.0 (Q-Value(24))</td>
<td>0x0000A066@Data</td>
</tr>
<tr>
<td>Gui_Vin</td>
<td>int</td>
<td>0.00683594 (Q-Value(11))</td>
<td>0x0000A023@Data</td>
</tr>
<tr>
<td>Gui_Vout1</td>
<td>int</td>
<td>0.00488281 (Q-Value(12))</td>
<td>0x0000A020@Data</td>
</tr>
<tr>
<td>Gui_VSet1</td>
<td>int</td>
<td>0.0 (Q-Value(12))</td>
<td>0x0000A028@Data</td>
</tr>
<tr>
<td>Gui_JL1</td>
<td>int</td>
<td>0.0187988 (Q-Value(12))</td>
<td>0x0000A024@Data</td>
</tr>
<tr>
<td>Gui_ItripSet</td>
<td>int</td>
<td>6.0 (Q-Value(12))</td>
<td>0x0000A01F@Data</td>
</tr>
<tr>
<td>FaultFg1</td>
<td>int</td>
<td>0</td>
<td>0x0000A003@Data</td>
</tr>
<tr>
<td>ClearFault1</td>
<td>int</td>
<td>0</td>
<td>0x0000A002@Data</td>
</tr>
<tr>
<td>Active_LD1_EN</td>
<td>int</td>
<td>0</td>
<td>0x0000A001@Data</td>
</tr>
<tr>
<td>Continuous_ON</td>
<td>int</td>
<td>0</td>
<td>0x0000A009@Data</td>
</tr>
<tr>
<td>pid2p2z_Gui</td>
<td>int</td>
<td>1</td>
<td>0x0000A010@Data</td>
</tr>
<tr>
<td>coeff_change</td>
<td>int</td>
<td>1</td>
<td>0x0000A011@Data</td>
</tr>
<tr>
<td>Pgain1_Gui</td>
<td>int</td>
<td>100</td>
<td>0x0000A00D@Data</td>
</tr>
<tr>
<td>Igain1_Gui</td>
<td>int</td>
<td>5</td>
<td>0x0000A008@Data</td>
</tr>
</tbody>
</table>

**Figure 11. Expressions Window**
## Table 4. Expressions Window Variable Description

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Duty1A</td>
<td>Q24 is the value of duty input to PWMDRV_1ch. This is slewed value based on Duty1A_Set. If used, SFRA signal is injected here.</td>
</tr>
<tr>
<td>Duty1A_Set</td>
<td>Q24 is the value of duty input. This value is set by the user in build 1.</td>
</tr>
<tr>
<td>Gui_Vin</td>
<td>This is the input voltage in volts (when displayed in Q11)</td>
</tr>
<tr>
<td>Gui_Vout1</td>
<td>This is the output voltage in volts (when displayed in Q12)</td>
</tr>
<tr>
<td>Gui_VSet1</td>
<td>This is the output voltage in volts set by user (when displayed in Q12). Not used in build 1.</td>
</tr>
<tr>
<td>Gui_IL1</td>
<td>This is the instantaneous inductor current in amps (when displayed in Q12.)</td>
</tr>
<tr>
<td>Gui_ItripSet</td>
<td>This is the overcurrent shut-down level in amps (when displayed in Q12.)</td>
</tr>
<tr>
<td>FaultFlg1</td>
<td>This is the overcurrent PWM shut-down flag.</td>
</tr>
<tr>
<td>ClearFault1</td>
<td>Clears fault flag and re-enable PWM.</td>
</tr>
<tr>
<td>Active_LD1_EN</td>
<td>Enables and disable the active load feature.</td>
</tr>
<tr>
<td>Continuous_ON</td>
<td>When 1 is active, the load is permanently connected at the output. When 0 is active, the load switches in and out of the circuit for transient performance tests.</td>
</tr>
<tr>
<td>pid2p2z_Gui</td>
<td>Selects between loosely tuned PID-based compensator (0) and compensator designed using the Compensation Designer GUI (1).</td>
</tr>
<tr>
<td>coeff_change</td>
<td>Calculates and updates coefficients.</td>
</tr>
<tr>
<td>Pgain1_Gui</td>
<td>This is the scaled (0–999) proportional gain value for PID.</td>
</tr>
<tr>
<td>Igain1_Gui</td>
<td>This is the scaled (0–999) integral gain value for PID.</td>
</tr>
</tbody>
</table>

### 1.3.5 Using Real-time Emulation

Real-time emulation is a feature that lets windows within Code Composer Studio update while the MCU is running. This capability also lets you change variables or memory location values and have those changes affect the MCU behavior by changing the voltage set-point, enabling the active load, and so forth. This ability helps when tuning control law parameters on-the-fly.

1. Enable real-time mode by hovering your mouse on the buttons on the horizontal toolbar
2. Click on the real-time mode button.
3. If a message box appears, select YES to enable debug events.

**NOTE:** Selecting yes will set bit 1 (DGBM bit) of status register 1 (ST1) to 0. The DGBM is the debug enable mask bit. When the DGBM bit is set to 0, memory and register values can be passed to the host processor for updating the debugger windows.

4. Right-click in the expressions window.
5. Select the Continuous Refresh Interval....

**NOTE:** When a large number of windows are open, the bandwidth over the emulation link is limited. Updating too many windows and variables in continuous refresh can cause the refresh frequency to bog down. Slow down the refresh rate for the expressions window variables by changing the continuous refresh interval (milliseconds) value to a rate of 1000 ms.

6. Click for the watch view.
1.3.6 Run the Code  
1. Press F8 to run the code.  
2. Turn on the 9-V supply.  
3. Observe Gui_Vin in the watch window (it should now be 9 V).  
4. Turn on switch SW1 on the BoosterPack.  
5. Increase Duty1A_Set to 0.3 in Q24.  
6. Observe the output voltage Gui_Vout1.

**NOTE:** The output voltage should be around 2.6 V (in an open loop, this voltage depends on the input voltage value and may vary from 2.3 V to 3.3 V). If output voltage is close to 0 V and FaultFlg1 is set, change Duty1A_Set to 0.0 (Q24). Set ClearFault1 to 1. Check whether FaultFlg1 is 0. You can now change Duty1A_Set to 0.3 (Q24) and the output voltage should change. The Gui_ItripSet expressions window entry programs the overcurrent shut-down level (in amperes).

*Figure 12* shows a watch window that corresponds to the operation of the system when Duty1A_Set is 0.3.

<table>
<thead>
<tr>
<th>Expression</th>
<th>Type</th>
<th>Value</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Duty1A</td>
<td>long</td>
<td>0.2999998927</td>
<td>00000A06C@Data</td>
</tr>
<tr>
<td>Duty1A_Set</td>
<td>long</td>
<td>0.2999998927</td>
<td>00000A066@Data</td>
</tr>
<tr>
<td>Gui_Vin</td>
<td>int</td>
<td>9.23779 (Q-Value(11))</td>
<td>00000A023@Data</td>
</tr>
<tr>
<td>Gui_Vout1</td>
<td>int</td>
<td>2.61841 (Q-Value(12))</td>
<td>00000A020@Data</td>
</tr>
<tr>
<td>Gui_Vset1</td>
<td>int</td>
<td>0.0 (Q-Value(12))</td>
<td>00000A028@Data</td>
</tr>
<tr>
<td>Gui_I1L1</td>
<td>int</td>
<td>0.271729 (Q-Value(12))</td>
<td>00000A024@Data</td>
</tr>
<tr>
<td>Gui_ItripSet</td>
<td>int</td>
<td>6.0 (Q-Value(12))</td>
<td>00000A01F@Data</td>
</tr>
<tr>
<td>FaultFlg1</td>
<td>int</td>
<td>0</td>
<td>00000A003@Data</td>
</tr>
<tr>
<td>ClearFault1</td>
<td>int</td>
<td>0</td>
<td>00000A002@Data</td>
</tr>
<tr>
<td>Active_LD1_EN</td>
<td>int</td>
<td>0</td>
<td>00000A003@Data</td>
</tr>
<tr>
<td>Continuous_ON</td>
<td>int</td>
<td>0</td>
<td>00000A009@Data</td>
</tr>
<tr>
<td>pid2p2z_Gui</td>
<td>int</td>
<td>1</td>
<td>00000A010@Data</td>
</tr>
<tr>
<td>coeff_change</td>
<td>int</td>
<td>0</td>
<td>00000A011@Data</td>
</tr>
<tr>
<td>Pgain1_Gui</td>
<td>int</td>
<td>100</td>
<td>00000A00D@Data</td>
</tr>
<tr>
<td>Igain1_Gui</td>
<td>int</td>
<td>5</td>
<td>00000A00B@Data</td>
</tr>
</tbody>
</table>

*Add new expression*

*Figure 12. Expressions Window in Build 1*

7. Change the Gui_ItripSet level to 0.2 A (Q12)

**NOTE:** The output voltage goes to 0 and the FaultFlg1 should change to 1 in the expressions window because of an overcurrent trip.

8. Change Gui_ItripSet to 6 A in the expressions window.
9. Change Duty1A_Set to 0.
10. Set ClearFault1 to 1 in the expressions window. (FaultFlg1 should be 0. The overcurrent condition has been cleared.)
11. Change Duty1A_Set to 0.3 (Q24).
12. Watch the output voltage return to the previous value.
Using SFRA to View the Frequency Response

**NOTE:** You can use the SFRA to view the frequency response of plant. For best frequency response measurements, increase the output load.

1. Set Active_LD1_EN to 1 in the expressions window.
2. Set Continuous_ON to 1.
3. Click the main.cfg file.
4. Open SFRA.
5. Click Setup Connection.
6. Select the appropriate COM port.
7. Set the baud rate to 57600.
8. Ensure Boot on Connect is unchecked.
9. Click OK.
10. Click Connect on the SFRA GUI.
11. When the GUI is connected, click Start Sweep.

![Software Frequency Response Analyzer](image)

**Figure 13. Frequency Response for Build 1**
NOTE: SFRA will start injecting different frequencies and collecting the response for frequency analysis. When the frequency sweep completes the response displays on the SFRA GUI.

12. Close the SFRA GUI.

NOTE: You can change the Duty1A_Set value from the expressions window to ensure that the BoosterPack operates within its capabilities. Duty1A_Set has been restricted to a maximum value of 0.45 (Q24) in this build.

13. Turn off SW1 on the BoosterPack.
14. Turn off the 9-V power supply.

Fully Halting the MCU in Real-time Mode
1. Click on the toolbar to halt the processor.
2. Click to take the MCU out of real-time mode.
3. Click Run.
4. Click Reset.
5. Click CPU Reset.

NOTE: Leave Code Composer Studio running for the next build or close CCS.

2 Build 2: Closed-Loop Control with SFRA

2.1 Objective
The objective of this build is to regulate the output voltage of a buck power stage using closed-loop feedback control realized in the form of a software-coded control loop. Use a compensator designed using the Compensation Designer GUI to achieve desired closed-loop performance. You can use SFRA GUI during runtime to capture the frequency response of the system. An active-load circuit turned on by the software provides a repetitive step change in the load to test the transient performance of the system in time-domain with the help of the CCS graph window.

2.2 Overview
The software has been configured to provide closed-loop voltage control for the buck power stage. A 2-pole 2-zero controller block (CNTL_2P2Z) implements the control law. The output voltage feedback Adc_Vout1 is an input to this block. The reference input to the CNTL_2P2Z block comes from the slewed output voltage command Vout_ref_wInj. If used, the SFRA signal is injected at the input of the controller. Similar to build 1, the controller output Duty1A must be connected to the PWMDRV_1ch block. You can adjust the output voltage command from the expressions window using the variable Gui_VSet1. Figure 14 shows the system diagram for this build.
Similar to build 1, the on-chip analog comparator (selected in the main.cfg file) and corresponding DAC mechanism provide overcurrent protection (for the BoosterPack board this is COMP2). You can use the Gui_ItripSet variable to set the reference trip level for the comparator. The comparator output is configured to generate a one-shot trip action on the PWM module (selected in the main.cfg file) whenever the sensed current is greater than the set limit. The flexibility of the trip mechanism on C2000 devices offers the possibility to assign which actions should be taken by the PWM pin (set hi, set low, or set high-impedance) when specific trip events occur. In this project, EPWM4A and EPWM4B outputs are driven low immediately on a comparator event to protect the power stage.

The buck converter is driven with a 200 kHz PWM. The DPL_ISR_wFRA interrupt routine is triggered by the selected PWM module. This interrupt calls the C-callable 20 assembly function DPL_Func. This is where the CNTL_2P2Z, PWMDRV_1ch and ADCDRV_1ch macros are executed.

Similar to build 1, a task state machine is part of the background code. Tasks are arranged in groups (A1, A2, A3..., B1, B2, B3..., C1, C2, C3...). Each group is executed according to three CPU timers which are configured with periods of 1 ms, 10 ms, and 100 ms, respectively. Within each group, each task is run in a round-robin manner. For example, group B executes every 10 ms and group B has four tasks. The collective execution time for B1, B2, B3, and B4 tasks is 40 ms.

Figure 14. System Diagram
2.3 Procedure

2.3.1 Hardware Setup

Follow the steps listed under hardware setup section in the procedure from build 1.

2.3.2 Software Setup

If CCS is already open after completing build 1, skip to step 7.

1. Open CCS.
2. Navigate to View.
4. Navigate to controlSUITE.
5. Navigate to English.
6. Navigate to powerSUITE.
7. Navigate to Development Kits.
10. Navigate to Buck_VMC_F28069M (the demonstration).
11. Click Import the example project into CCS.

NOTE: The Buck_VMC_F28069M project appears in the CCS Project Explorer window. This project starts to invoke all the necessary tools (for example, the compiler, the assembler, and the linker) to build the project. A project contains all the files and build options needed to develop an executable output file (.out) that can be run on the MCU hardware.

12. Click the arrow sign to the left of the project name
13. Double-click main.cfg. (This file displays all the relevant device resources information, power stage parameters, and project options.)
14. Under Project Options, select Closed Voltage Loop. (See Figure 15.)
Figure 15. Main.cfg File Selections for Build 2
15. Click Compensation Designer.

**NOTE:** The Compensation Designer GUI uses the power stage model based on the data entered in the main.cfg file and allows you to design compensators to achieve the closed-loop performance you desire. Leave the default parameters for the designer.

16. Select the first compensator.

**NOTE:** This compensator is a 2-pole 2-zero compensator. When you select a new compensator from the list, you update the magnitude and phase plots for the plant, open loop, and compensator. These plots also update when you change the compensator. The default software supports a 2-pole 2-zero compensator.

17. Note the bandwidth, gain margin, and phase margin for the default compensator 1.

18. Close the Compensation Designer GUI.

19. Using the default project options, save the main.cfg.

You are ready to build and load the project.

### 2.3.3 Build and Load the Project

1. Click Project. (Alternatively, If another build option was built, right-click on the project name then click Clean Project.)
2. Click Build All.
3. Watch the tools run in the build window.
4. Click 🚀. (Alternatively, click Run then click Debug.)

**NOTE:** The build 2 code compiles and loads. The CCS Debug icon in the upper right-hand corner indicates you are in the Debug Perspective view. The program will stop at the start of main().

### 2.3.4 Debug Environment Windows

Watching local and global variables while debugging code is standard debug practice. Code Composer Studio can create time and frequency domain plots. This capability lets you view waveforms using graph windows.

1. To populate the expressions window entries, click View.
2. Click Scripting Console on the menu bar.
3. Open the AddWatchWindowVars.js file from the project directory using 📝, the scripting console Open File command.

The expressions window should appear as shown in Figure 16.
### Table 5. Expressions Window Variable Description

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Duty1A</td>
<td>This is the Q24 value of duty input to PWMDRV_1ch.</td>
</tr>
<tr>
<td>Duty1A_Set</td>
<td>This is the Q24 value of duty input. Not used in build 2.</td>
</tr>
<tr>
<td>Gui_Vin</td>
<td>This is the input voltage in volts (when displayed in Q11.).</td>
</tr>
<tr>
<td>Gui_Vout1</td>
<td>This is the output voltage in volts (when displayed in Q12.).</td>
</tr>
<tr>
<td>Gui_VSet1</td>
<td>This is the output voltage set by user (when displayed in Q12.).</td>
</tr>
<tr>
<td>Gui_IL1</td>
<td>This is the instantaneous inductor current in amp (when displayed in Q12.).</td>
</tr>
<tr>
<td>Gui_ItripSet</td>
<td>This is the overcurrent shut down level in amp (when displayed in Q12.).</td>
</tr>
<tr>
<td>FaultFlag1</td>
<td>This is the overcurrent PWM shut-down flag.</td>
</tr>
<tr>
<td>ClearFault1</td>
<td>Clears fault flag and re-enable PWM.</td>
</tr>
<tr>
<td>Active_LD1_EN</td>
<td>Enables and disables the active load feature.</td>
</tr>
<tr>
<td>Continuous_ON</td>
<td>When 1 is active, the load is permanently connected at the output. When 0 is active, the load switches in and out of the circuit for transient performance tests.</td>
</tr>
<tr>
<td>pid2p2z_Gui</td>
<td>Selects between loosely-tuned PID based compensator (0) and compensator designed using the Compensation Designer GUI (1).</td>
</tr>
<tr>
<td>coeff_change</td>
<td>Calculates and updates coefficients.</td>
</tr>
<tr>
<td>Pgain1_Gui</td>
<td>This is the scaled (0–999) proportional gain value for PID.</td>
</tr>
<tr>
<td>Igain1_Gui</td>
<td>This is the scaled (0–999) integral gain value for PID.</td>
</tr>
</tbody>
</table>

### Opening a Graph Window
1. Click Tools.
2. Click Graph.
3. Click Single Time.

**NOTE:** This creates a one time graph window to plot the data log buffer DBUFF1.

4. Change Acquisition Buffer Size to 400.
5. Change the Display Data Size to 400.
6. Select 16-bit unsigned integer as Data Type.
7. Select 12 as the Q Value.
8. Select DBUFF1 as the start address. (See Figure 17.)

![Graph Properties](image)

**Figure 17.** Graph Window Property Settings

9. Click OK.
2.3.5 Using Real-time Emulation

Real-time emulation is a special emulation feature that lets windows within Code Composer Studio update while the MCU is running. This feature lets graphs and watch windows update, while letting you change variables or memory location values. The changes to variable and memory location values immediately affect the MCU behavior. This feature is useful when tuning control law parameters on-the-fly.

1. Hover the mouse over the icon on the horizontal toolbar.

2. Click [Enable Silicon Real-time Mode (service critical interrupts when halted, allow debugger accesses while running)].

**NOTE:** A message box may appear. If so, select YES to enable debug events. This sets bit 1 (DGBM bit) of status register 1 (ST1) to 0.

The DGBM is the debug enable mask bit. When the DGBM bit is set to 0, memory and register values can be passed to the host processor for updating the debugger windows.

When a large number of windows are open, as bandwidth over the emulation link is limited, updating too many windows and variables in continuous refresh can cause the refresh frequency to bog down.

(a) Right-click in the Expressions window.

(b) Select the Continuous Refresh Interval…

(c) Change the Continuous refresh interval (milliseconds) value to slow down the refresh rate for the expressions window variables. (A rate of 1000 ms is typical.)

(d) Click in the graph window.

(e) Click in the expressions window.

2.3.6 Run the Code

1. Press F8. (Alternatively, click Run.)

2. Turn on the 9-V supply.

3. Observe that the Gui_Vin value is approximately 9 V in the watch window.

4. Turn on switch SW1 on the BoosterPack.

**NOTE:** If this is the first time SW1 has been turned on since a device reset, the output voltage (Gui_Vout1) ramps up to 2 V.

If not, change Gui_VSet1 to 2.0 (Q12).

If output voltage does not ramp up to 2 V and FaultFlg1 is Set, do the following:

(a) Change Gui_VSet1 to 0.0 (Q12).

(b) Set ClearFault1 to 1. (FaultFlg1 should be 0.)

(c) Change Gui_VSet1 to 2.0 (Q12). (The output voltage should change from 0 V to 2 V. See Figure 18.)
5. To get good frequency response measurements, set Active_LD1_EN to 1 in the expressions window.
6. Set Continuous_ON to 1. (The output voltage should still be regulated at 2 V.)
7. Click the main.cfg file.
8. Open SFRA.
9. Click Setup Connection.
10. Select the appropriate COM port.
11. Ensure that Boot on Connect is unchecked.
12. Click OK.
13. Click Connect on the SFRA GUI.
14. When the GUI is connected, click Start Sweep.

**NOTE:** SFRA will start injecting different frequencies and collecting the response for frequency analysis. You might also see the effect of this process on the output voltage in the graph window. The high-frequency signals riding on the output voltage indicate an active SFRA run.
When the frequency sweep completes the response displays on the SFRA GUI. The bandwidth, gain margin, and phase margin should be similar to the values noted on the compensation designer GUI. These values may differ to varying degrees based on how closely the model parameters match the actual power stage values.
15. Close the SFRA GUI.
16. Set Continuous_ON to 0.
17. Change pid2p2z_Gui to 0.
18. Set coeff_change to 1.

**NOTE:** Changing pid2p2z_Gui to 0 and setting coeff_change to 1 change the compensator from the compensator designed using the Compensation Designer GUI to a loosely tuned PID-based compensator that has been manually tuned using Pgain1_Gui and Igain1_Gui.

19. Zoom-in on the output voltage transients on the graph window. (The output voltage spikes whenever the active load is turned on and off.)
21. Set coeff_change to 1.
22. Return to the original compensator from the Compensation Designer GUI. (The output voltage spikes voltage transients have reduced. See Figure 24.)

Figure 24. Transient Response Obtained Using Compensation Designer GUI Coefficients

The reduced spikes in the output voltage verifies the improved transient performance of the system in time-domain.

23. Turn off SW1 on the BoosterPack.
24. Turn off the 9-V power supply.
25. Click on the toolbar to halt the processor. (Alternatively, to halt the processor, click Run then .)
26. Click to take the MCU out of real-time mode.
27. Click Run.
28. Click Reset.
29. Click CPU reset. (Leave Code Composer Studio running for the next build or close CCS.)

3 Build 3: Closed-Loop Control Without SFRA

This build option is similar to build 2 but runs without SFRA. This omission provides a method to control the system without the additional run-time SFRA software. To run this build, follow the steps in build 2. The SFRA tool may not be used with this build.
Adapting This TI Solution

When you have evaluated this TI solution, use the solution adapter to adapt this solution to run on your custom digital power supply that uses the same topology and similar resources.

1. Open a new CCS workspace.
2. Open Resource Explorer.
3. Navigate to controlSUITE.
4. Navigate to English.
5. Navigate to powerSUITE.
7. Click Buck.

Figure 25. Navigating to Solution Adapter in TI Resource Explorer
8. Click Voltage Mode Control.

![VOLTAGE MODE CONTROL: TMS320F28069](image)

**Figure 26. Project Selections From TI Resource Explorer**

9. Specify a destination location for the project.
10. Click OK.

The main.cfg file opens. You can change the resource mapping for different ADC inputs and PWM outputs to match your design. You can also change voltage and current scaling, and enter power stage parameters to match your design.
This capability lets you quickly adapt the TI software without having to write code. You can test a custom design with the procedures in the previous sections.

If further code customization is necessary, you may need to modify the source files (Buck_VMC-Main.c, Buck_VMC-DPL.asm, Buck_VMC-DevInit_F2806x.c, and so forth).

Figure 27. Main.cfg for Solution Adapter
TI has specifically designated certain components as meeting ISO/TS16949 requirements, mainly for automotive use. In any case of use of military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components that have executed an agreement specifically governing such use. No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have performed the necessary analysis, evaluation and judgment in designing Buyer’s systems and products. TI reference designs have been created using standard laboratory conditions and engineering practices. TI has not conducted any testing other than that specifically described in the published documentation for a particular reference design. TI may make corrections, improvements, and other changes to its reference designs.

Buyers are authorized to use TI reference designs with the TI component(s) identified in each particular reference design and to modify the reference design in the development of their end products. HOWEVER, NO OTHER LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE TO ANY OTHER TI INTELLECTUAL PROPERTY RIGHT, AND NO LICENSE TO ANY THIRD PARTY TECHNOLOGY OR INTELLECTUAL PROPERTY RIGHT, IS GRANTED HEREIN, including but not limited to 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. TI references DESIGNS ARE PROVIDED "AS IS", TI MAKES NO WARRANTIES OR REPRESENTATIONS WITH REGARD TO THE REFERENCE DESIGNS OR USE OF THE REFERENCE DESIGNS, EXPRESS, IMPLIED OR STATUTORY, INCLUDING ACCURACY OR COMPLETENESS. TI DISCLAIMS ANY WARRANTY OF TITLE AND ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, QUIET ENJOYMENT, QUIET POSSESSION, AND NON-INFRINGEMENT OF ANY THIRD PARTY INTELLECTUAL PROPERTY RIGHTS WITH REGARD TO TI REFERENCE DESIGNS OR USE THEREOF. TI SHALL NOT BE LIABLE FOR AND SHALL NOT DEFEND OR INDEMNIFY BUYERS AGAINST ANY THIRD PARTY INFRINGEMENT CLAIM THAT RELATES TO OR IS BASED ON A COMBINATION OF COMPONENTS PROVIDED IN A TI REFERENCE DESIGN. IN NO EVENT SHALL TI BE LIABLE FOR AND SHALL NOT DEFEND OR INDEMNIFY BUYERS AGAINST ANY THIRD PARTY INFRINGEMENT CLAIM THAT RELATES TO OR IS BASED ON A COMBINATION OF COMPONENTS PROVIDED IN A TI REFERENCE DESIGN. IN NO EVENT SHALL TI BE LIABLE FOR ANY ACTUAL, SPECIAL, INCIDENTAL, CONSEQUENTIAL OR INDIRECT DAMAGES, HOWEVER CAUSED, ON ANY THEORY OF LIABILITY AND WHETHER OR NOT TI HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES, ARISING IN ANY WAY OUT OF TI REFERENCE DESIGNS OR BUYER’S USE OF TI REFERENCE DESIGNS.

TI reserves 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 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 for TI components 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.

Reproduction of significant portions of TI information in TI data books, data sheets or reference designs 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.

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 the necessary expertise to create and implement safeguards that anticipate dangerous failures, monitor failures and their consequences, lessen the likelihood of dangerous failures 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 Buyer's 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 an agreement specifically governing such use.

Only those TI components that 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 that have not been so designated is solely at Buyer's risk, and 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.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265
Copyright © 2015, Texas Instruments Incorporated