HVMotorCtrl+PFC (R1.1) Kit How to Run Guide

C2000 Systems and Applications Team

This Guide explains the steps needed to run the HVMTRPFCKIT with the software supplied through controlSUITE. The software is located at:

controlSUITE\developement_kits\HVMotorCtrl+PfcKit_v2.0

Following projects are currently available for the kit:

- Fixed point projects based on Piccolo (TMS320F2803x)

- HVACI_Scalar: Scalar Control of AC Induction Motor
- HVACI_Sensorless: Sensorless Field Oriented Control of AC Induction Motor
- HVACI_Sensored: Sensored Field Oriented Control of AC Induction Motor
- HVPM_Sensorless: Sensorless Field Oriented Control of Permanent Magnet Motor
- HVPM_Sensored: Sensored Field Oriented Control of Permanent Magnet Motor
- HVBLDC_Sensorless: Sensorless Trapezoidal Control of BLDC Motors
- HVBLDC_Sensored: Sensored Trapezoidal Control of BLDC Motors

- Floating point projects based on Delfino (TMS320F2833x)

- HVPM_Sensorless_2833x: Sensorless Field Oriented Control of Permanent Magnet Motor
- HVPM_Sensored_2833x: Sensored Field Oriented Control of Permanent Magnet Motor

The document assumes the user has read the Kit's Hardware Reference Guide and understood all the safety measures that need to taken. The guide is found under

controlSUITE\developement_kits\HVMotorCtrl+PfcKit_v2.0\~Docs

WARNING

This EVM is meant to be operated in a lab environment only and is not considered by TI to be a finished end-product fit for general consumer use.

This EVM must be used only by qualified engineers and technicians familiar with risks associated with handling high voltage electrical and mechanical components, systems and subsystems.

This equipment operates at voltages and currents that can result in electrical shock, fire hazard and/or personal injury if not properly handled or applied. Equipment must be used with necessary caution and appropriate safeguards must be employed to avoid personal injury or property damage.

It is the user's responsibility to confirm that the voltages are identified and understood, prior to energizing the board and or simulation. When energized, the EVM or components connected to the EVM should not be touched.

Isolation transformers must be used when connecting grounded equipment to the EVM.

Hardware Configuration (Motor Control)

To experiment with the digital motor control part of the kit the following hardware components are needed:

- HVMTRPFCKIT

- TMS320F28035 or TMS320F28335 controlCARD;

- Three-phase motor

- An incremental encoder or sprocket (optional);

- PC with Code Composer Studio (CCSv4) installed;

- Additional instruments such as oscilloscope, digital multi-meter, current sensing probe and function generator. (Please read the Hardware Reference Guide [1] to determine the isolation needs for the equipment).

- A high voltage DC power supply (isolating)

- 15V Power supply (supplied with the kit)

The experimental setup and connection are illustrated in the following section. Refer to the Hardware Guide and HVDMC kit schematic file for detailed configuration of each component and connection of the system in detail.

Note: Keep all the power supplies to zero unless directed to energize.

The HVDMC kit is separated into function-specific macro blocks. Following is the list of all the macros with description of what each is responsible for on the board.

- [Main] - controlCARD connection, jumpers, communications (isoCAN), Instrumentation (DAC's), QEP and CAP connection and voltage translation.

- [M1] - AC power entry takes AC power from the wall/mains power supply and rectifies it. This can then be used for input of the PFC stage or used to generate the DC bus for the inverter directly.

- [M2] - Auxiliary power supply, 400V to 5V and 15V module can generate 15V, 5V power for the board from rectified AC power.

- [M3] - Isolated USB Emulation provides isolated JTAG connection to the controller and can be used as isolated SCI when JTAG is not required.

- [M4] - Two-phase interleaved PFC stage can be used to increase efficiency of operation.

- [M5] - Three-phase inverter, to enable control of high voltage 3-phase motors.

- [M6] - DC power entry generates 15V, 5V and 3.3V for the board from DC power fed through the DC-jack using the power supply shipped with the board.

To easily find a component, each component is referenced with their macro number in the brackets followed by a dash and the reference number. For example, [M3]-J1 refers to the jumper J1 located in the macro M3. [Main]-J1 refers to the jumper J1 located on the board, but out of the defined macro blocks above.
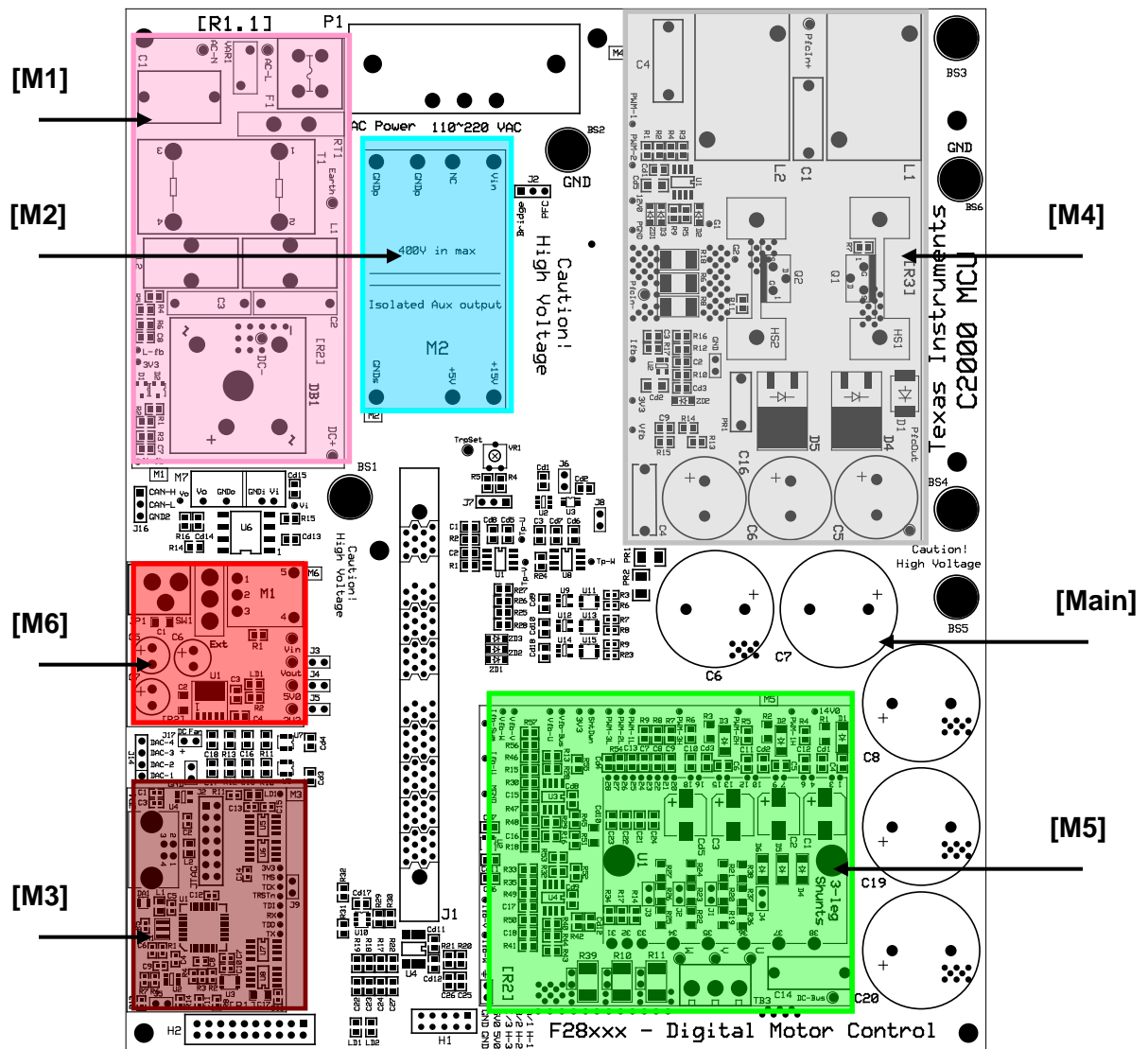
The Layout of HVDMC Board

▪ [Main] - controlCARD connection, jumper configurations, trip zones
▪ [M1] - AC power entry
▪ [M2] - Auxiliary power supply, 400V to 5V and 15V
▪ [M3] - Isolated USB Emulation
▪ [M4] - Two-phase interleaved PFC stage
▪ [M5] - Three-phase inverter
▪ [M6] - DC Power entry

There are two main power domains on the HVDMC platform:

1) **Controller Power Domain** which provides the 15V, 5V and 3.3V for the microcontroller, the logic and the sensing circuit present on the board. Power for this can be driven from two sources:

- Using the 15V DC Power supply, connecting to the DC Jack ([M6]-JP1) present on the DC Power entry Macro. This is the recommended source for all the experiments.
- The Aux Power supply module [M2] present on the board can generates 15V and 5V DC from rectified AC.

2) **DC Inverter Bus Power** is the high voltage line that provides the voltage to the inverter to generate the 3 phase AC to control the motor. There are two options to provide this power to the inverter:

- An external **DC power supply** can be used by connecting to Banana Jacks [BS5] and [BS6]. Using this power is recommended for all the experiments. (Max 350V)
- AC Main ([M1]-P1, 110V* or 220V AC Power supply), power can be rectified and converted to DC power by the rectifier and the capacitor bank present on the board. For safety purpose, use of a variac (variable AC transformer) and isolator is recommended when starting to use this power source.

* Note that the 3-ph Induction motors are typically rated at 220V AC, so the 320 V DC-bus voltage is needed. Thus when using 110V AC power source to generate the DC Bus for the inverter the motor can run properly only at a certain speed and torque range without saturating the PID regulators in the control loop. As an option, the user can run the PFC on HV DMC drive platform as boost converter to increase the DC bus voltage level or directly connect a DC power supply.

⚠️ **Note that the ground planes of both the power domains are the same, hence proper isolation requirements must be met before connecting any test equipment with the board.**

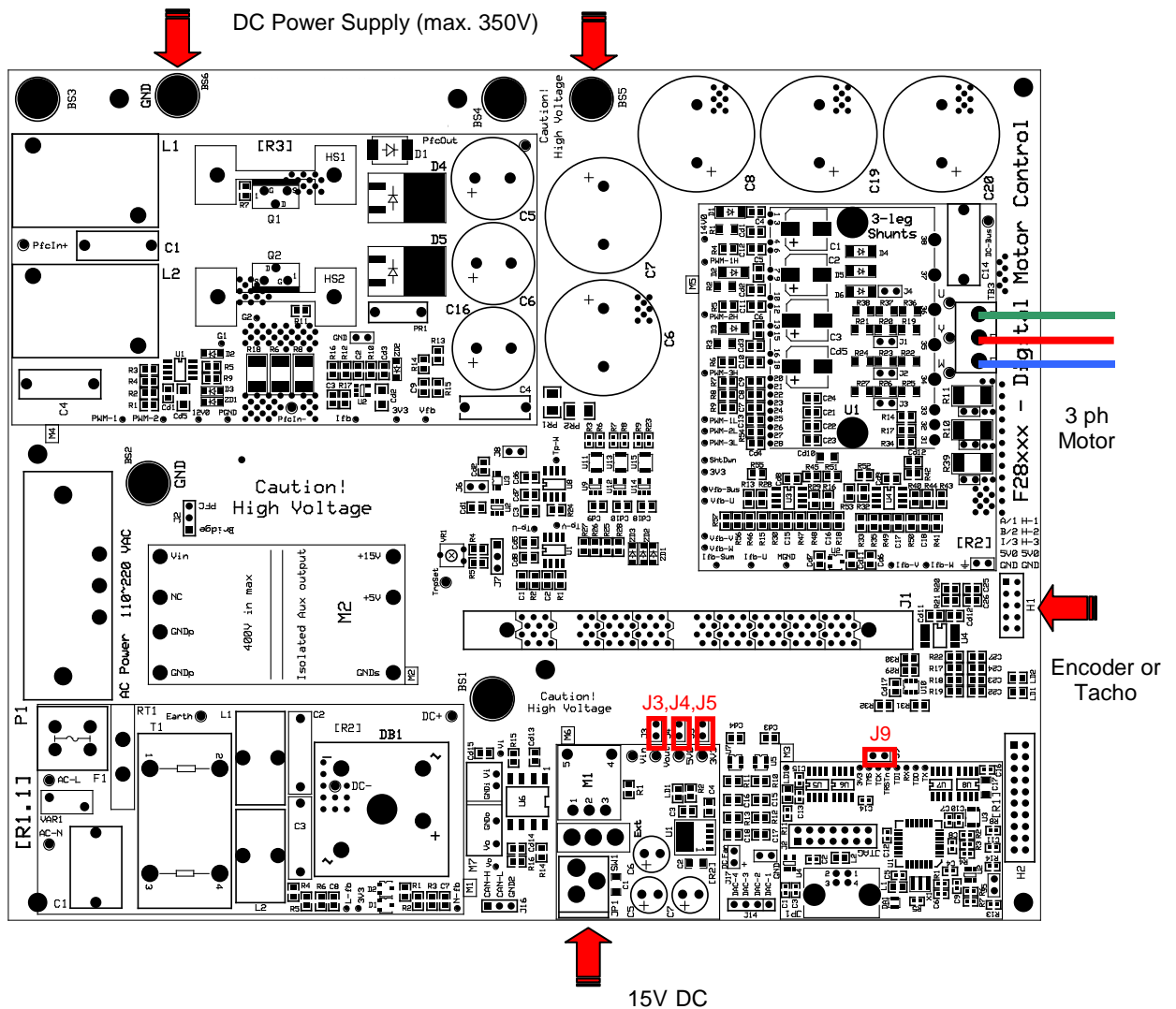## Motor Control Experiment HW Setup Instructions

For the purpose Motor Control Only Projects we would use the DC power entry macro to get the voltages for the controller. For the DC Bus for the inverter either of the two options can be used.

1. Open the Lid of the HV Kit

2. Install the Jumpers [Main]-J3, J4 and J5, J9 for 3.3V, 5V and 15V power rails and JTAG reset line, make sure that the jumpers [M3] - J5 is not populated.

3. Unpack the DIMM style controlCARD and place it in the connector slot of [Main]-J1. Push vertically down using even pressure from both ends of the card until the clips snap and lock. (to remove the card simply spread open the retaining clip with thumbs)

4. Connect a USB cable to connector [M3]-JP1. This will enable isolated JTAG emulation to the C2000 device. [M3]-LD1 should turn on.

5. If a third party JTAG emulator is used, connect the JTAG header. [M3]-J5 needs to be populated to put the onboard JTAG chip in reset.

6. If ISO-DIMM card is used, then do not populate [Main]-J9.

7. Ensure that [M6]-SW1 is in the "Off" position. Connect 15V DC power supply to [M6]-JP1.

8. Turn on [M6]-SW1. Now [M6]-LD1 should turn on. Notice the control card LED would light up as well indicating the control card is receiving power from the board.

9. Note that the motor should be connected to the [M5]-TB3 terminals after you finish with the first incremental build step.
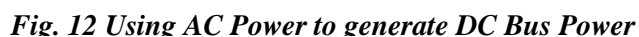
10. **Note the DC Bus power should only be applied when instructed to do so**. The two options to get DC Bus power are discussed below,

- To use DC power supply, set the power supply output to zero and connect [Main]-BS5 and BS6 to DC power supply and ground respectively.

- To use AC Mains Power, Connect [Main]-BS1 and BS5 to each other using banana plug cord. Now connect one end of the AC power cord to [Main]-P1.The other end need to be connected to output of a variac. Make sure that the variac output is set to zero and it is connected to the wall supply through an isolator.

Note for QSG, isolator is not required as measurement equipment is not connected to the board.

Using External DC power supply to generate DC-Bus for the inverter

Fig. 12 Using AC Power to generate DC Bus Power

⚠️ **WARNING: DC bus Capacitors would remain charged for a long time after the mains supply is disconnected. Use caution!**

Software Setup for HVMotorCtrl+PFC Kit Projects

Installing Code Composer and controlSUITE

1. If not already installed, please install Code Composer v4.x from the DVD included with the kit.

2. Go to http://www.ti.com/controlsuite and run the controlSUITE installer. Select to install the "HVMotorCtrl+PFC" software and allow the installer to also download all automatically checked software.

Setup Code Composer Studio to Work with the HVMotorCtrl+PFC kit

3. Open "Code Composer Studio v4".

4. Once Code Composer Studio opens, the workspace launcher may appear that would ask to select a workspace location,: (please note workspace is a location on the hard drive where all the user settings for the IDE i.e. which projects are open, what configuration is selected etc. are saved, this can be anywhere on the disk, the location mentioned below is just for reference. Also note that if this is not your first-time running Code Composer this dialog may not appear)

    - Click the "Browse…" button
    - Create the path below by making new folders as necessary.
    - "C:\Documents and Settings\My Documents\ CCSv4_workspaces\ HVMotorCtrl+PFCKit_workspace"
    - Uncheck the box that says "Use this as the default and do not ask again".
   - Click "OK"
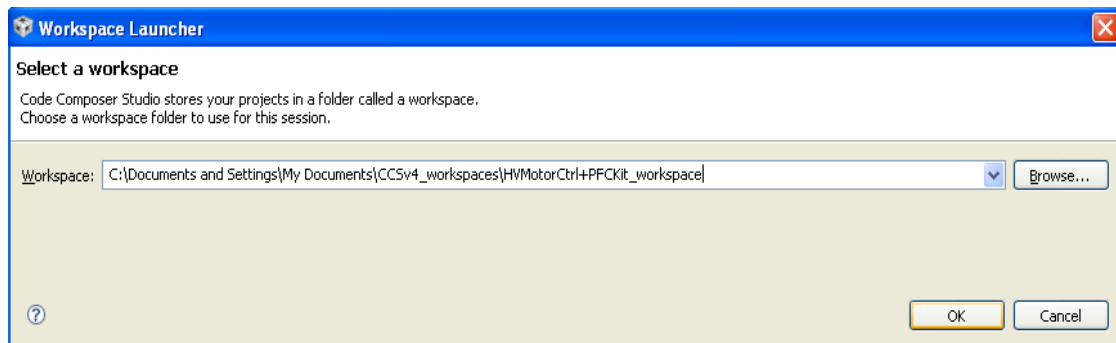
Figure 1: Workspace Launcher

5. Next we will configure Code Composer to know which MCU it will be connecting to. Click "Target -> New Target Configuration…". Name the new configuration xds100-f28035.ccxml or any other name depending on the target device. Make sure that the "Use shared location" checkbox is checked and then click Finish.
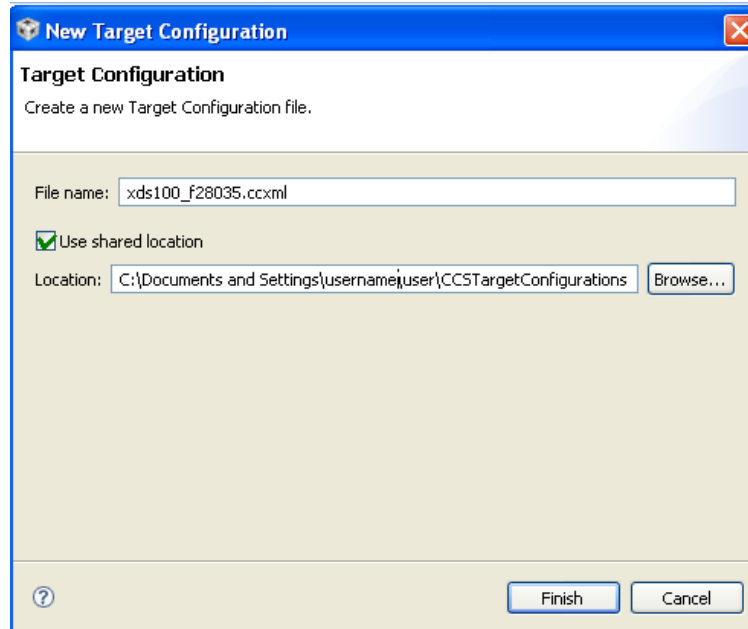
Figure 2: Creating a target configuration

6. This should open up a new tab as seen in Figure 2. Select and enter the options as shown:

    ▪ Connection – Texas Instruments XDS100v1 USB Emulator

    ▪ Device – TMS320F28035 (or TMS320F28335 if Delfino controlCARD is used)

    ▪ Click Save

    ▪ Close the xds100-f28035.ccxml tab (or xds100-f28335.ccxml if Delfino controlCARD is used)
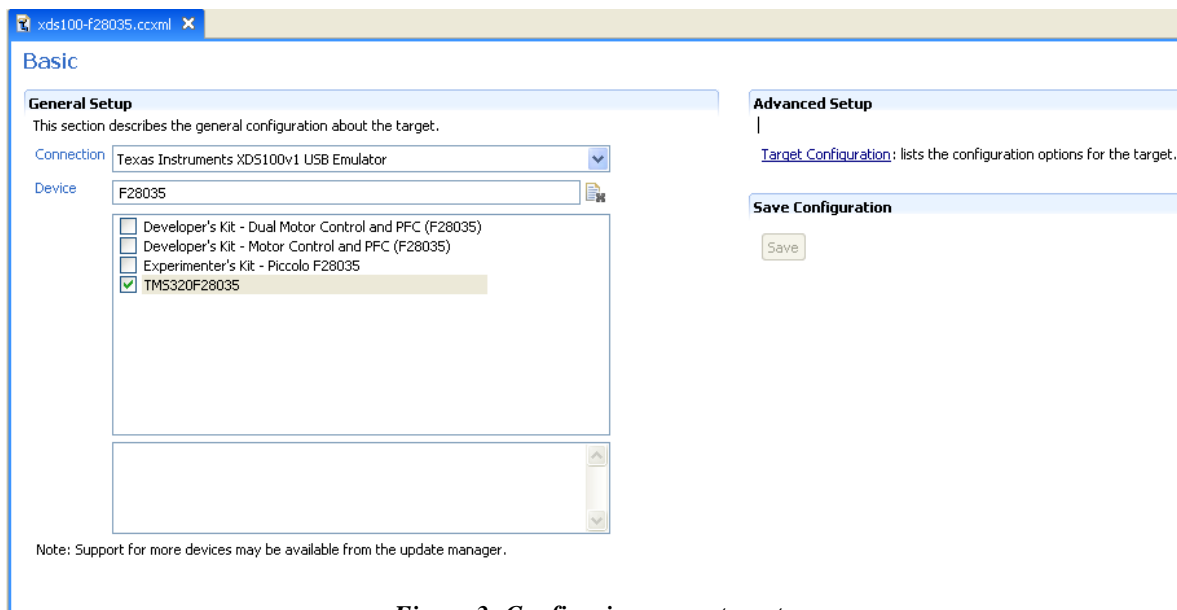
Figure 3: Configuring a new target

7. Assuming this is your first time using Code Composer, the configuration is now set as the default target configuration for Code Composer. Please check this by going to"View->Target Configurations". In the "User Defined" section, right-click on the xds100-F28035.ccxml file and select "Set as Default". This tab also allows you to reuse existing target configurations and link them to specific projects.

8. Add all the motor control projects into your current workspace by clicking "Project->Import Existing CCS/CCE Eclipse Project".

   ▪ Select the root directory of the HVMotorCtrl+PFC. This will be: "C:\TI\controlSUITE\development_kits\HVMotorCtrl+PFC_v2.0\"
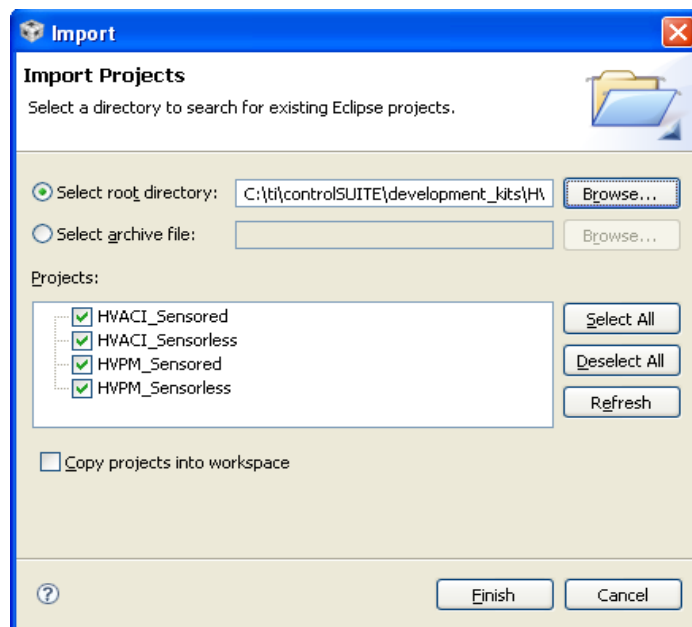
Figure 4: Adding all HVMotorCtrl+PFC kit projects to your workspace

   ▪ Click Finish, this would copy all the projects relevant for the kit into the workspace. If you want only a particular project to be copied uncheck the box next to the other project names.

Configuring a Project

9. Expand the file structure of the project you would like to run from the C/C++ Projects tab. Right-click on this project's name and select "Set as Active Project".

10. Assuming this is your first time using Code Composer, the xds100-F28035 should have been set as the default target configuration. Do verify this by viewing the xds100-f28035.ccxml file in the expanded project structure and a [Active/Default] written next to it. By going to "View-> Target Configurations" you may edit existing target configurations or change the default or active configuration. You can also link a target configuration to a project in the workspace by right clicking on the Target configuration name and selecting Link to Project.

11. Each project can be configured to create code and run in either flash or RAM. You may select either of the two, however for lab experiments we will use RAM configuration most of the time and move to the FLASH configuration for production. As shown in Figure 4, right-click on an

individual project and select Active Build Configuration-> `F2803x_RAM (or F2833x_RAM)` configuration.
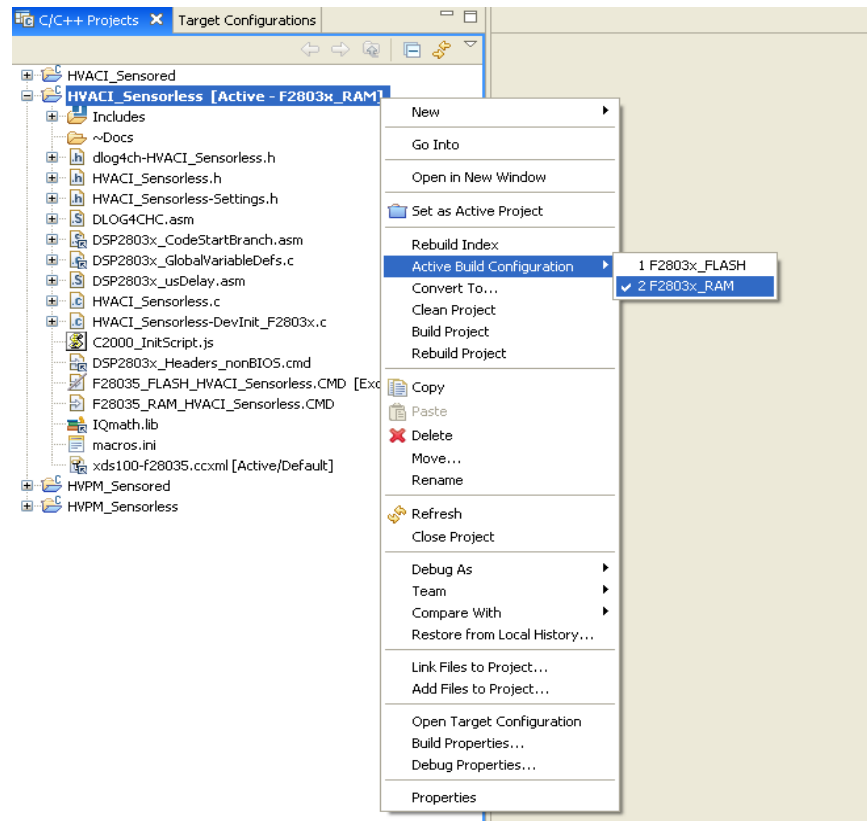
Figure 5: Selecting the F2803x_RAM configuration
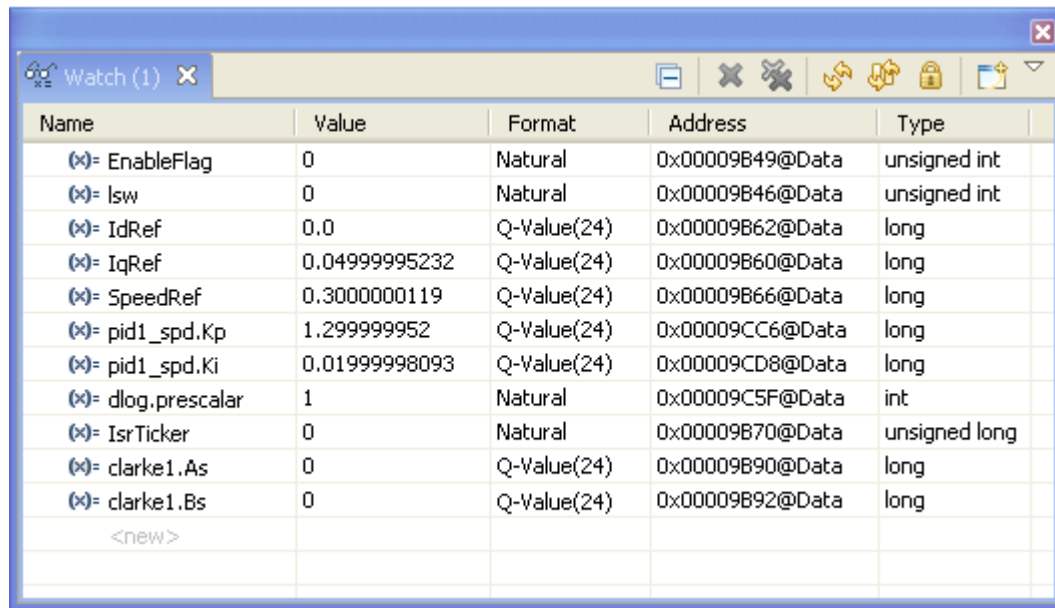
Build and Load the Project

12. The TI motor control software is provided with incremental builds where different components / 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 [Project-Name]-Settings.h and make sure that BUILDLEVEL is set to LEVEL1 and save this file. After we test build 1, this variable will need to be redefined to move on to build 2, and so on until all builds are complete.

13. Open and inspect [Project-Name] -DevInit_F2803x.c (or DevInit_F2803x.c) by double clicking on the filename in the project window. Confirm that GPIO00 to GPIO05 are configured to be PWM outputs.

14. Open the [Project-Name].c file and go to the function MainISR(). Locate the following piece of code in incremental build 1 and confirm that the Datalog buffers are pointing to the right variables. These Datalog buffers are large arrays that contain value-triggered data that can then be displayed to a graph. Note that in other incremental builds different variables may be put into this buffer to be graphed. Following is an example where the datalog are pointed to the space vector generator module.

```
DlogCh1 = (int16)_IQtoIQ15(svgen_dq1.Ta);
DlogCh2 = (int16)_IQtoIQ15(svgen_dq1.Tb);
DlogCh3 = (int16)_IQtoIQ15(svgen_dq1.Tc);
DlogCh4 = (int16)_IQtoIQ15(svgen_dq1.Ta-svgen_dq1.Tb);
```

15. Now 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.

16. 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.

17. Click "Tools->Debugger Options->Generic Debugger Options". You can enable the debugger to reset the processor each time it reloads program by checking "Reset the target on program load or restart" and click "Remember My Settings" to make this setting permanent.

18. Now click on the "Enable silicon real-time mode" button 🔒 and "Enable polite real-time mode" button 🔒. This will allow the user to edit and view variables in real-time. Do not reset the CPU without disabling these realtime options!

19. A message box *may* appear. If so, select YES to enable debug events. This will set bit 1 (DGBM bit) of 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.
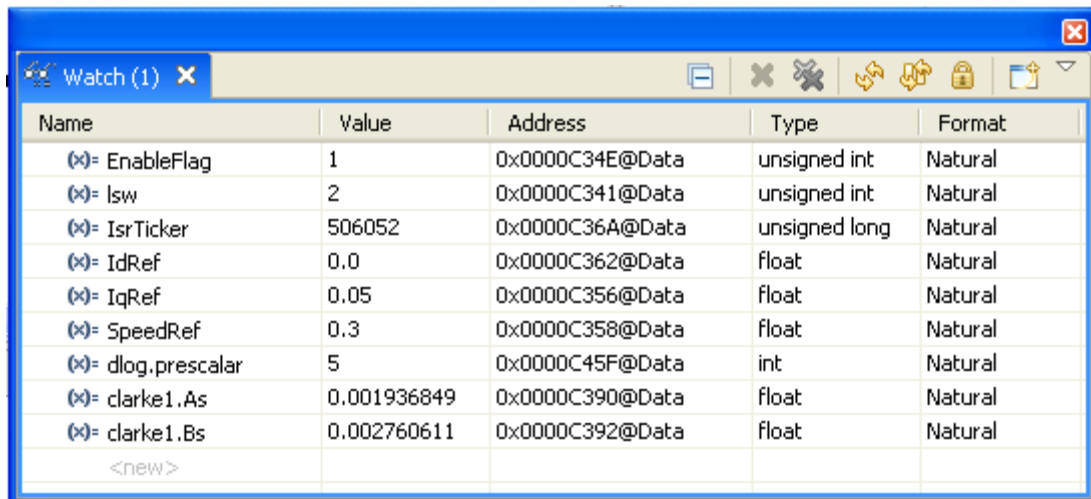
Setup Watch Window & Graphs

Click: View → Watch on the menu bar to open a *watch window* to view the variables being used in the project. Add variables to the watch window as shown below. By right-clicking on the variable it is possible to change the number format of the variable. Refer to the project specific document to know what variables need to be added to the watch window. You can select the appropriate Q format for the variable you want to watch. Figure below shows a typical watch window.

Name	Value	Format	Address	Type
(×)= EnableFlag	0	Natural	0x00009B49@Data	unsigned int
(×)= lsw	0	Natural	0x00009B46@Data	unsigned int
(×)= IdRef	0.0	Q-Value(24)	0x00009B62@Data	long
(×)= IqRef	0.04999995232	Q-Value(24)	0x00009B60@Data	long
(×)= SpeedRef	0.3000000119	Q-Value(24)	0x00009B66@Data	long
(×)= pid1_spd.Kp	1.299999952	Q-Value(24)	0x00009CC6@Data	long
(×)= pid1_spd.Ki	0.01999998093	Q-Value(24)	0x00009CD8@Data	long
(×)= dlog.prescalar	1	Natural	0x00009C5F@Data	int
(×)= IsrTicker	0	Natural	0x00009B70@Data	unsigned long
(×)= clarke1.As	0	Q-Value(24)	0x00009B90@Data	long
(×)= clarke1.Bs	0	Q-Value(24)	0x00009B92@Data	long
<new>				

(a)

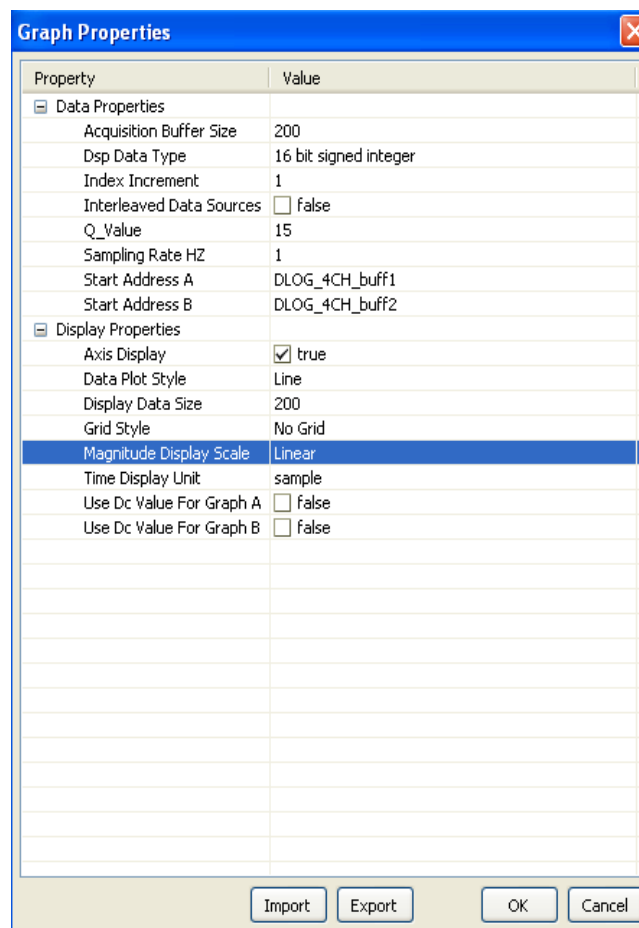| Name	Value	Address	Type	Format
(×)= EnableFlag	1	0x0000C34E@Data	unsigned int	Natural
(×)= lsw	2	0x0000C341@Data	unsigned int	Natural
(×)= IsrTicker	506052	0x0000C36A@Data	unsigned long	Natural
(×)= IdRef	0.0	0x0000C362@Data	float	Natural
(×)= IqRef	0.05	0x0000C356@Data	float	Natural
(×)= SpeedRef	0.3	0x0000C358@Data	float	Natural
(×)= dlog.prescalar	5	0x0000C45F@Data	int	Natural
(×)= clarke1.As	0.001936849	0x0000C390@Data	float	Natural
(×)= clarke1.Bs	0.002760611	0x0000C392@Data	float	Natural
<new>				

(b)

Figure 6: Configuring the Watch Window for (a) fixed point devices (b) floating point devices

20. Click on the Continuous Refresh button 🔄 in the watch window. This enables the window to run with real-time mode. By clicking the down arrow in this watch window, you may select "Customize Continuous Refresh Interval" and edit the refresh rate of the watch window. Note that choosing too fast an interval may affect performance.

21. The datalog buffers point to different system variables depending on the build level. They provide a means to visually inspect the variables and judge system performance. Open and setup time graph windows to plot the data log buffers as shown below. Alternatively, the user can import graph configurations files in the project folder; however, these files are not supported by all CCS4 versions. In order to import them, Click: Tools -> Graph -> DualTime… and select import and browse to the following location C:\TI\ControlSUITE\developement_kits\HVMotorCtrl+PfcKit_v2.0\<project directory> and select Graph1.graphProp, the Graph Properties window should now look like the figure7. Hit OK, this should add the Graphs to your debug perspective. Click on Continuous Refresh button 🔄 on the top left corner of the graph tab.

Note: If a second graph window is used, you could import Graph2.prop, the start Addresses for this should be DLOG_4CH_buff3 and DLOG_4CH_buff4.

Note: The default dlog.prescaler is set to 5 which will allow the dlog function to only log one out of every five samples.
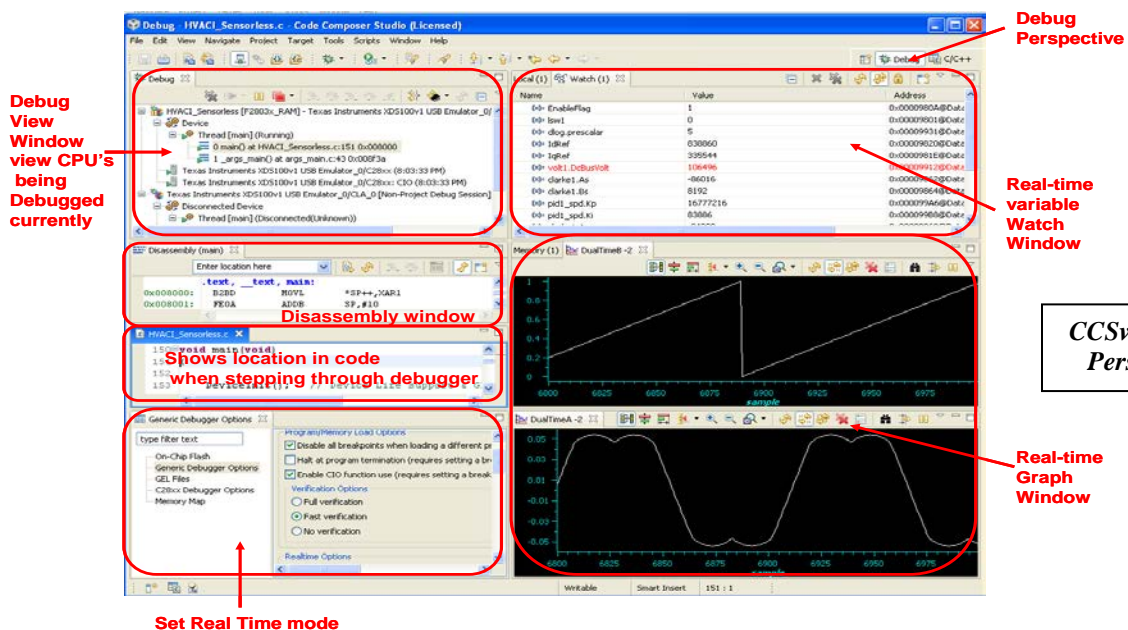

Figure 7. Graph window settings

Run the Code

22. Run the code by pressing Run Button ▶ in the Debug Tab.

23. The project should now run, and the values in the graphs and watch window should keep on updating. Below are some screen captures of typical CCS perspective while using this project, You may want to resize the windows according to your preference.

24. Once complete, reset the processor 🔧 (Target->Reset->Reset CPU) and then terminate the debug session by clicking 🔲 (Target->Terminate All). This will halt the program and disconnect Code Composer from the MCU.

C/C++ Perspective

CCS v4 C/C++ Perspective

Motor Control Code

Project Files

Console window to display Build progress / Errors

Summary of all the Build Errors and warnings go here

Debug Perspective

Debug View Window view CPU's being Debugged currently

Real-time variable Watch Window

Disassembly window

Shows location in code when stepping through debugger

CCSv4 Debug Perspective

Real-time Graph Window

Set Real Time mode

Next Steps

25. It is not necessary to terminate the debug session each time the user changes or runs the code again. Instead the following procedure can be followed. After rebuilding the project, (Target->Reset->Reset CPU) 🐞, (Target->Reset->Restart) 🔄, and enable realtime options. Once complete, disable realtime options, and reset CPU. Terminate the project if the target device or the configuration is changed (Ram to Flash or Flash to Ram), and before shutting down CCS.

26. Customize the project to meet your motor. Change the motor parameters which can be found in [motorproject].h. Feel free to also change the PWM switching frequency (ISR frequency) and the base Q-value to balance accuracy and CPU bandwidth.

27. Now the user can open the lab manual found in : C:\TI\controlSUITE\development_kits\HVMotorCtrl+PfcKit_v2.0\HVxxx_Sensorxx\~Docs and start experiments.