Application Report
SPRABS9–July 2013

High-Voltage Half-Bridge LLC Resonant DC/DC Converter Software With Synchronous Rectification Kit

Daniel Chang

ABSTRACT

This application report presents a solution to control a half-bridge LLC resonant DC/DC converter with synchronous rectification system using TMS320F2802x microcontrollers. The Piccolo™ TMS320F2802x series of devices are part of the family of C2000™ microcontrollers that enable cost-effective design of power supply systems. With these devices, it is possible to control power stages in an efficient and accurate way. In addition to this, the speed of the C2000 microcontroller allows it to integrate many supplemental tasks that would, in a normal system, increase chip count and complexity. These tasks could include synchronous rectification, system management, and various communication protocols.

This document covers how to run and get familiar with the high-voltage half-bridge LLC resonant DC/DC converter with synchronous rectification kit's HVLLC project. For an in-depth discussion of LLC resonant DC/DC converters and their design considerations, along with a design process example, see the SEM1900 Topic 3, *Designing an LLC Resonant Half-Bridge Power Converter* by Texas Instruments. The details of the discussion found in the SEM1900 Topic 3 will not be repeated in this document.

Contents

List of Figures

List of Tables

1 System Overview

1.1 Hardware Overview

The high-voltage half-bridge LLC resonant DC/DC converter with synchronous rectification kit power board has the following electrical specifications:

- Input voltage: 375 to 405 VDC
- Rated output power: 300 W
- Output voltage : 12 VDC
- Rated output current: 25 A
- Output voltage line regulation (Io = 1 A): ≤ 1%
- Output voltage load regulation (V_{IN} = 390 V): ≤1%
- Output voltage peak-to-peak ripple (V_{IN} = 390 V and Io = 25 A): ≤120mV
- Efficiency (V_{IN} = 390 V and Io = 25 A): >90%
- Switching frequency (normal operation): 80 kHz to 150 kHz
- Resonant frequency: f0 = ~130 kHz

Figure 1 shows the LLC resonant power stage of the board in a circuit diagram format and illustrates the major connections and feedback values being mapped to the C2000 MCU; Table 1 lists these resources. It is important to note that not all resources are available on every C2000 MCU. For more detailed information, see the schematics and device-specific data sheets.

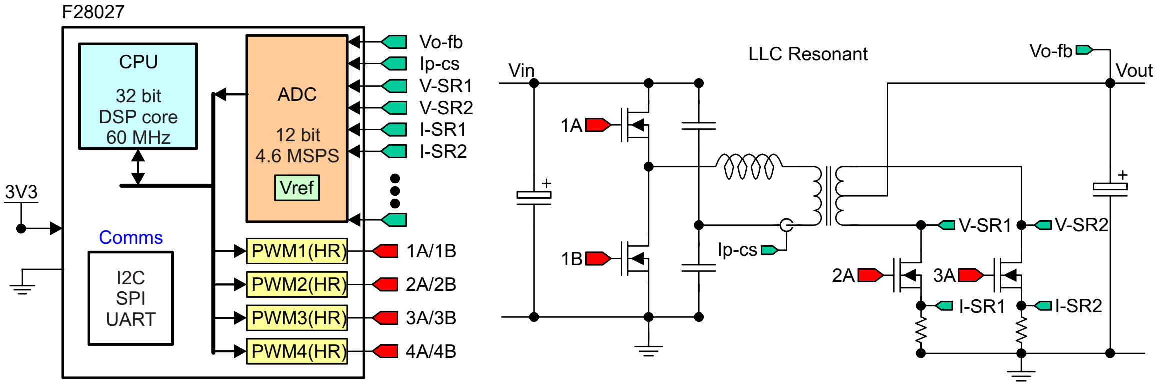


Figure 1. TMDSHVRESLLCKIT Circuit Diagram

Table 1. PWM and ADC Resource Allocation

Macro Name			Signal Name	PWM/ADC Channel	Description
LLC Resonant + SR	[M1]	PWM-1	PWM-1A	PWM-1A	Half-bridge high-side PWM signal
		PWM-2	PWM-1B	PWM-1B	Half-bridge low-side PWM signal
		PWM-3	PWM-2A	PWM-2A	Rectifier 1 PWM signal (negative half-cycle)
		PWM-4	PWM-3A	PWM-3A	Rectifier 2 PWM signal (positive half-cycle)
		Vo-fb	Vo-fb	ADC-A7	Output voltage sense
		V-SR1	V-SR1	ADC-A2	Rectifier 1 Vds voltage sense (muxed with I-SR1)
		V-SR2	V-SR2	ADC-A4	Rectifier 2 Vds voltage sense (muxed with I-SR2)
		Ip-cs	Ipri-cs	ADC-B1	Resonant tank current sense (rectified)
		I-SR1	I-SR1	COMP1 (ADC-A2)	Rectifier 1 current sense (muxed with V-SR1)
		I-SR2	I-SR2	COMP2 (ADC-A4)	Rectifier 2 current sense (muxed with V-SR1)

1.2 Software Overview

1.2.1 Build Options

In order for you to slowly build up and understand the project, the project is divided into various builds separated by #if options in the HVLLC-Main.c and HVLLC-ISR.asm files. The build used is set by the variable INCR_BUILD in HVLLC-Settings.h. Below is a short description of the different builds available in the HVLLC project.

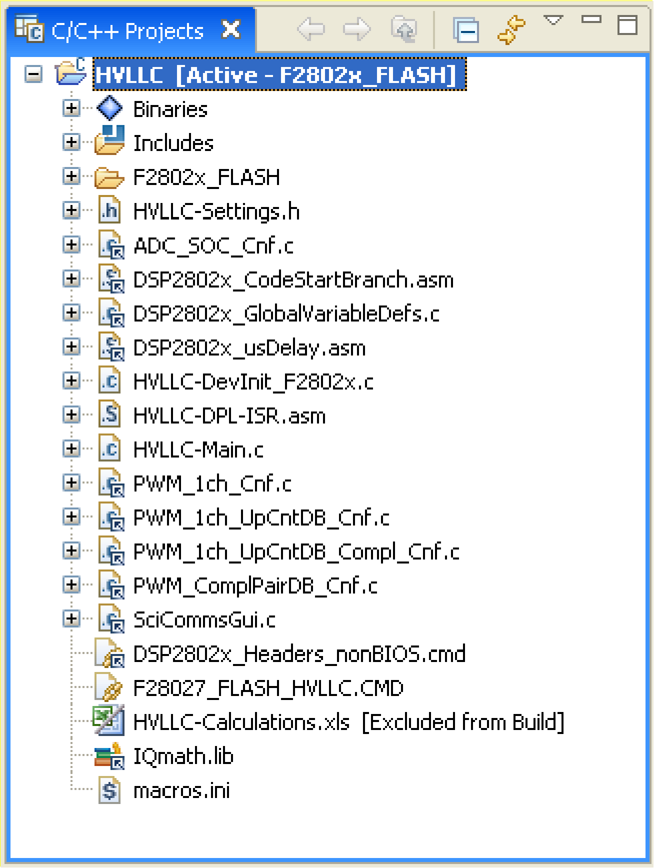
- Build 1: Open Loop operation for checking functionality of the DC/DC stage
- Build 2: Closed Loop operation of the DC/DC stage
- Build 3: Closed Loop operation of the DC/DC stage with analog comparators enabled

The major variables are used in the HVLLC project to control and monitor the half-bridge LLC resonant DC/DC converter with synchronous rectification, and listed in Table 2 along with brief descriptions of each. The primary project files are shown in Figure 2. The Digital Power Library files used are shown in Table 3.

Table 2. Major Variables

LLC_Enable	Enables the half-bridge PWM signals
SR_Enable	Enables the synchronous rectification PWM signals
Comp_Enable (Build 3 only)	Enables the Analog Comparators for current level based SR turn-off
Gui_Vset (Builds 2,3 only)	Sets the closed loop output voltage target value (Q9)
Gui_Vout	Monitors the output voltage (Q9)
Pgain (Builds 2,3 only)	Proportional gain for PID control loop
Igain (Builds 2,3 only)	Integral gain for PID control loop
Dgain (Builds 2,3 only)	Derivative gain for PID control loop
Duty{X}	Sets the duty cycle of the PWM signals; nominally 50%
Period	The period (1/frequency) of the PWMs. In Build 1, this is manually set by the user. In builds 2 and 3, this is set by the control loop.
Min_Period (Builds 2,3 only)	Software enforced minimum PWM period limit
Max_Period (Builds 2,3 only)	Software enforced maximum PWM period limit
RED	Half-bridge PWM rising edge delay (high-side rising edge delay)
FED	Half-bridge PWM falling edge delay (low-side rising edge delay)
REM{X}	SR{X} PWM rising edge margin (rising edge delay)
FEM{X}	SR{X} PWM falling edge margin (falling edge advance)
COMP{X} (Build 3 only)	SR{X} PWM comparator trip value (register counts).

1.2.2 Primary Project Files



C/C++ Projects

HVLLC [Active - F2802x_FLASH]

- Binaries
- Includes
- F2802x_FLASH
- HVLLC-Settings.h
- ADC_SOC_Cnf.c
- DSP2802x_CodeStartBranch.asm
- DSP2802x_GlobalVariableDefs.c
- DSP2802x_usDelay.asm
- HVLLC-DevInit_F2802x.c
- HVLLC-DPL-ISR.asm
- HVLLC-Main.c
- PWM_1ch_Cnf.c
- PWM_1ch_UpCntDB_Cnf.c
- PWM_1ch_UpCntDB_Compl_Cnf.c
- PWM_ComplPairDB_Cnf.c
- SciCommsGui.c
- DSP2802x_Headers_nonBIOS.cmd
- F28027_FLASH_HVLLC.CMD
- HVLLC-Calculations.xls [Excluded from Build]
- IQmath.lib
- macros.ini

Figure 2. HVLLC Project Files

The key framework files used in this project are:

- HVLLC-Main.c – This file is used to initialize, run, and manage the application. This is the “brains” behind the application.
- HVLLC-DevInit_F2802x.c – This file is responsible for initialization and configuration of the device (in this case the F28027), and includes functions for setting up the clocks, PLL, GPIO, and so forth
- HVLLC-ISR.asm – This file contains timing critical “control type” code. This file has an initialization section that is executed one time by the C-callable assembly subroutine _DPL_Init. The file also contains the _DPL_ISR routine which executes at a rate determined by the PWM or timer used to trigger it.
- HVLLC-Settings.h – This file is used to set global definitions for the project (build options). Note that it is linked to both LED-ColorMix-Main.c and LED-ColorMix-ISR.asm.
- HVLLC-Calculations.xls – This spreadsheet that calculates the values of the various scaling factors used in converting Q-value numbers, used by the MCU, to real world values. The variables K_Vout and iK_Vset are examples of scaling factors.

1.2.3 Digital Power Library Files

To reduce the effort for developing code each time, an approach of using Macro Blocks is used. These macro blocks are part of the Digital Power Library and are written in C-callable assembly and can have a C and assembly component. Table 3 shows the list of Digital Power Library files being used in this project.

Table 3. Digital Power Library Files Used

C Configuration Function	ASM Macro
ADC_SOC_Cnf.c	ADCDRV_1ch.asm
None	CNTL_2P2Z.asm
PWM_1ch_Cnf.c	None
PWM_ComplPairDB_Cnf.c	PWMDRV_LLC_ComplPairDB.asm
PWM_1ch_UpCntDB_Cnf.c	PWMDRV_LLC_1ch_UpCntDB.asm
PWM_1ch_UpCntDB_Compl_Cnf.c	PWMDRV_LLC_1ch_UpCntDB_Compl.asm

As the configuration of the peripherals is done in C, it can be easily modified. The ASM driver macro provides the necessary compact code to run in real time. For full details, see the Digital Power Library documentation. The role of each file in the project is described below.

- ADC_SOC_Cnf.c – Used to configure the ADC peripheral as specified.
- ADCDRV_1ch.asm – This macro abstracts the usage of ADC module. The driver macro copies the result from the ADCRegisters into a NetBus array variable
- CNTL_2P2Z.asm – This macro is a second order compensator realized from an IIR filter structure. The five coefficients needed for this function are declared in the C background loop as an array of longs.
- PWM_1ch_Cnf.c – Used to configure PWM4. PWM4 is used to trigger the control loop ISR
- PWM_ComplPairDB_Cnf.c – Used to configure PWM1. PWM1 is used to control the Half-Bridge MOSFETs. PWM1 is also used to trigger the PWM update ISR.
- PWM_1ch_UpCntDB_Cnf.c – Used to configure PWM3. PWM3 is used to control Synchronous Rectifier MOSFET 2.
- PWM_1ch_UpCntDB_Compl_Cnf.c – Used to configure PWM2. PWM2 is used to control Synchronous Rectifier MOSFET 1.
- PWMDRV_LLC_ComplPairDB.asm – Used to update PWM1 registers during operation based on user and/or control-loop inputs.
- PWMDRV_LLC_1ch_UpCntDB.asm – Used to update PWM3 registers during operation based on user and/or control-loop inputs.
- PWMDRV_LLC_1ch_UpCntDB_Compl.asm - Used to update PWM2 registers during operation based on user and/or control-loop inputs.

1.2.4 RAM and CPU Utilization

Table 4. RAM Memory Usage of the Final Project

Build Level	Program Memory Usage 2802x	Data Memory Usage 2802x [1]
Build 3 (FLASH)	220 words	499 words

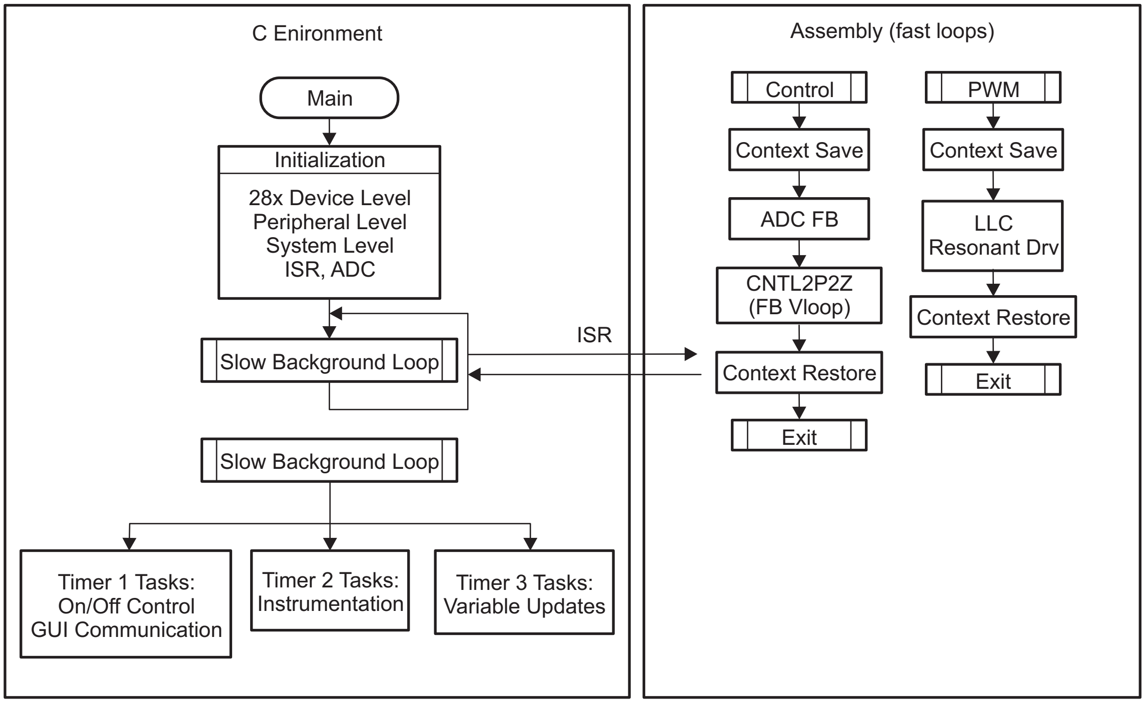
[1] Excluding the stack size

Table 5. CPU Utilization of the Final Project

Name of Macros	Number of Cycles
Control Loop ISR (100 kHz)	82
Context Save, Restore, ISR management, and so forth	27
ADCDRV_1ch	6
CNTL_2P2Z	47
PWM Update ISR (100 kHz to 130 kHz)	121
Context Save, Restore, ISR management, and so forth	27
PWMDRV_LLC_ComplPairDB	26
PWMDRV_LLC_1ch_UpCntDB	38
PWMDRV_LLC_1ch_UpCntDB_Compl	31
Total Number of Cycles	**203**
CPU Utilization @ 60 Mhz (at 100 kHz)	**33.83%**

Table 6. System Features

Development/Emulation	Code Composer Studio™ v4.1 (or above) With Real-Time Debugging
Target Controller	TMS320F2802x
PWM Frequency	PWM1,PWM2,PWM3 – Variable 100 kHz to 130 kHz PWM4 – Fixed 100 kHz
Interrupts	Control Loop ISR – Fixed 100 kHz. Triggered by EPWM4 PWM Update ISR – Variable 100 kHz to 130 kHz. Triggered by EPWM1



2 Hardware Setup

Some of the major connectors and features of the half-bridge LLC resonant DC/DC conversion with synchronous rectification board are listed in Figure 3. Each component is named first with their macro number, followed by the reference name. For example, [M2]-J1 would refer to the jumper J1 located in the macro M2 and [Main]-J1 would refer to the J1 located on the board outside of the other defined macro blocks. For full details, see the Hardware Guide documentation.

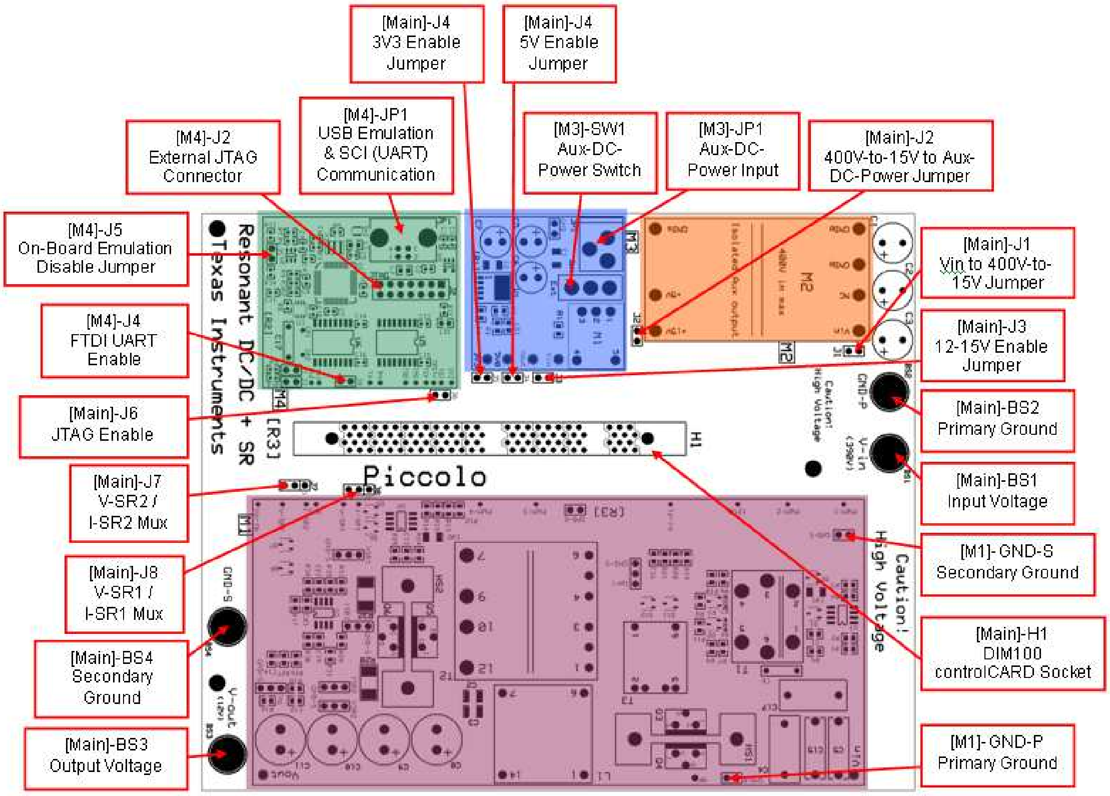

[Main]-J4 3V3 Enable Jumper
[Main]-J4 5V Enable Jumper
[M4]-JP1 USB Emulation & SCI (UART) Communication
[M3]-SW1 Aux-DC-Power Switch
[M3]-JP1 Aux-DC-Power Input
[M4]-J2 External JTAG Connector
[Main]-J2 400V-to-15V to Aux-DC-Power Jumper
[M4]-J5 On-Board Emulation Disable Jumper
[Main]-J1 Vin to 400V-to-15V Jumper
[M4]-J4 FTDI UART Enable
[Main]-J3 12-15V Enable Jumper
[Main]-J6 JTAG Enable
[Main]-BS2 Primary Ground
[Main]-J7 V-SR2 / I-SR2 Mux
[Main]-BS1 Input Voltage
[M1]-GND-S Secondary Ground
[Main]-J8 V-SR1 / I-SR1 Mux
[Main]-H1 DIM100 controlCARD Socket
[Main]-BS4 Secondary Ground
[Main]-BS3 Output Voltage
[M1]-GND-P Primary Ground
Resonant DC/DC + SR
Texas Instruments
Piccolo
M1
M2
M3
M4
H1
Caution! High Voltage
GND-P
V-in (390V)
GND-S
V-out (12V)

Figure 3. Hardware Features

Perform or verify the following steps to prepare the high-voltage half-bridge LLC resonant DC/DC converter with synchronous rectification kit for use:

1. Insert a F28027 control card into socket [Main]-H1.
2. Connect your computer to the board using a USB cable. [M4]-LD1 near the USB connector [M4]-JP1 should turn on.
3. Verify the following jumper settings:
   (a) No jumpers are placed on [Main]-J1,J2.
   (b) Jumpers are placed on [Main]-J3, J4, J5,J6.
   (c) Jumpers are placed on pins 1-2 on [Main]-J7,J8.
   (d) A jumper is placed on [M4]-J4.
4. Connect the 12 VDC power supply to [M3]-JP1 to power the auxiliary power rail.
5. Connect a 390 VDC, 1A (max), power supply across [Main]-BS1, BS2.
6. Connect a 300W (max) load across [Main]-BS3, BS4.
7. Set the power switch [M3]-SW1 so that it is pointed towards the “Ext” label. [M3]-LD1 should turn on and a green LED should turn on the controlCARD.

NOTE: If Code Composer Studio has never been installed, it may be necessary to install drivers to make the board work correctly. If a popup comes up when the USB cable is connected from the board to the computer, have the install wizard install drivers from the XDS100v1 directory of the USB drive included with this kit.

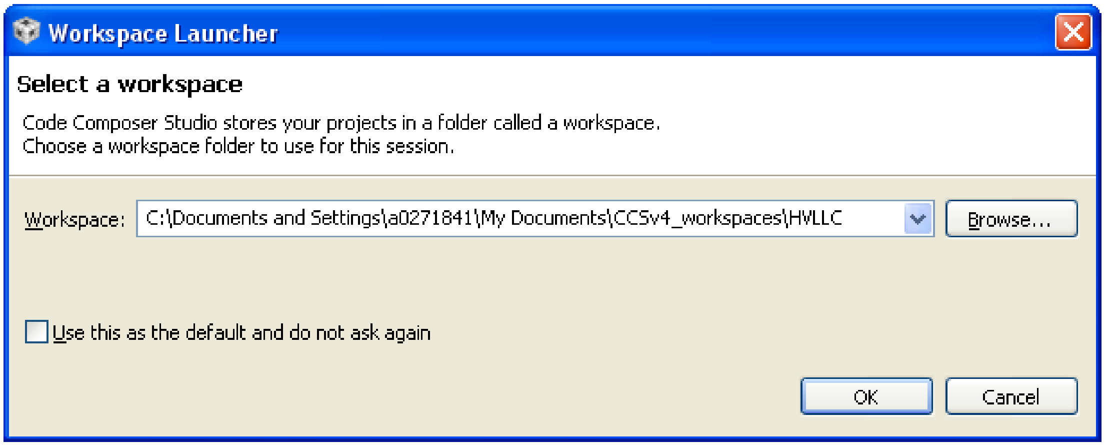
3 Software Setup

3.1 Installing Code Composer and controlSUITE™

1. Install Code Composer v4.x from the USB drive included in the kit, if not already installed.
2. Go to http://www.ti.com/controlsuite and run the controlSUITE installer. Select to install the "Multi-DC/DC Color LED Kit" software and allow the installer to also download all automatically checked software.

3.2 Setup Code Composer Studio to Work With the Kit

1. Open "Code Composer Studio v4".
2. Once Code Composer Studio opens, the workspace launcher may appear that would ask to select a workspace location, (note that workspace is a location on the hard drive where all the user settings for the IDE are saved (for example, which projects are open, what configuration is selected, where they are saved, and so forth). This can be anywhere on the disk, the location shown in Figure 4 is just for reference. Also note that if this is not your first-time running Code Composer this dialog may not appear.
   (a) Click the "Browse…" button
   (b) Create the path below by making new folders as necessary
   (c) "C:\Documents and Settings\<username>\My Documents\CCSv4_workspaces\HVLLC"
   (d) Uncheck the box that says "Use this as the default and do not ask again"
   (e) Click "OK"



Workspace Launcher

Select a workspace

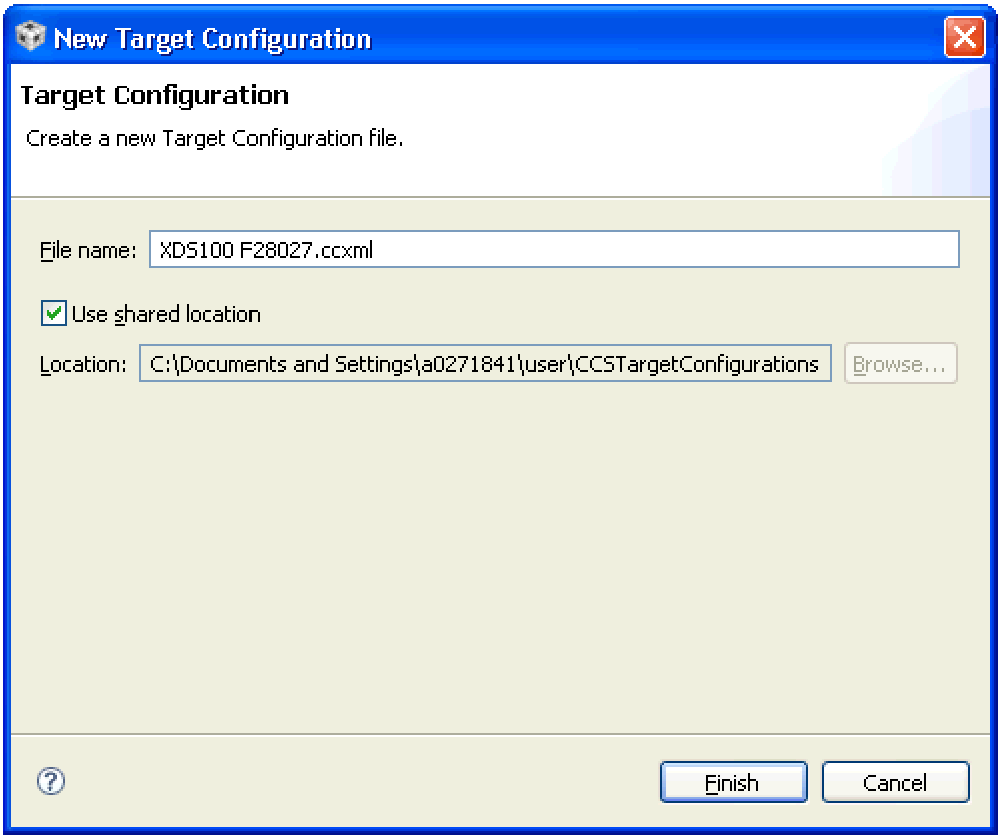
Code Composer Studio stores your projects in a folder called a workspace.
Choose a workspace folder to use for this session.

Workspace: C:\Documents and Settings\a0271841\My Documents\CCSv4_workspaces\HVLLC

☐ Use this as the default and do not ask again

Figure 4. Workspace Launcher

3. Configure Code Composer to know which MCU it will be connecting to. Click “Target → New Target Configuration…”. Name the new configuration “XDS100 F28027.ccxml”. Make sure that the “Use shared location” checkbox is checked and then click “Finish”.



New Target Configuration

Target Configuration

Create a new Target Configuration file.

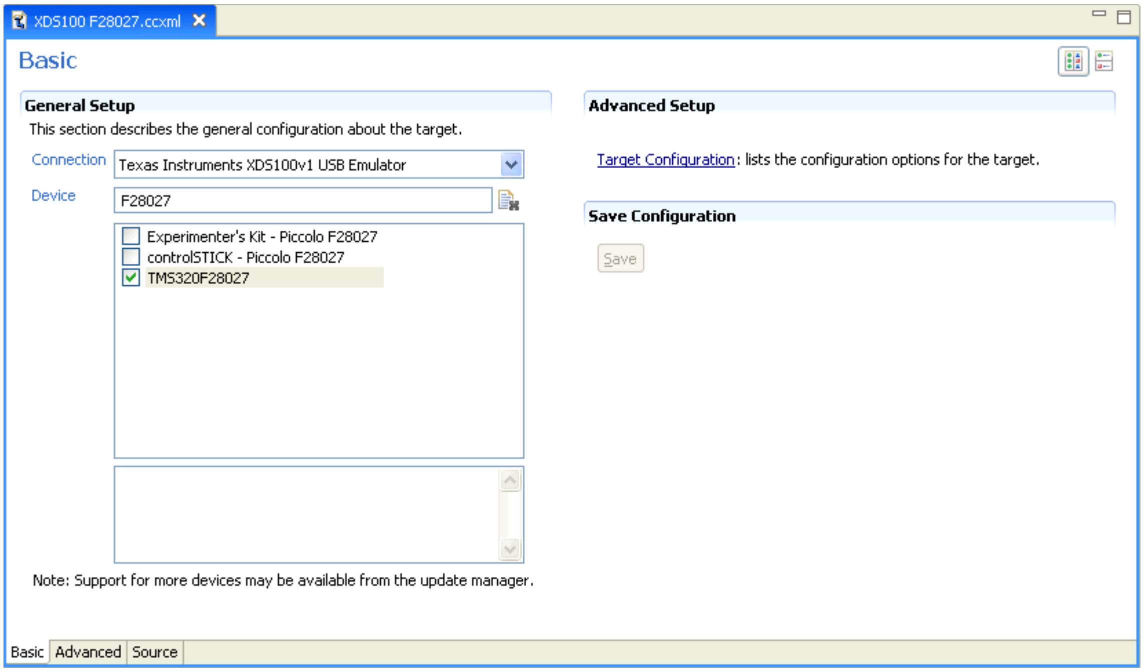
File name: XDS100 F28027.ccxml

☑ Use shared location

Location: C:\Documents and Settings\a0271841\user\CCSTargetConfigurations

Figure 5. Creating a Target Configuration

4. This should open up a new tab as seen in Figure 5. Select and enter the options as shown:
   (a) Connection – Texas Instruments XDS100v1 USB Emulator
   (b) Device – TMS320F28027
   (c) Click Save
   (d) Close the "XDS100 F28027.ccxml" tab



XDS100 F28027.ccxml

Basic

General Setup

This section describes the general configuration about the target.

Connection: Texas Instruments XDS100v1 USB Emulator

Device: F28027

- ☐ Experimenter's Kit - Piccolo F28027
- ☐ controlSTICK - Piccolo F28027
- ☑ TMS320F28027

Note: Support for more devices may be available from the update manager.

Advanced Setup

Target Configuration: lists the configuration options for the target.

Save Configuration

Figure 6. Configuring a New Target

5. Assuming this is your first time using Code Composer, the “XDS100 F28027” configuration is now set as the default target configuration for Code Composer. Check this by going to “View → Target Configurations”. In the “User Defined” section, right-click on the “XDS00 F28027.ccxml” file and select “Set as Default”. This tab also allows you to reuse existing target configurations and link them to specific projects.

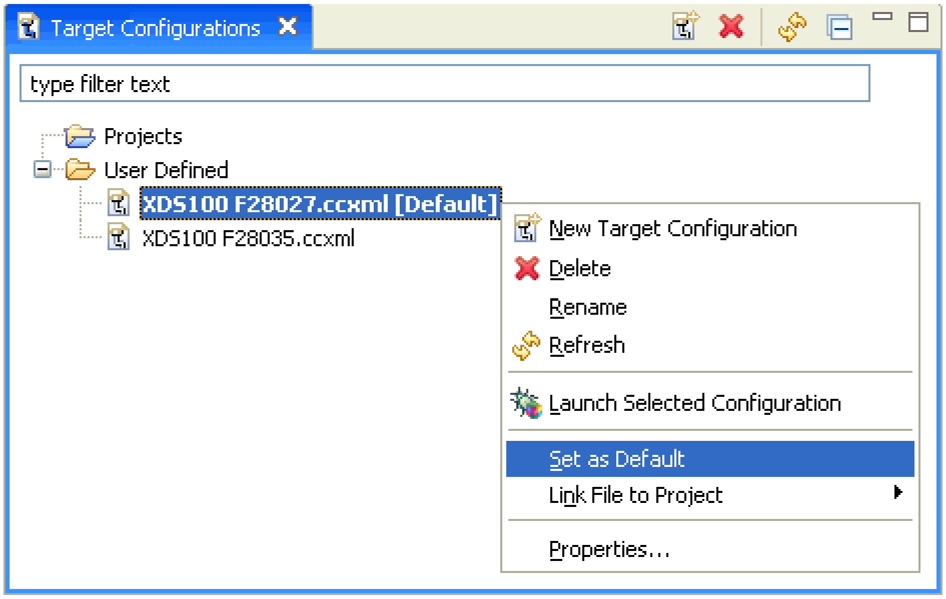
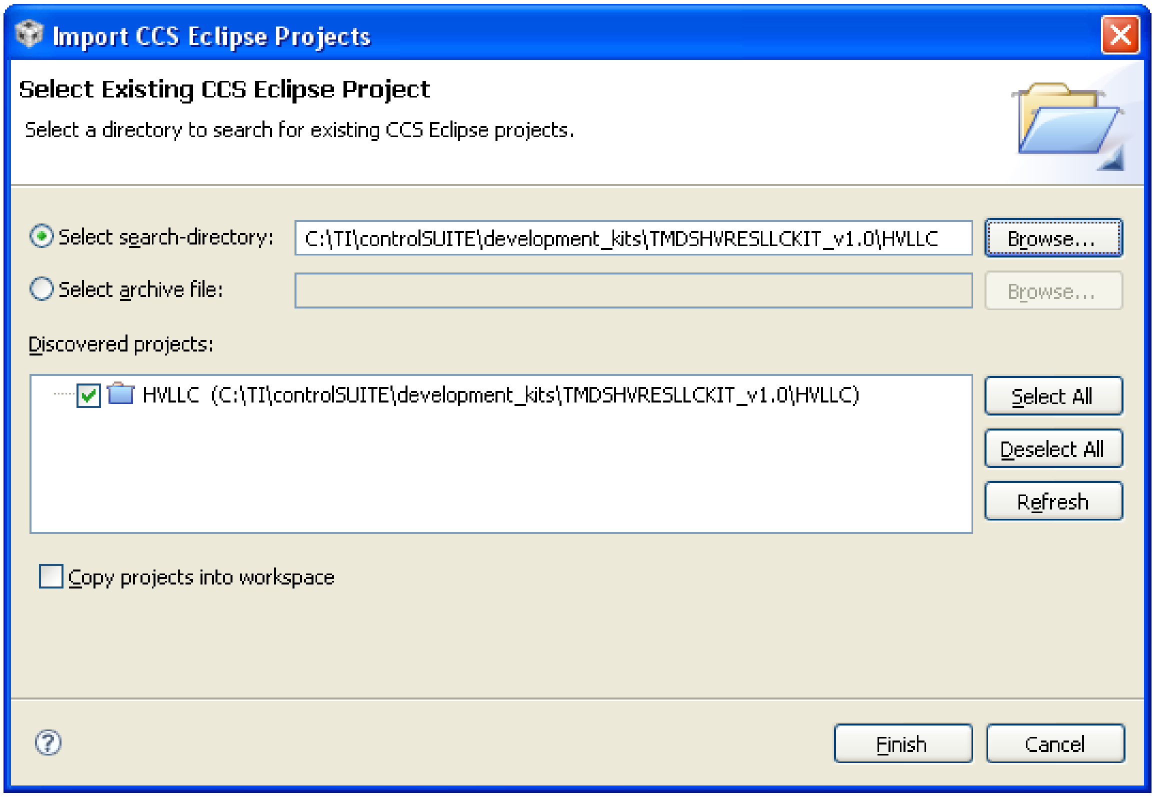


Figure 7. Setting a Default Target Configuration

6. Add the project into your current workspace by clicking “Project → Import Existing CCS/CCE Eclipse Project”:
   (a) Browse to the project directory within the kit folder. The default location is:
   `C:\TI\controlSUITE\development_kits\TMDSHVRESLLCKIT_v1.0\HVLLC`
   (b) Click “Finish” to load the project into the workspace.



Import CCS Eclipse Projects

Select Existing CCS Eclipse Project

Select a directory to search for existing CCS Eclipse projects.

● Select search-directory: C:\TI\controlSUITE\development_kits\TMDSHVRESLLCKIT_v1.0\HVLLC

○ Select archive file:

Discovered projects:

☑ HVLLC (C:\TI\controlSUITE\development_kits\TMDSHVRESLLCKIT_v1.0\HVLLC)

□ Copy projects into workspace

Figure 8. Importing the Project Into Your Workspace

7. The HVLLC project should be set as the active project. Right-click on the project name and click “Set as Active Project”. Expand the file structure of the project.

3.3 *Incremental System Builds*

In order for you to slowly build up and understand the project, the project is divided into various builds separated by #if options in the HVLLC-Main.c and HVLLC-ISR.asm files. The build used is set by the variable INCR_BUILD in HVLLC-Settings.h. Below is a short description of the different builds available in the HVLLC project.

- Build 1: Open loop operation for checking functionality of the DC/DC stage
- Build 2: Closed loop operation of the DC/DC stage
- Build 3: Closed loop operation of the DC/DC stage with analog comparators enabled

4 Build 1: Open Loop Operation for Checking Functionality

NOTE: This section assumes that the Hardware Setup and Software Setup sections have been completed, go through those sections before continuing.

The objective of this build is as follows:

- Verify that the PWM and ADC signals are working properly
- Verify that the power stage on the board is working correctly
- Manually control the period (1/frequency) of the power stage on the board and evaluate its output

The components of the system as used in the software are described in Figure 9.

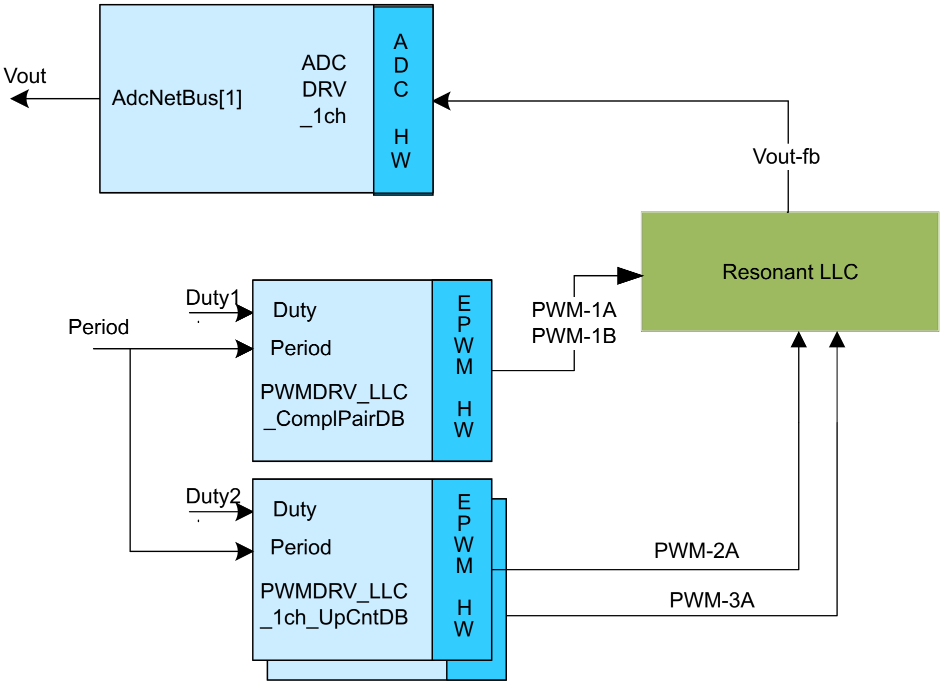


Figure 9. Build Level 1 Block Diagram

4.1 *Inspect the Project*

- After initial boot processes are complete, the software begins from the main function. Open up HVLLC-Main.c and find the main() function (line 241).
- The first thing that the software does in the main() function is call a function called DeviceInit() found in HVLLC-DevInit_F2802x.c. Open and inspect HVLLC-DevInit_F2802x.c by double clicking on the filename in the project window. In this file, the various peripheral clocks are enabled or disabled and the functional pinout, that configures which peripherals come out of which pins, is defined.
  - Confirm that the ADC and PWM1-4 peripheral clocks are enabled (lines 99-117). Also, confirm that GPIO00-GPIO07 are configured to be PWM outputs (lines 136-182).
- The project is provided with incremental builds where different components and macro blocks of the system are pieced together one by one to form the entire system. This helps in step-by-step debug and understanding of the system.
  - From the C/C++ Project tab, open the file *HVLLC-Settings.h* and make sure that INCR_BUILD is set to 1, and save this file. After you test build 1, this variable needs to be redefined to move on to build 2, and so on until all builds are complete.

- Go back to the HVLLC-Main.c file. Notice the various incremental build configuration code. The Build LEVEL1 configuration code is found on lines 478-493. In it, the ADC, PWM, and macro block connections are configured. Note that if INCR_BUILD is set to 1, lines 496-539 are ignored by the compiler because they do not belong in the current build. A similar method of enabling or disabling code based on the value of INCR_BUILD is done in HVLLC-ISR.asm as well.

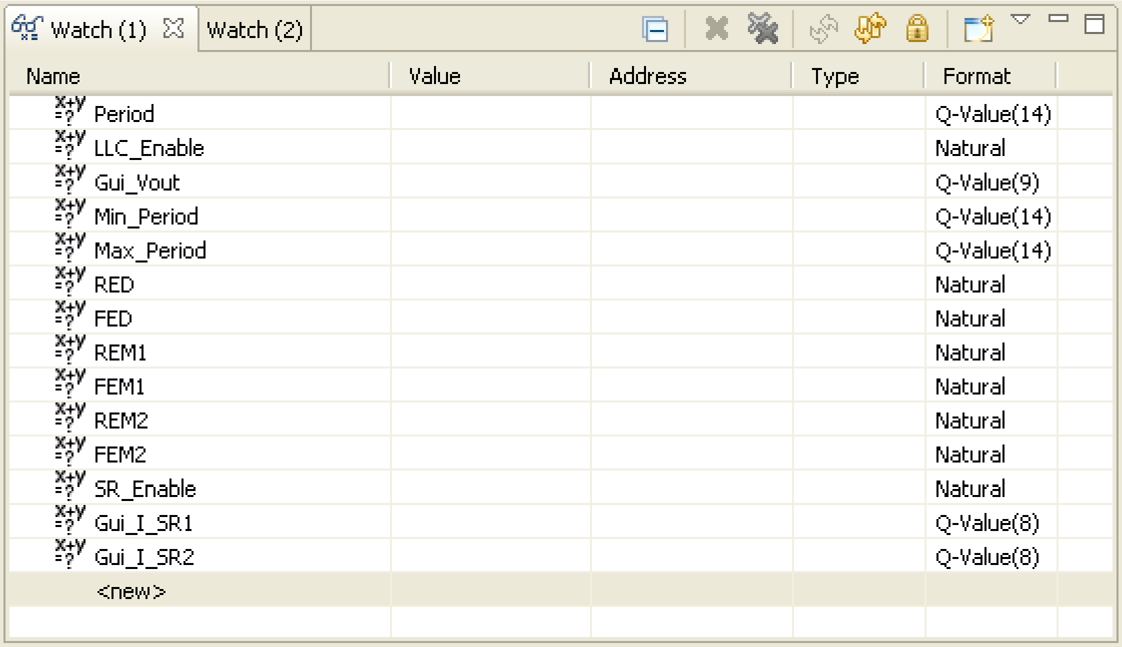
4.2 Build and Load the Project

1. Right Click on the Project Name and click on “Rebuild Project” and watch the Console window. Any errors in the project will be displayed in the Console window.
2. On successful completion of the build click the “Debug” button , located in the top-left side of the screen. The IDE will now automatically connect to the target, load the output file into the device and change to the Debug perspective.
3. Click the real-time mode button that says “Enable silicon real-time mode”. This allows you to edit and view variables in real time without halting the program.
4. A message box may appear. If so, select YES to enable debug events. This sets bit 1 (DGBM bit) of the status register 1 (ST1) to a “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.

4.3 Setup Watch Window and Graphs

1. Click: View → Watch on the menu bar to open a watch window to view the variables being used in the project. Add the variables found in Figure 10 to the watch window. The format of a variable can be changed by right-clicking on a particular variable then selecting a Q-Value. Change the format of each variable to match the figure. The variables in this watch window are used to control and monitor the status of the board while in Build 1.

   Hint: Shift-Click can be used to select multiple variables or elements and then right-clicking and changing the format will affect all variables selected.



Name	Value	Address	Type	Format
Period				Q-Value(14)
LLC_Enable				Natural
Gui_Vout				Q-Value(9)
Min_Period				Q-Value(14)
Max_Period				Q-Value(14)
RED				Natural
FED				Natural
REM1				Natural
FEM1				Natural
REM2				Natural
FEM2				Natural
SR_Enable				Natural
Gui_I_SR1				Q-Value(8)
Gui_I_SR2				Q-Value(8)
<new>				

Figure 10. Configuring the Watch Window for Build 1

2. Click on the Continuous Refresh button in the watch window. This enables the window to run with

real-time mode.

4.4 Run the Code

1. Run the code by pressing the Run Button in the Debug Tab. The project should now run, and the values in the watch window should keep on updating.
2. Verify the PWM and ADC circuitry with the primary power supply set to 0V or turned off:
   (a) Verify that all voltage and current feedback variables are close to 0. They may not be exactly 0.0 as it is normal for some low level noise to be present on the ADCs.
   (b) Enable the PWMs by setting “LLC_Enable” and “SR_Enable” to 1. Use an oscilloscope to verify the PWM waveforms. Adjust the “Period” value and verify that the frequency of the PWM waveforms change.
3. Set the load to a minimum of 1 A.
4. Ramp the primary power supply up to 390 VDC and verify that voltage is present on the output.
5. Carefully adjust “Period” and verify that “Gui_Vout” increases when you decrease “Period” and that it decreases when you increase “Period”.
6. Carefully adjust “RED” and “FED” to see how they affect the half-bridge PWM signals.
7. Carefully adjust “REM{X}” and “FEM{X}” to see how they affect the rectifier PWM signals.
8. Once complete, turn off or set the primary power supply to 0 VDC.
9. Disable the real-time mode and Reset the processor (Target → Reset → Reset CPU) and then terminate the debug session by clicking (Target → Terminate All). This will disconnect Code Composer Studio from the MCU.

5 Build 2: Closed Loop Operation

The objective of this build is as follows:

- Regulate the output voltage with closed-loop feedback using PID control.

The components of the system as used in the software are described in Figure 11.

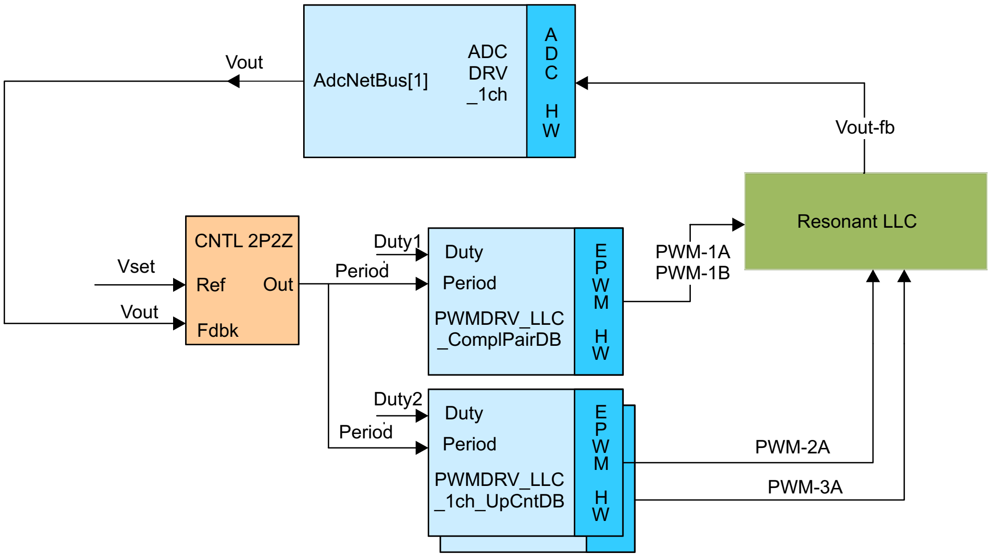


Figure 11. Build Level 2 Block Diagram

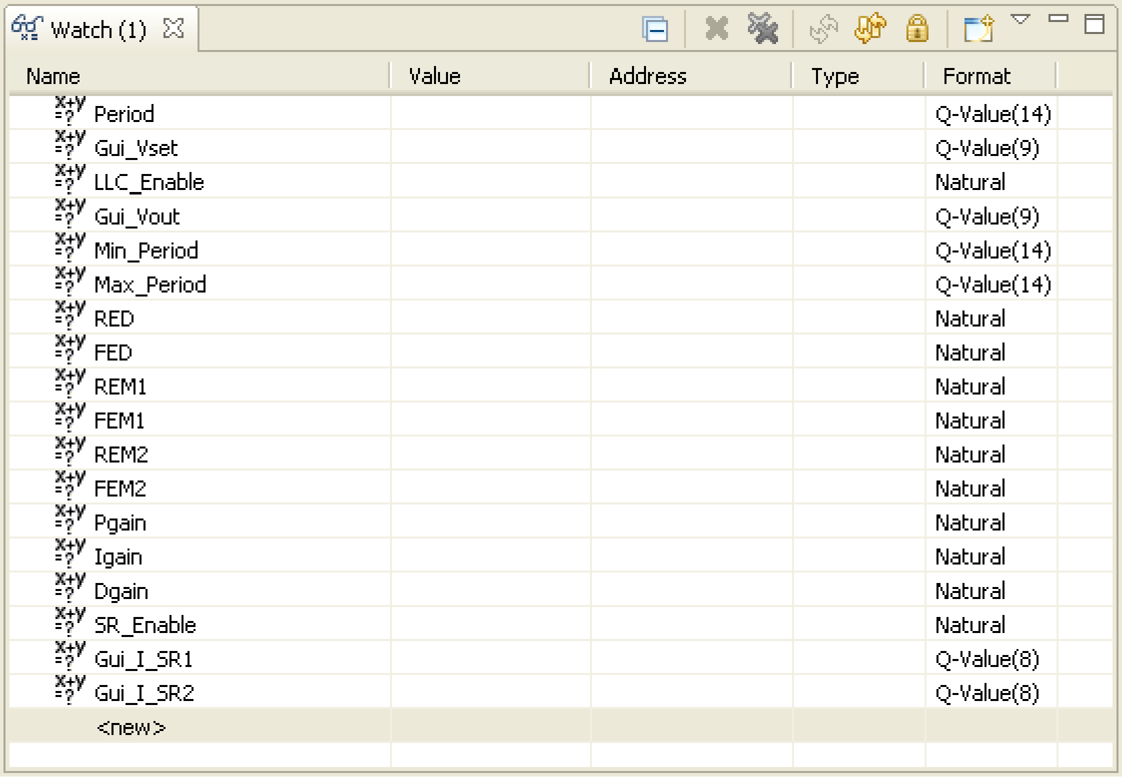
The instructions for this build are more succinct than in the first build. For further guidance, see the instructions in Section 4. Please run build 1 prior to running build level 2.

5.1 Build and Load the Project

1. Open *HVLLC-Settings.h* and change the incremental build level to 2 (#define INCR_BUILD 2). Save the file.
2. Right-click on the project name and select “Rebuild Project”.
3. Click the “Debug” button on successful completion of the build, located in the top-left side of the screen. The IDE automatically connects to the target, loads the output file into the device, and changes to the Debug perspective.
4. Click the real-time mode button that says “Enable silicon real-time mode”. This allows you to edit and view variables in real time without halting the program.

5.2 Setup Watch Window and Graphs

1. Add a new watch window to the workspace; add the variables and set them to use the correct format as shown in Figure 12. The variables in this watch window are used to control and monitor the status of the board while in Build 2.



Name	Value	Address	Type	Format
Period				Q-Value(14)
Gui_Vset				Q-Value(9)
LLC_Enable				Natural
Gui_Vout				Q-Value(9)
Min_Period				Q-Value(14)
Max_Period				Q-Value(14)
RED				Natural
FED				Natural
REM1				Natural
FEM1				Natural
REM2				Natural
FEM2				Natural
Pgain				Natural
Igain				Natural
Dgain				Natural
SR_Enable				Natural
Gui_I_SR1				Q-Value(8)
Gui_I_SR2				Q-Value(8)
<new>				

Figure 12. Configuring the Watch Window for Build 2

2. Click on the Continuous Refresh button in the watch window.

5.3 Run the Code

1. Run the code by pressing the Run Button in the Debug Tab.
2. Enable the PWMs by setting “LLC_Enable” and “SR_Enable” to 1.
3. Set the load to a minimum of 1 A.
4. Ramp the primary power supply up to 390 VDC.
5. Set “Gui_Vset” to 12 VDC. “Period” should change automatically and “Gui_Vout” should change to 12 VDC to match “Gui_Vset”.
6. Experiment with changing the load. Notice that “Period” (1/frequency) increases with the increasing load and decreases with the decreasing load.
7. Once complete, turn off or set the primary power supply to 0 VDC.
8. Disable real-time mode and Reset the processor (Target->Reset->Reset CPU) and then terminate the debug session by clicking (Target->Terminate All).

6 Build 3: Closed Loop Operation With Analog Comparators Enabled

The objective of this build is as follows:

- Use the analog comparators to dynamically adjust the SR PWM turn-off timing based on SR current levels.

The components of the system as used in the software are described in Figure 13:

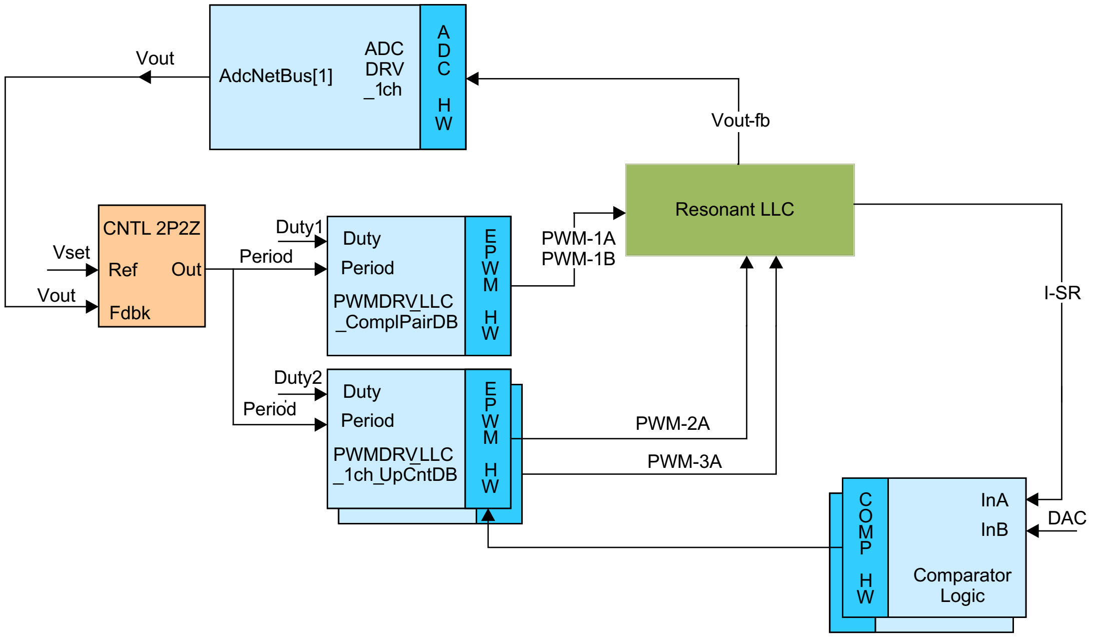


Figure 13. Build Level 3 Block Diagram

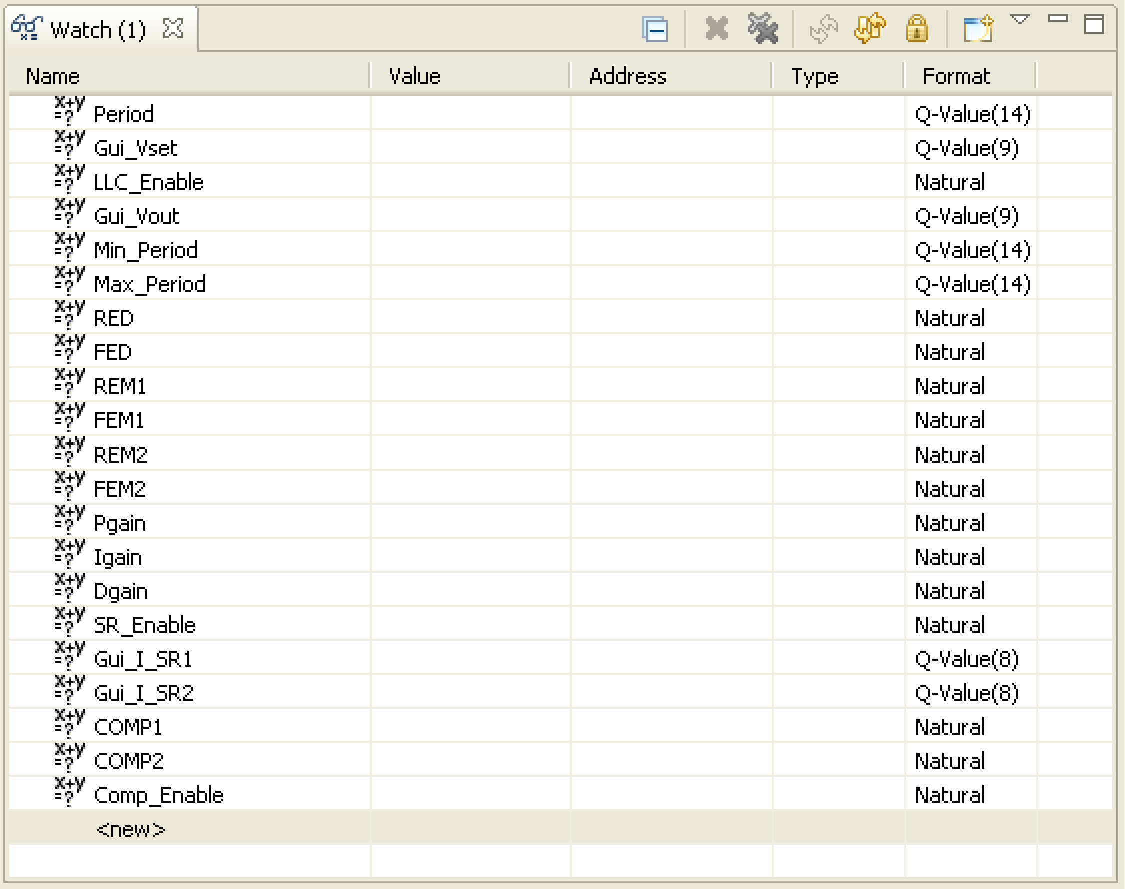
The instructions for this build will be more succinct than in the first build. For further guidance, see the instructions in Section 5. Please run build 2 prior to running build level 3.

6.1 Build and Load the Project

1. Open HVLLC-Settings.h and change the incremental build level to 3 (#define INCR_BUILD 3). Save the file.
2. Right-click on the project name and select “Rebuild Project”.
3. Click the “Debug” button on successful completion of the build, located in the top-left side of the screen. The IDE automatically connects to the target, loads the output file into the device, and changes to the Debug perspective.
4. Click the real-time mode button that says “Enable silicon real-time mode”. This allows you to edit and view variables in real time without halting the program.

6.2 Setup Watch Window and Graphs

1. Add a new watch window to the workspace; add the variables and set them to use the correct format as shown in Figure 14. The variables in this watch window are used to control and monitor the status of the board while in Build 3.



Name	Value	Address	Type	Format
Period				Q-Value(14)
Gui_Vset				Q-Value(9)
LLC_Enable				Natural
Gui_Vout				Q-Value(9)
Min_Period				Q-Value(14)
Max_Period				Q-Value(14)
RED				Natural
FED				Natural
REM1				Natural
FEM1				Natural
REM2				Natural
FEM2				Natural
Pgain				Natural
Igain				Natural
Dgain				Natural
SR_Enable				Natural
Gui_I_SR1				Q-Value(8)
Gui_I_SR2				Q-Value(8)
COMP1				Natural
COMP2				Natural
Comp_Enable				Natural
<new>				

Figure 14. Configuring the Watch Window for Build 3

2. Click on the Continuous Refresh button in the watch window.

6.3 Run the Code

1. Run the code by pressing the Run Button in the Debug Tab.
2. Enable the PWMs by setting “LLC_Enable” and “SR_Enable” to 1.
3. Use an oscilloscope to probe the SR PWM and SR Current signals.
4. Set the load to a minimum of 1 A.
5. Ramp the primary power supply up to 390 VDC.
6. Set “Gui_Vset” to 12 VDC.
7. Set the load to 10 A.
8. On the oscilloscope, make a note of where the SR PWM signal’s falling edge is relative to where the SR Current drops back to 0.
9. Set “Comp_Enable” to 1. Observe how the SR PWM signal’s falling edge has shifted. The SR PWM signal’s falling edge should now be closer to where the SR Current drops back to 0.
10. Experiment with changing the load. Notice that the SR PWM signal’s falling edge tracks where the SR Current drops back to 0.
11. Experiment with changing the “COMP{X}” value. This affects the current level at which the analog comparator will allow the SR PWM to turn-off.
12. Once complete, turn off or set the primary power supply to 0 VDC.
13. Disable real-time mode and Reset the processor (Target->Reset->Reset CPU) and then terminate the debug session by clicking (Target->Terminate All).

NOTE: The analog comparators is used to delay the SR PWM’s falling edge and will not advance the falling edge beyond that specified by “FEM{X}”.

6.4 Experimental Results

This section contains various experimental results for use as references while experimenting with the high-voltage half-bridge LLC resonant DC/DC converter kit.

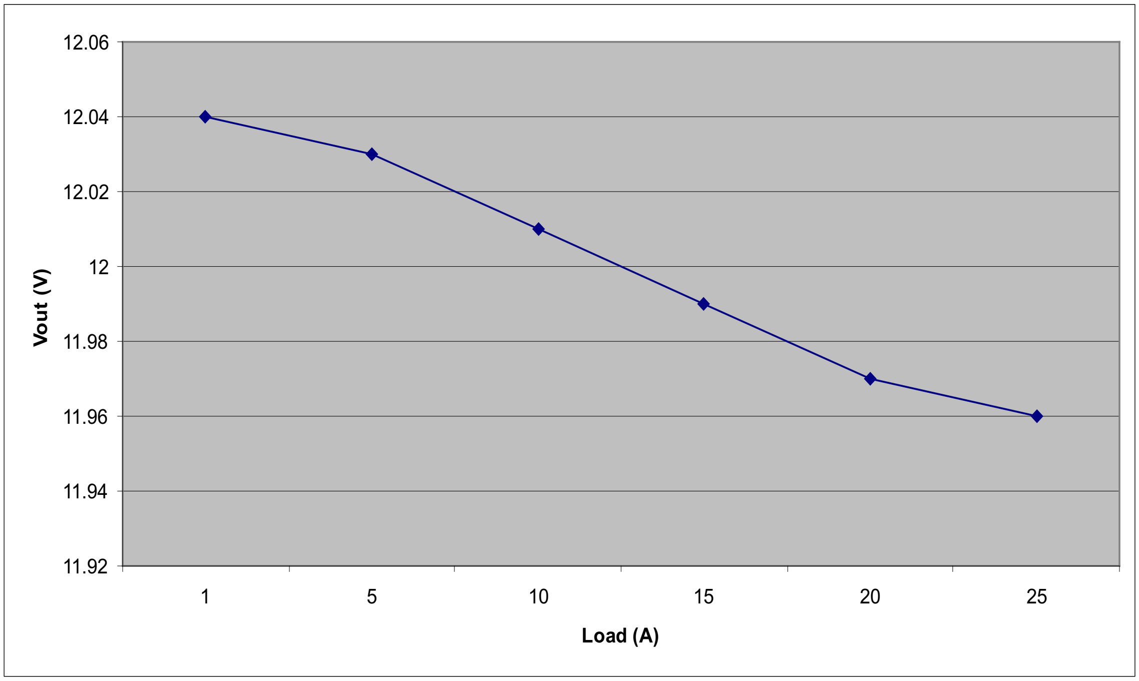


Figure 15. Output Voltage: Load Regulation (V_{IN} = 390 VDC)

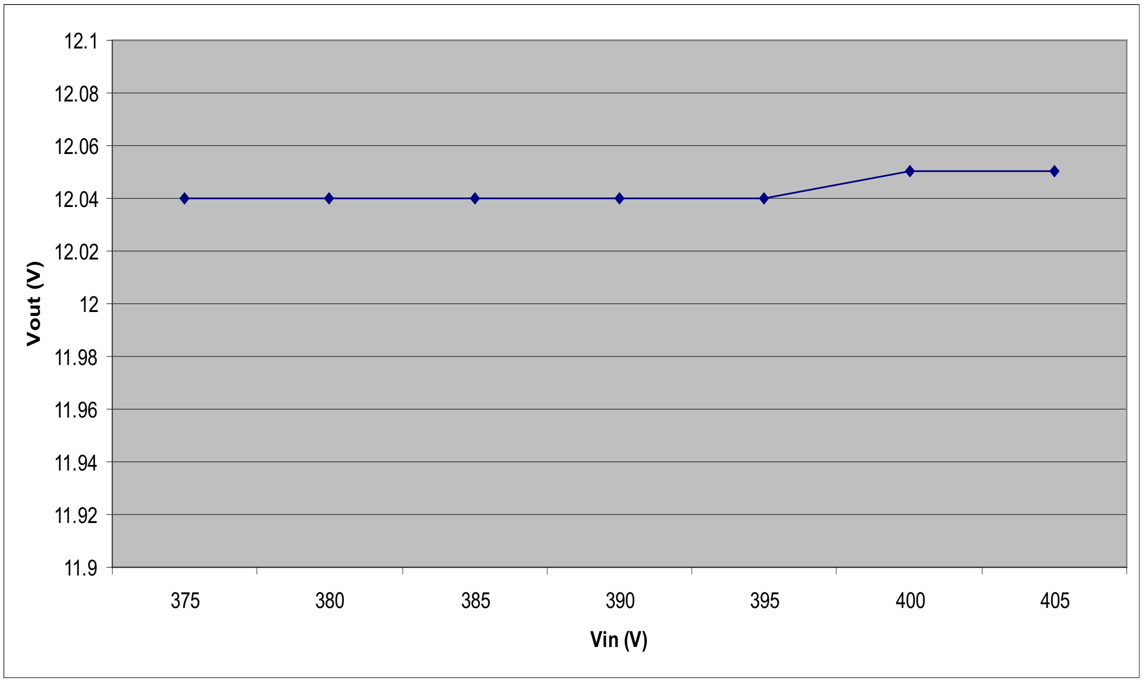


Figure 16. Output Voltage: Line Regulation (I_O = 1 A)

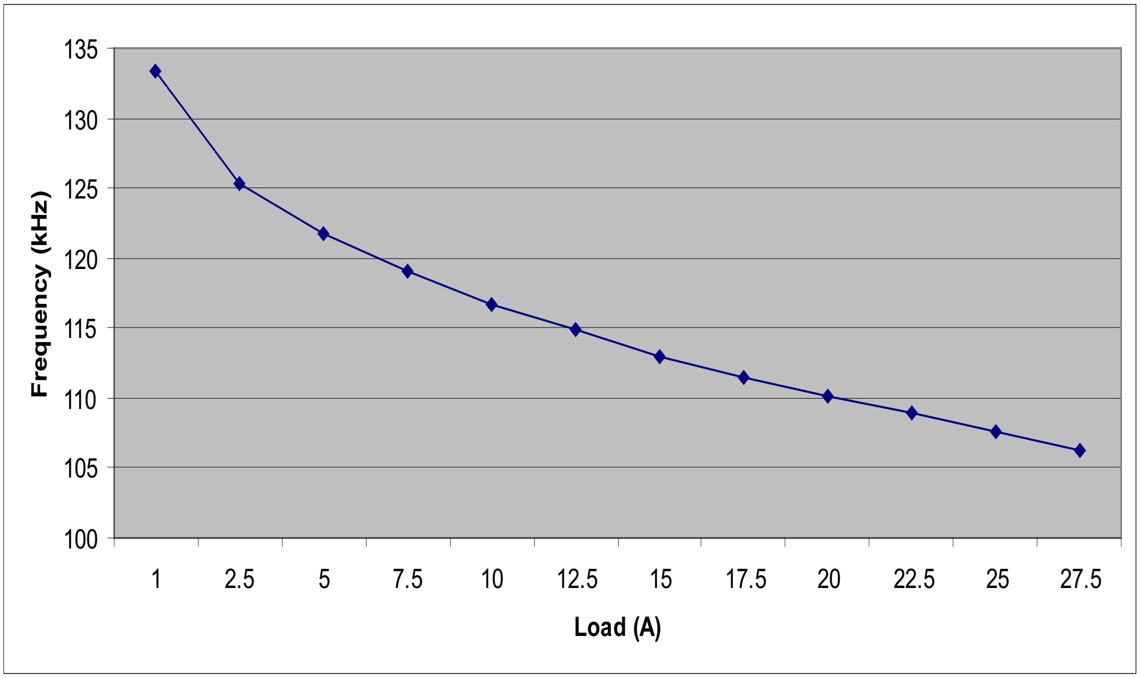


Figure 17. Switching Frequency vs Load (V_{IN} = 390 VDC)

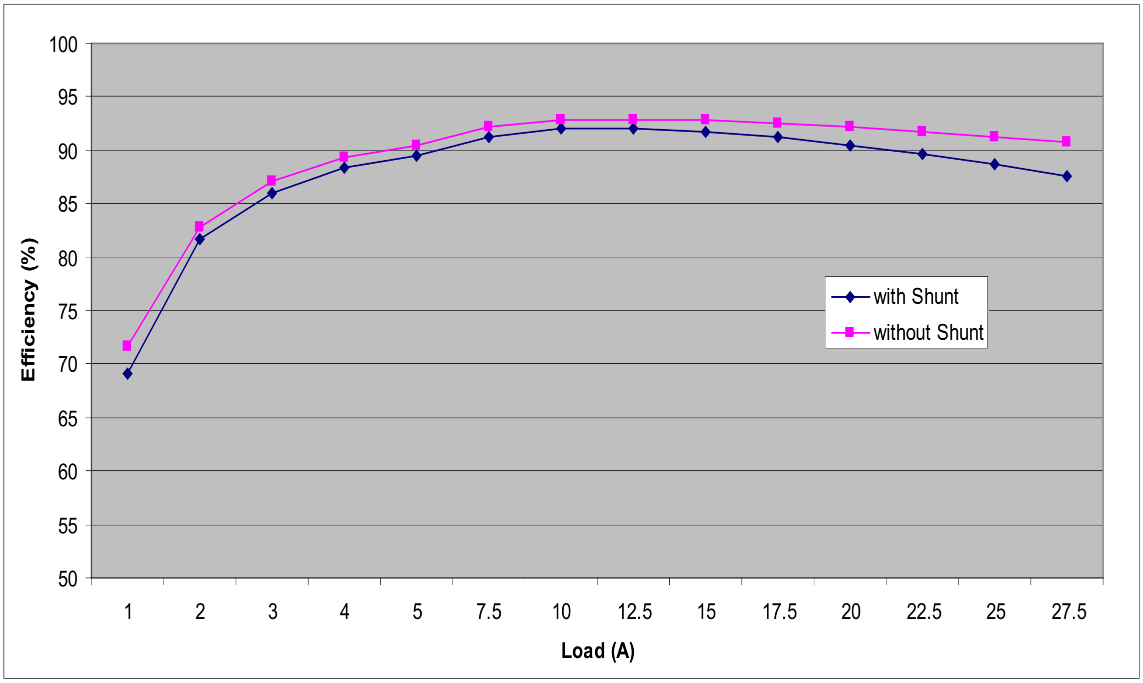


Figure 18. Efficiency vs Load (V_{IN} = 390 VDC)

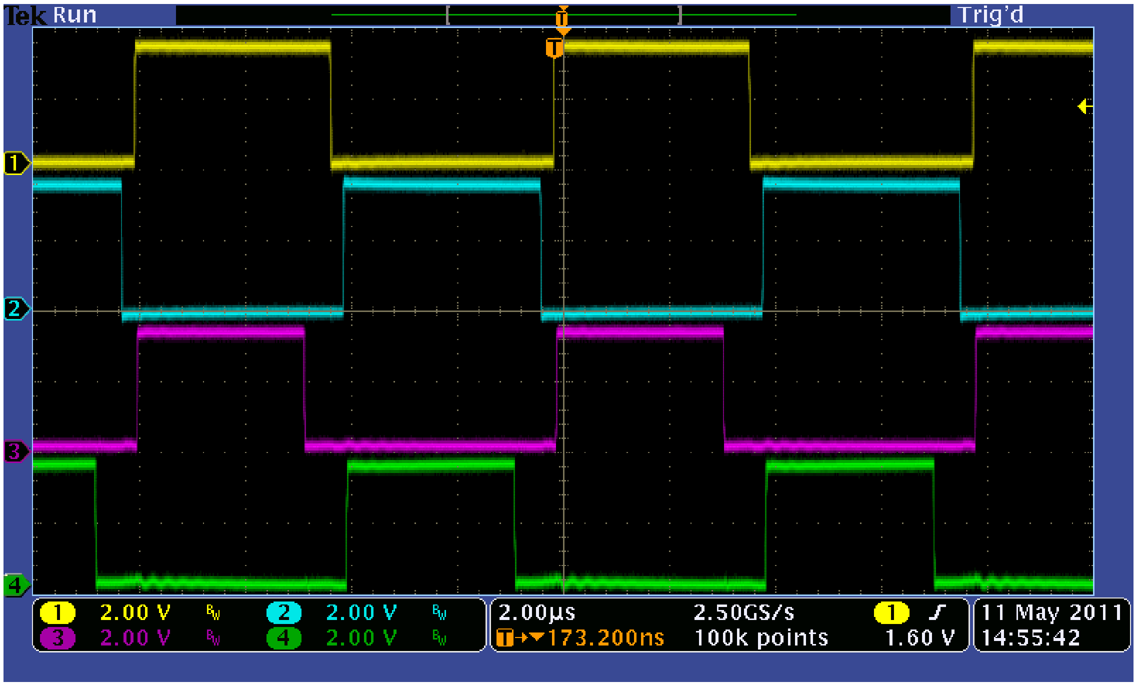


Figure 19. PWM Timings Example (Yellow – HB HS, Blue – HB LS, Purple – SR2, Green – SR1)

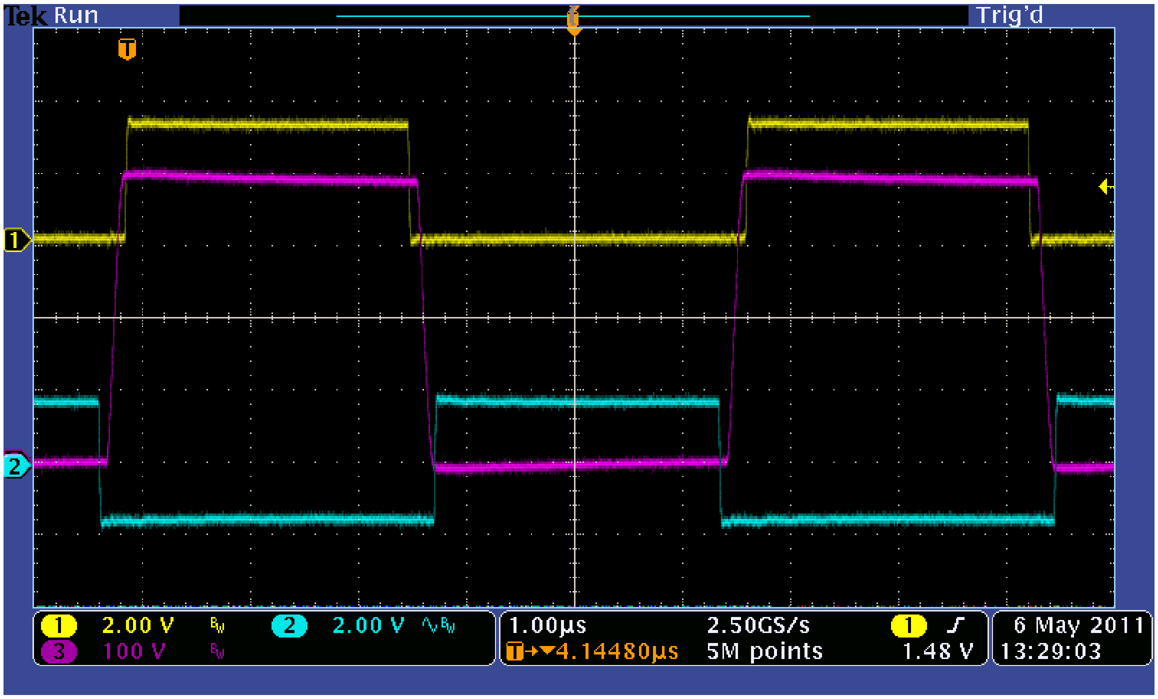


Figure 20. Half-Bridge Zero Voltage Switching (Yellow – HB HS PWM, Blue – HB LS PWM, Purple – MP Voltage)

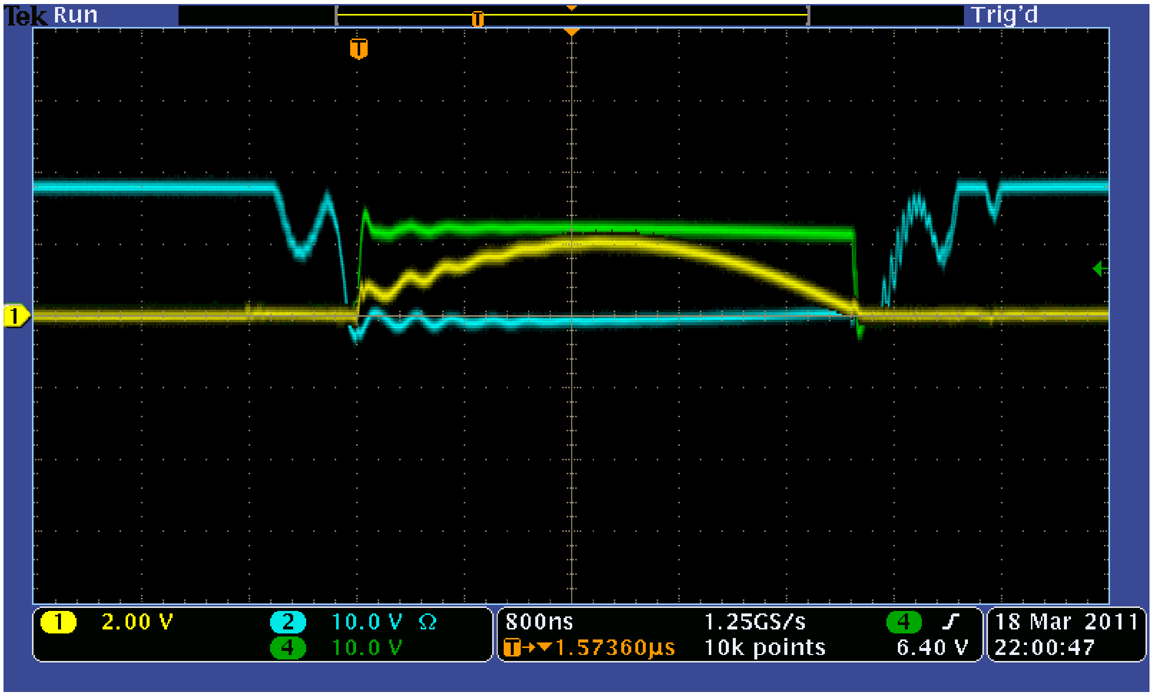


Figure 21. Rectifier Zero Current Switching (Yellow – SR Current, Blue – SR Mosfet Vds, Green – SR PWM)

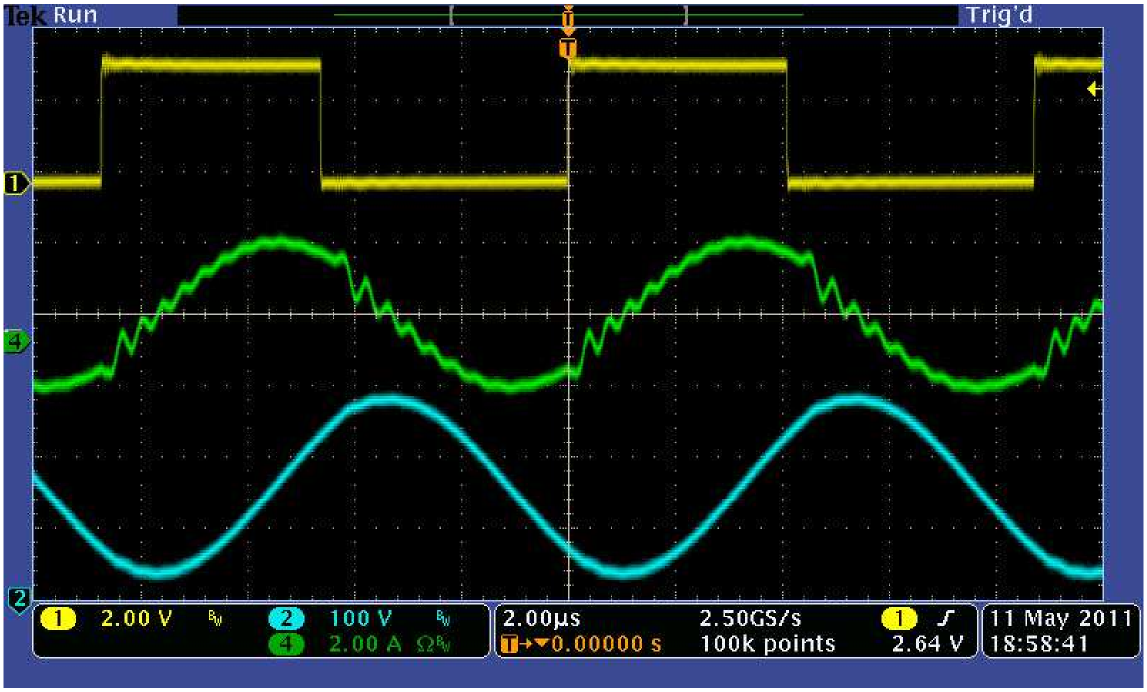


Figure 22. Resonant Tank Operational Waveforms (Yellow – HB HS PWM, Green – Tank Current, Blue – Capacitor Voltage)

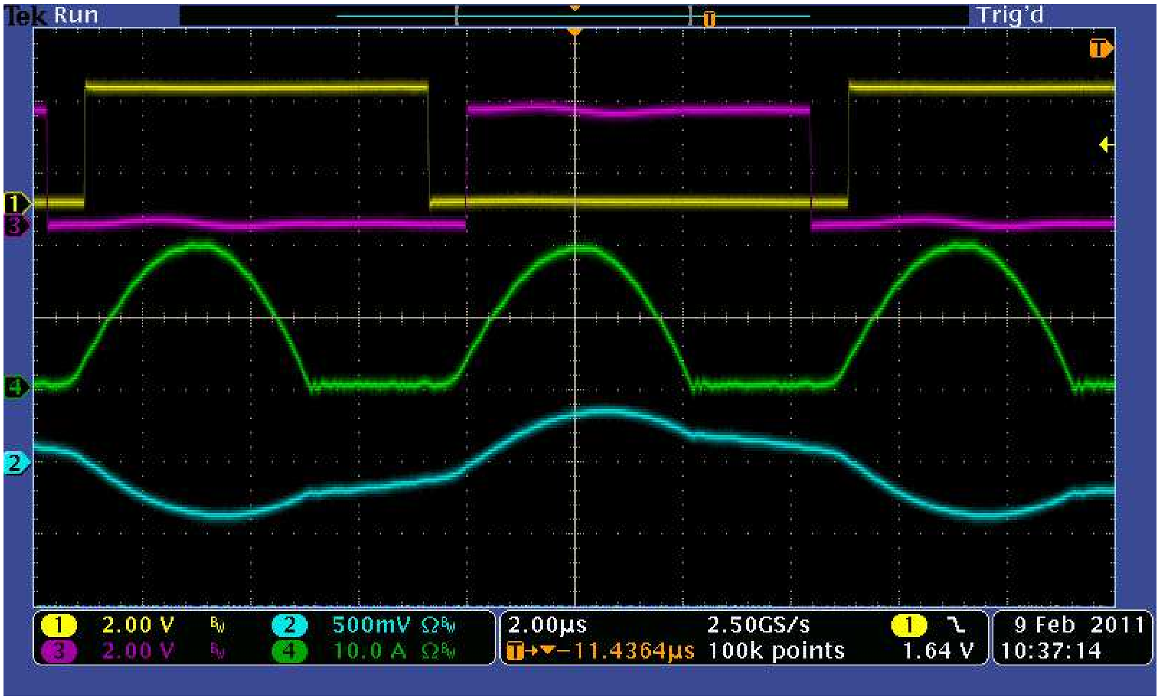


Figure 23. Operational Current Waveforms (Yellow – HB HS PWM, Purple – HB LS PWM, Green – Combined SR Currents, Blue – Tank Current)

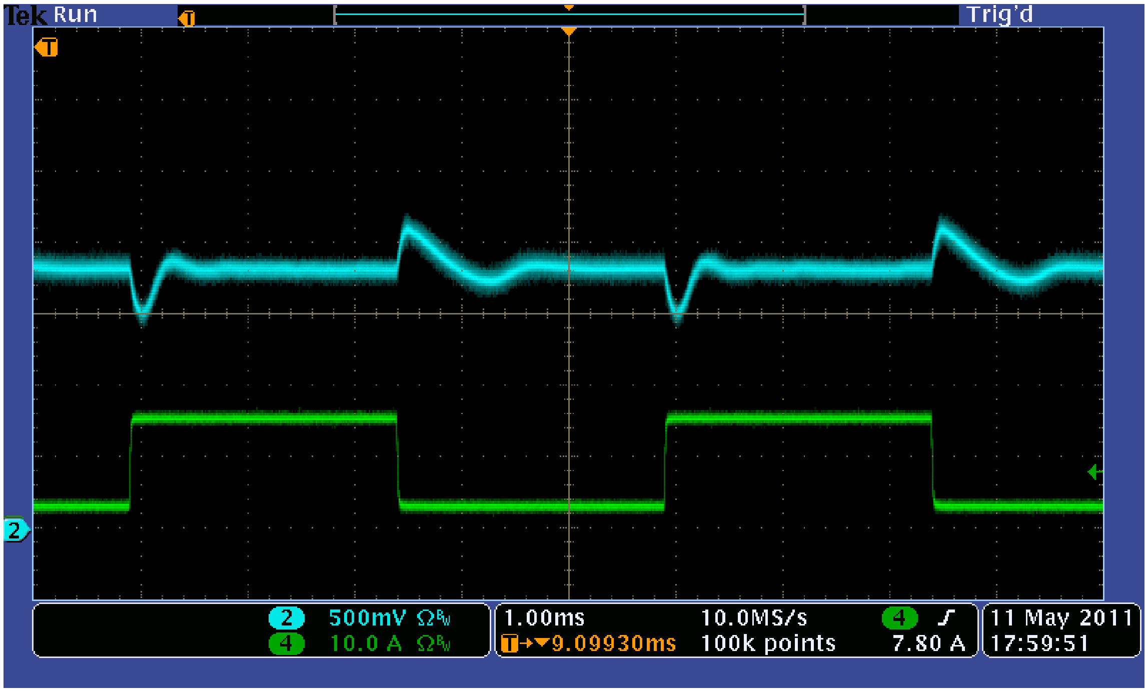


Figure 24. Transient Response: 10% ↔ 60% Load Steps (Blue – Output Voltage, Green – Load Current)

NOTE: Transient results taken using 2P2Z coefficients and 2790 µF output capacitance.

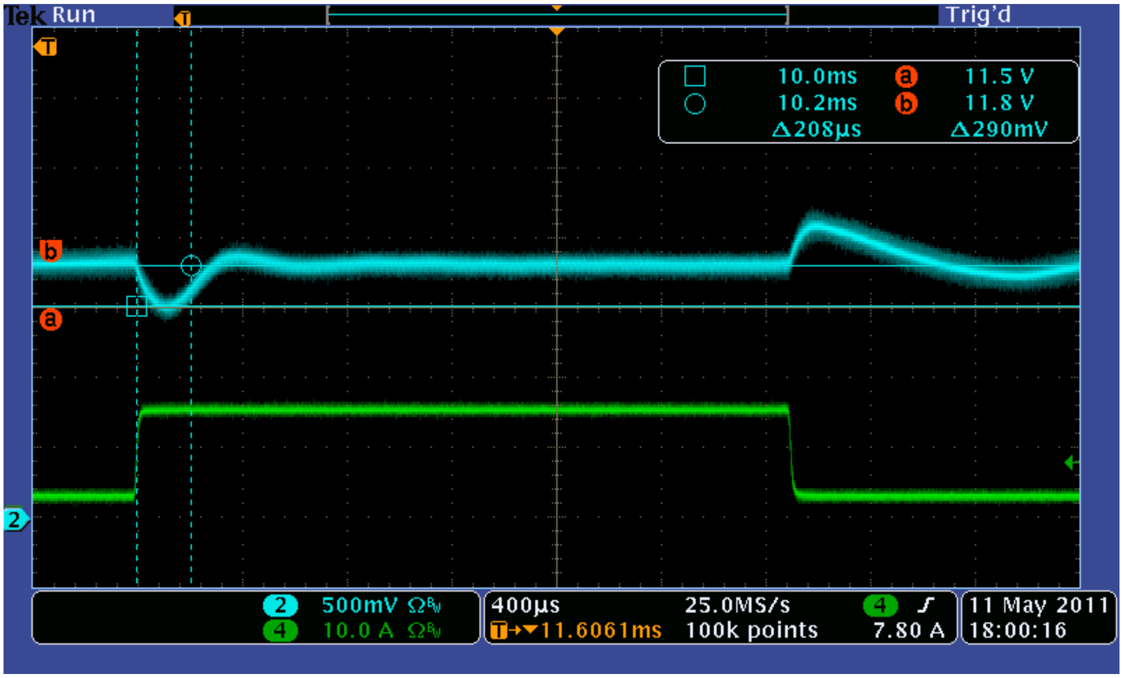


Figure 25. Transient Response: 10% → 60% Load Steps (Blue – Output Voltage, Green – Load Current)

NOTE: Transient results taken using 2P2Z coefficients and 2790 µF output capacitance.

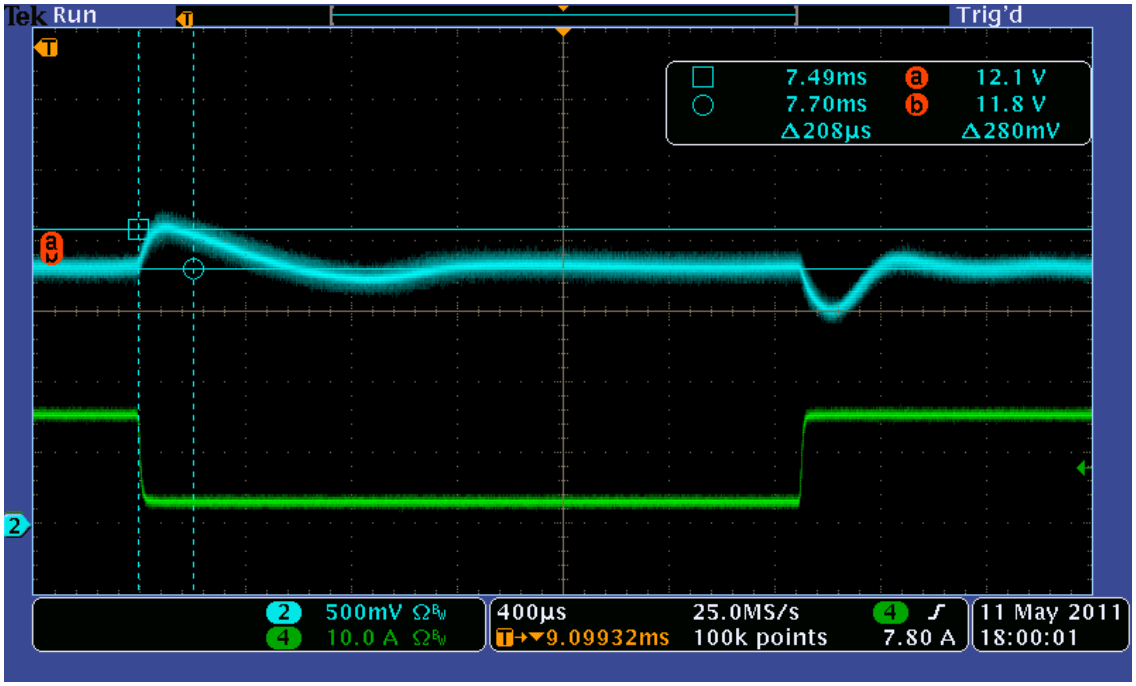


Figure 26. Transient Response: 60% → 10% Load Steps (Blue – Output Voltage, Green – Load Current)

NOTE: Transient results taken using 2P2Z coefficients and 2790 µF output capacitance.

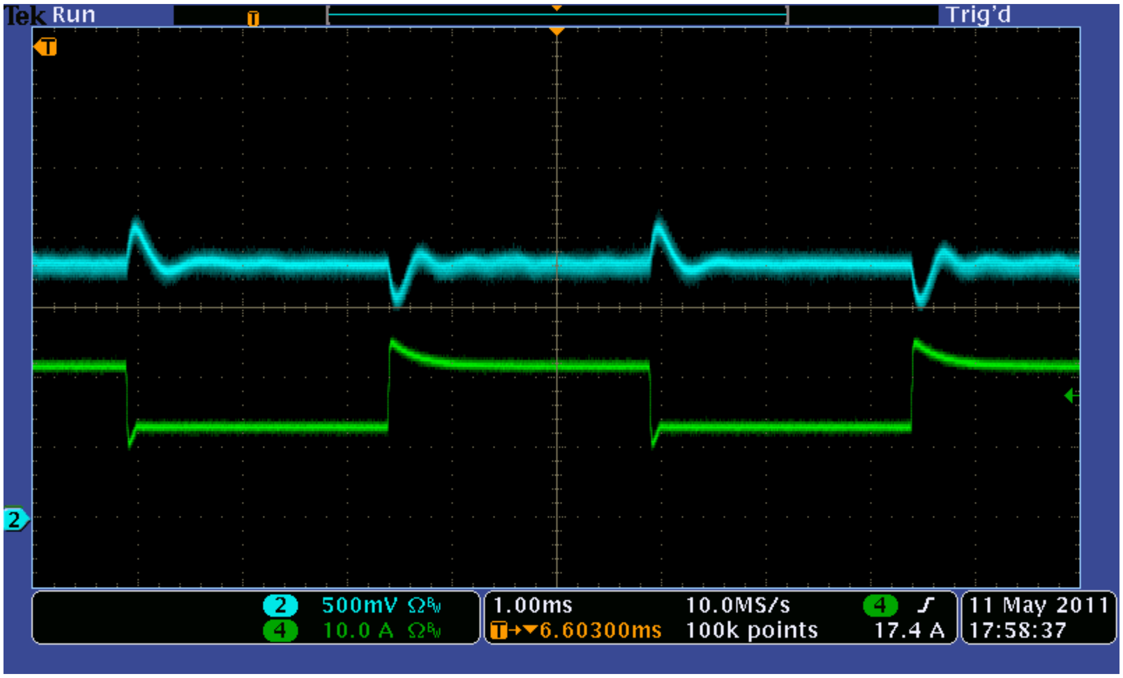


Figure 27. Transient Response: 50% ↔ 100% Load Steps (Blue – Output Voltage, Green – Load Current)

NOTE: Transient results taken using 2P2Z coefficients and 2790 µF output capacitance.

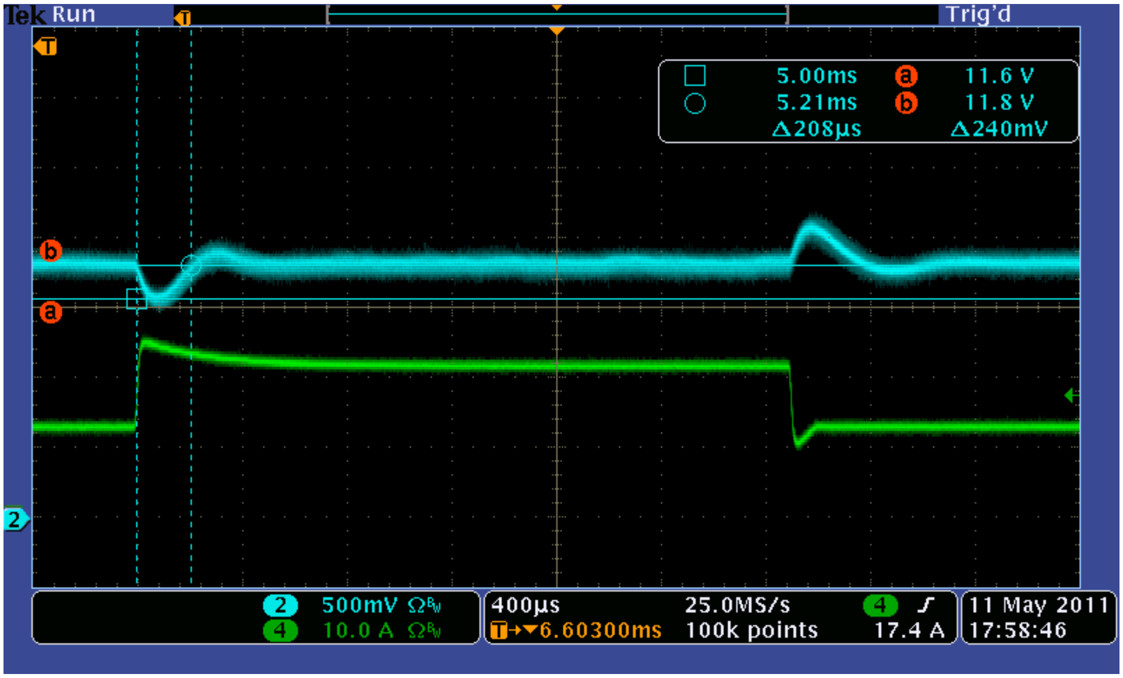


Figure 28. Transient Response: 50% → 100% Load Steps (Blue – Output Voltage, Green – Load Current)

NOTE: Transient results taken using 2P2Z coefficients and 2790 µF output capacitance.

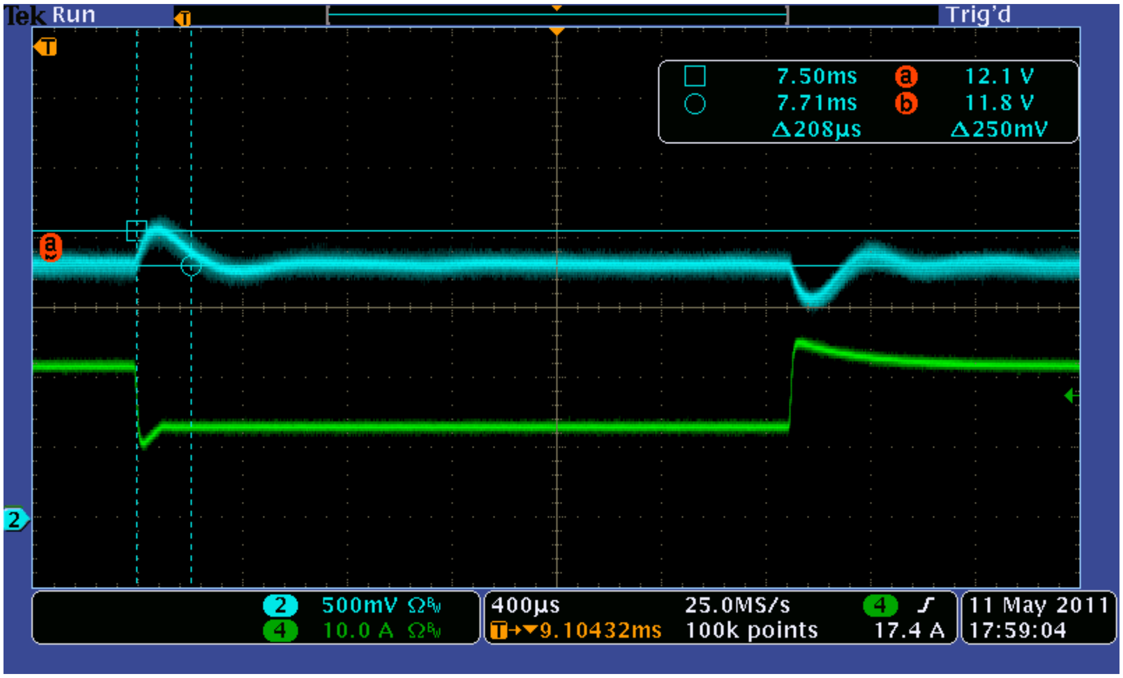


Figure 29. Transient Response: 100% → 50% Load Steps (Blue – Output Voltage, Green – Load Current)

NOTE: Transient results taken using 2P2Z coefficients and 2790 µF output capacitance.

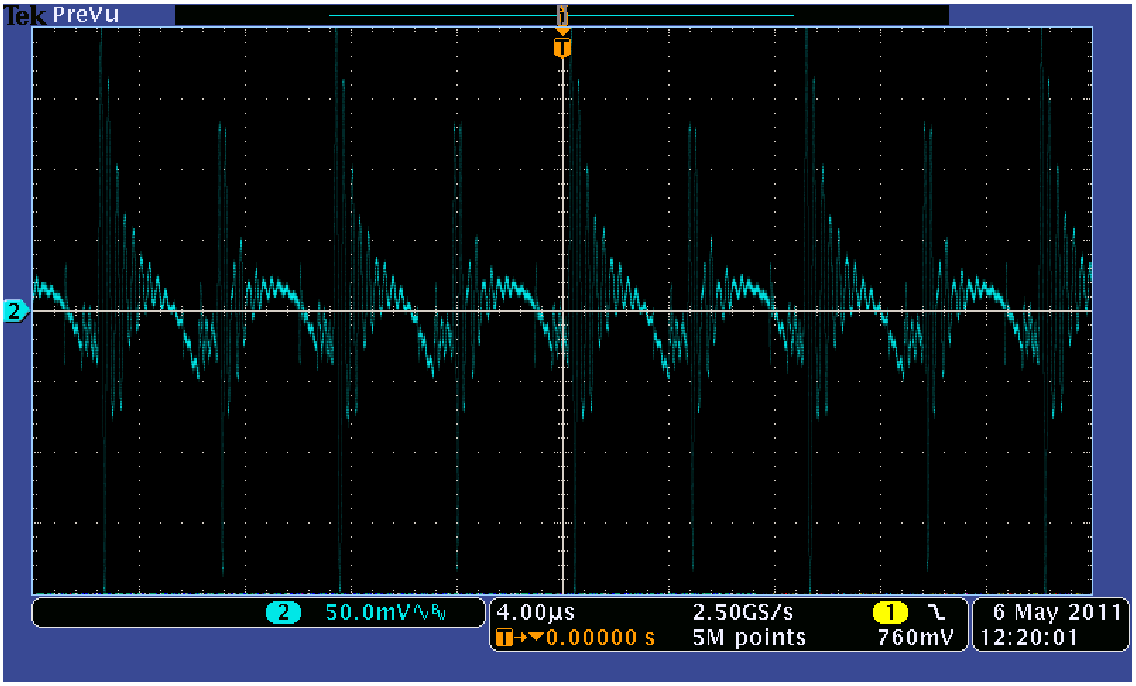


Figure 30. High Frequency Voltage Ripple (Blue – Output Voltage)

7 References

For more information, see the following guides that are located at: www.ti.com/controlsuite

- HVLLC-SWGuide– provides detailed information on the CCS4 project

  \TMDSHVRESLLCKIT_v1.0\~Docs\ HVLLC-SWGuide[R3].pdf
- HVLLC-HWGuide – provides detailed information on the hardware on the board

  \TMDSHVRESLLCKIT_v1.0\~Docs\ HVLLC-HWGuide[R3].pdf
- HVLLC-HWdevPkg – a folder containing files related to the hardware on the board (schematics, bill of materials, Gerber files, PCB layout, and so forth).

  \TMDSHVRESLLCKIT_v1.0\~HVLLC-HWdevPkg[R3]\

IMPORTANT NOTICE

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

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

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

TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI components or services are used. Information published by TI regarding third-party products or services does not constitute a license to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

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

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

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

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

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

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

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

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

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