Texas Instruments

Sensor-less 3-phase BLDC Drive, TIDA-00274

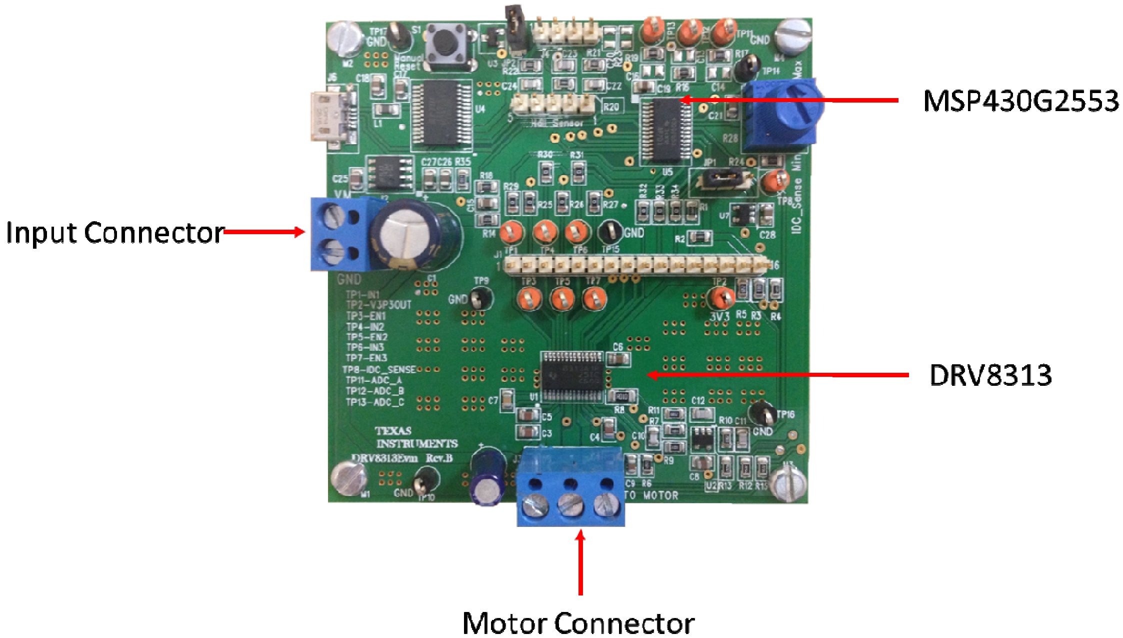


Figure 1: DRV8313 Reference design Circuit Card Assembly

1 Introduction

This reference design provides a cost effective sensor-less solution to run of 3-phase BLDC motor in trapezoidal commutation. This document gives the complete detail of design including its hardware details, jumper configuration, operating procedure to run the BLDC motor using GUI and in Stand alone mode.

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 employed to avoid personal injury or property damage. It is the user's responsibility to confirm that the voltages and isolation requirements are identified and understood, prior to energizing the oard and or simulation. When energized, the EVM or components connected to the EVM should not be touched.

Hardware Block Diagram:

Fig 2, illustrates a typical motor drive system running from dc power supply input (8V-48V). The DRV8313 based reference design motor control board has all the power and control blocks that constitute a typical motor drive system for a 3-phase BLDC motors. T provides easy to use GUI interface to enable motor operation from external PC or laptop.

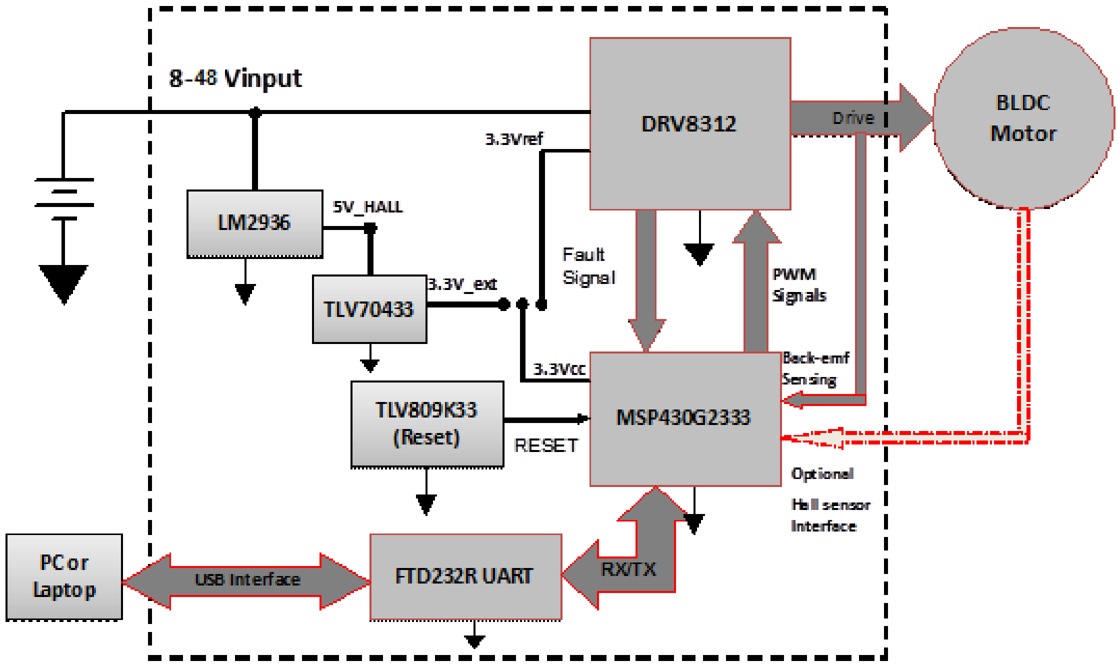


Fig 2: Hardware Block Diagram

Jumper configuration:

Two jumpers are used on the board:

1. Three-Pin Jumper JP1: This jumper is used to give user an option to select 3.3V source for on-board MCU U5 (MSP430G2553). There are two sources of 3.3V, one is 3.3V reference output of DRV8313 and other is generated from input voltage VM through on-board LDOs U6 (LM2936) and U7 (TLV70433).

a. Position 1-2: In this position, MCU U5 is powered from DRV8313. This position is used for input DC voltage range of 8 to 48V.

b. Position 2-3: In this position, MCU U5 is powered from on board LDOs. This mode is used to avoid over-heating of DRV8313 if input DC voltage VM is higher than 48V.

2. Two-Pin Jumper JP2: This jumper must remain closed during all normal operating conditions either through GUI or in stand-alone mode. The jumper is open only during the programming of MSP430G2553 through 4-pin SPI-BY-WIRE method.

Power-up Sequence:

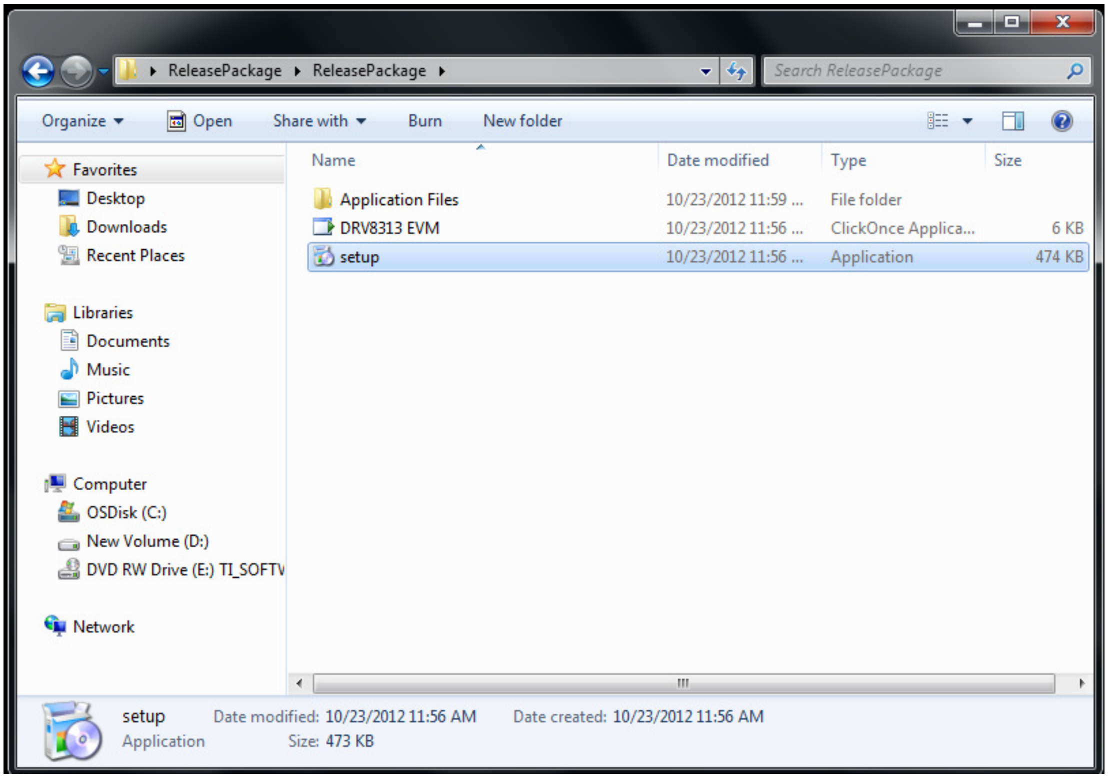
1. Make sure the jumpers are configured as explained in previous section.
2. Connect 3-phase BLDC motor terminal at terminal block J3. It is not important to observe polarity, as it only applies to direction of rotation. Make sure all three pins are connected.
3. Apply VM (+24V) at Terminal Block J2. Board does not have reverse polarity protection therefore make sure positive terminal of power source is connected to VM and negative terminal is connected to GND.
4. Connect USB connector to the J6 USB connector from external PC/Laptop.
5. Open DRV8313 EVM GUI from location Start > All Programs > Texas Instruments Inc. > DRV8313EVM" to enable motor operation.

Installing FTDI Drivers

Download the DRV8313EVM Software and Development package from the TI Web site in the DRV8313EVM Evaluation Module product folder. Find and unzip *Step 1 - INSTALL_USB_Driver* into a separate folder. See instructions on how to install the FTDI USB driver on a Windows™-based computer in the *FTDI_Drivers_Install_Readme.pdf* file.

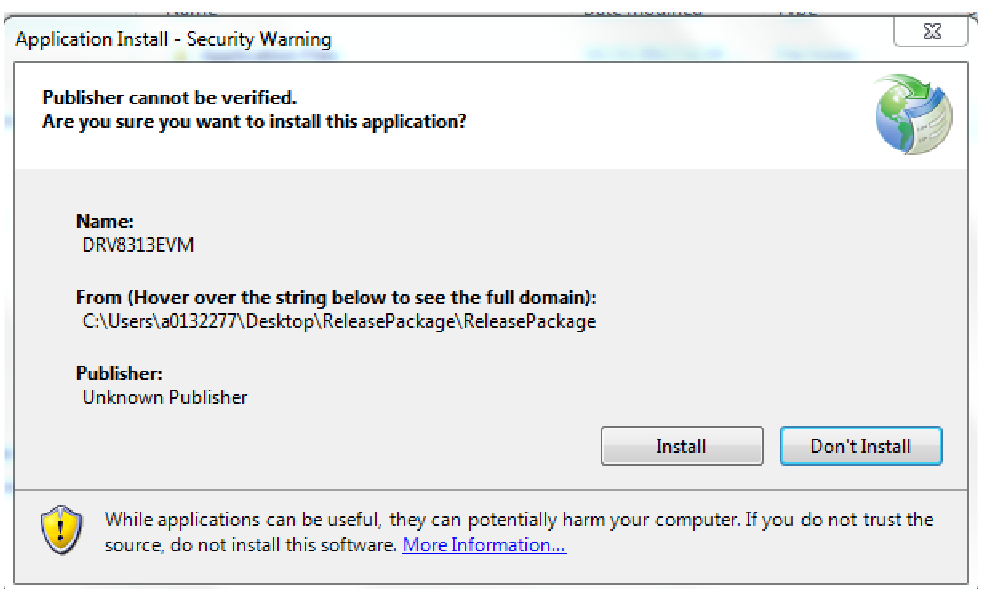
Installing windows Application

The available download also includes a file named *DRV8313EVM_Windows7_Application.zip*. Copy the contents of this folder to any desired folder on your computer. This causes the *Settings.ini* file to be written to the desired folder on software exit. This action cannot occur if the application is run within the zip file. Make sure to first unzip all the contents of the zip file into a folder on your computer. After you unzip the contents, your folder should look like this.



Before installing the application, make sure you have the FTDI drivers installed and your hardware connections are in place.

Double click on the icon 'DRV8313EVM' of 'setup' to run the application. On the first time that you run this application, you will see the following message.



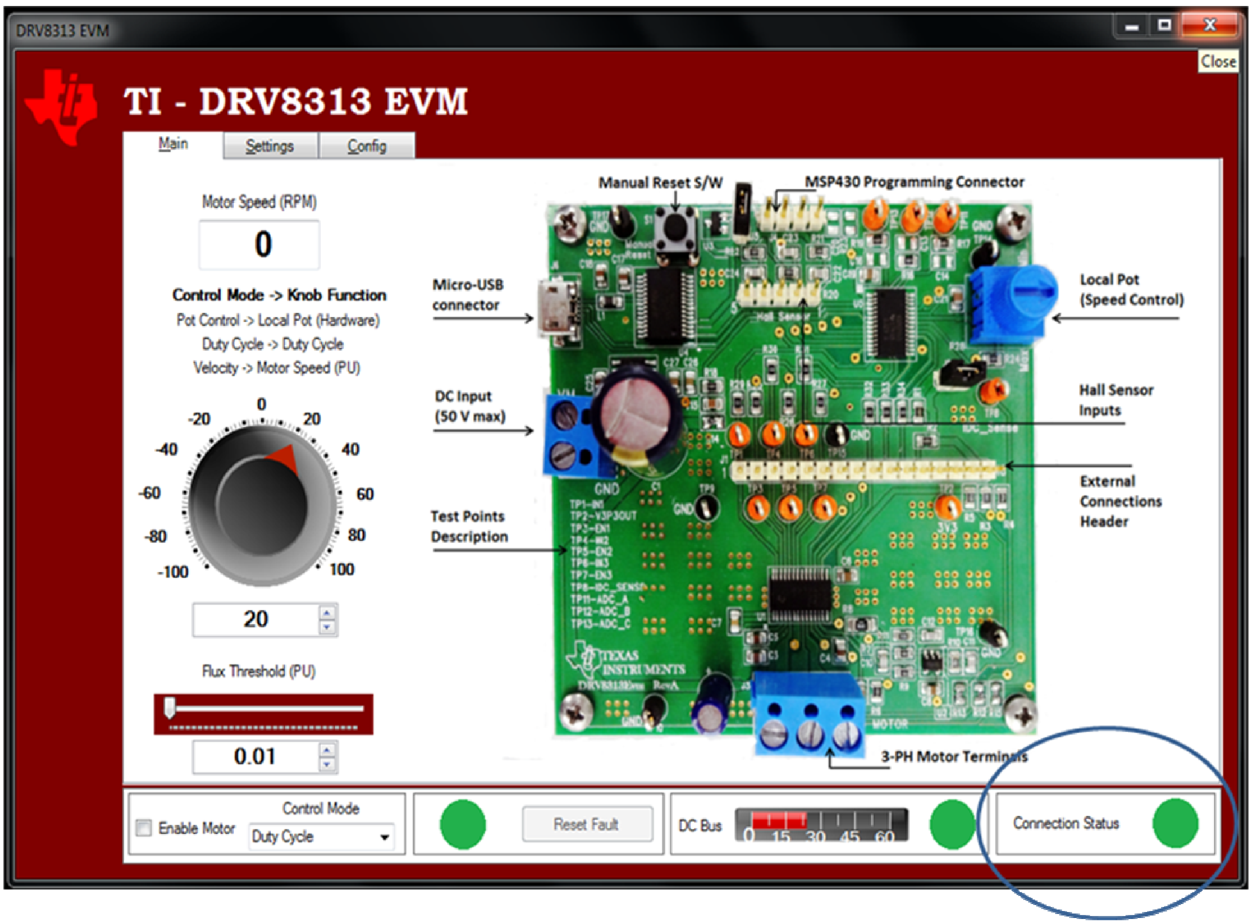
Click on 'Install' button. You should see the setup in action. Once done extracting the required files, the application will launch itself.

Running the Application

For future use, the application can be opened from the following path: “Start > All Programs > Texas Instruments Inc. > DRV8313EVM”

Connecting the Hardware to GUI :

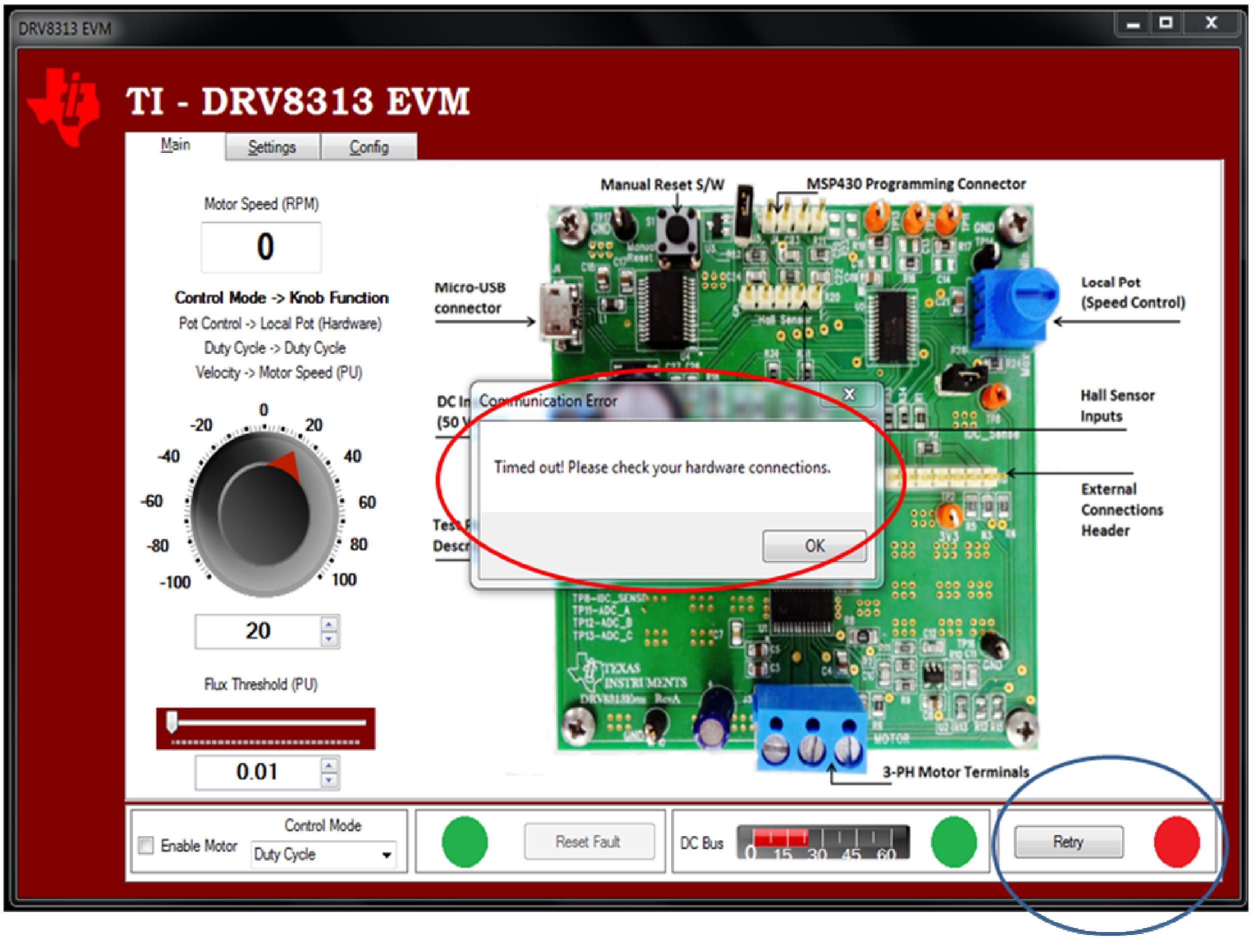
Once the application “DRV8313EVM” is opened, it tries to connect to the hardware. During this attempt the connection status indicator starts blinking at the rate of ~1 sec.



- If the Indicator starts blinking with Green and Light Green in color, the connection is established between the hardware and GUI.
- If the Indicator starts blinking with Green and Red in color, the connection attempt is still going on. This attempt will continue for 10 secs after which the communication error will be indicated as below where ‘Retry’ option will be visible.

If the connection could not be established please check for the following.

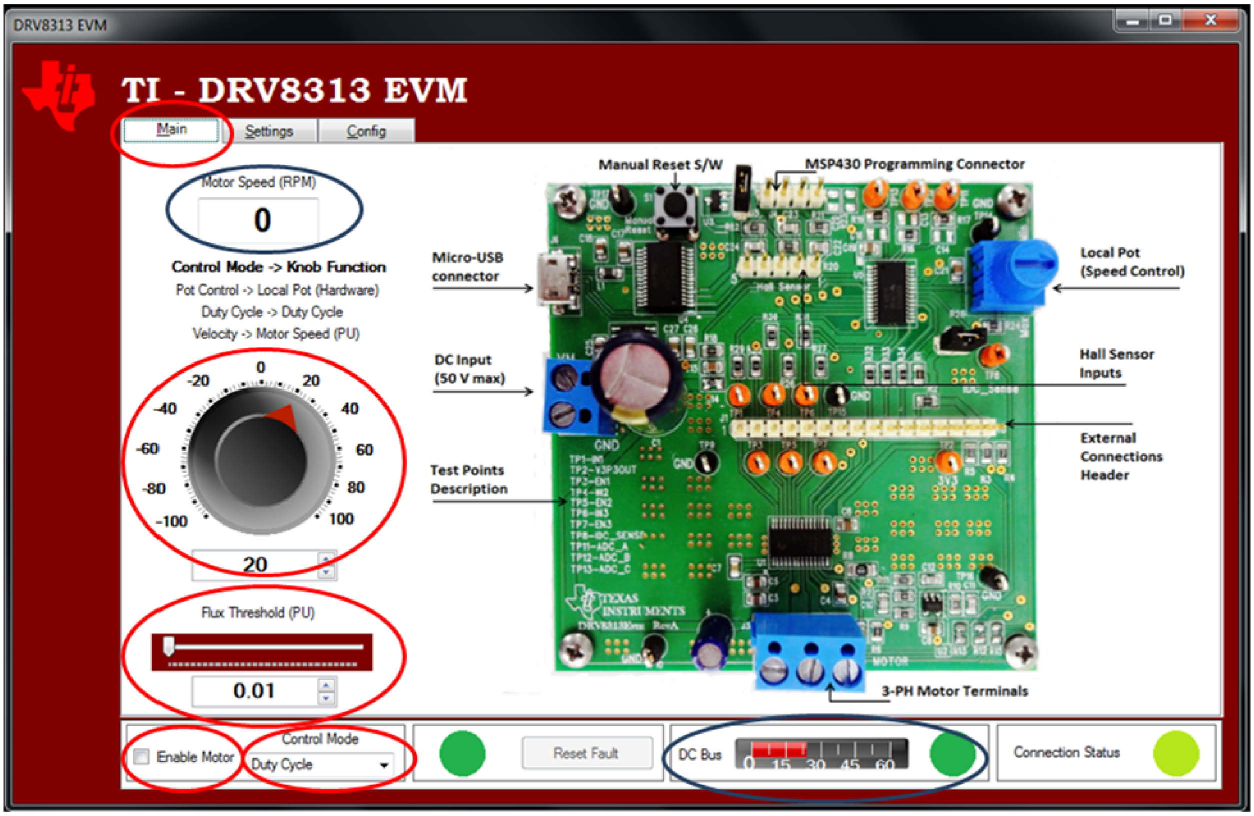
- The EVM is power is present.
- FTDI drivers are installed.
- The Reset jumper (J5) and 3V3 (JP1) jumper is in place.
- The connection cable between the PC and Hardware is connected.



Upon Retry the connection establishment is again attempted for 10 secs.

Running the Hardware:

Once the connection is established as mentioned in first section, the motor parameters can be tuned at 'Main' and 'Settings' tab. Description of each component as given below.



DC Bus:
The Bar graph will indicate the DC Bus voltage.

Motor Speed (RPM):
The Motor speed in indicated in the textbox.

Flux Threshold:
Flux threshold is Integration constant of the motor. Refer to the appendix-A for some of the relevant scope-shots for tuning this value. The Flux threshold can be altered during the runtime.

Control Knob:
Control Knob is used to set duty cycle or required speed during the different speed control modes as explained below. It also is used to reverse the direction of the motor. The parameters can be changed by knob or can be entered directly in text box below the knob.

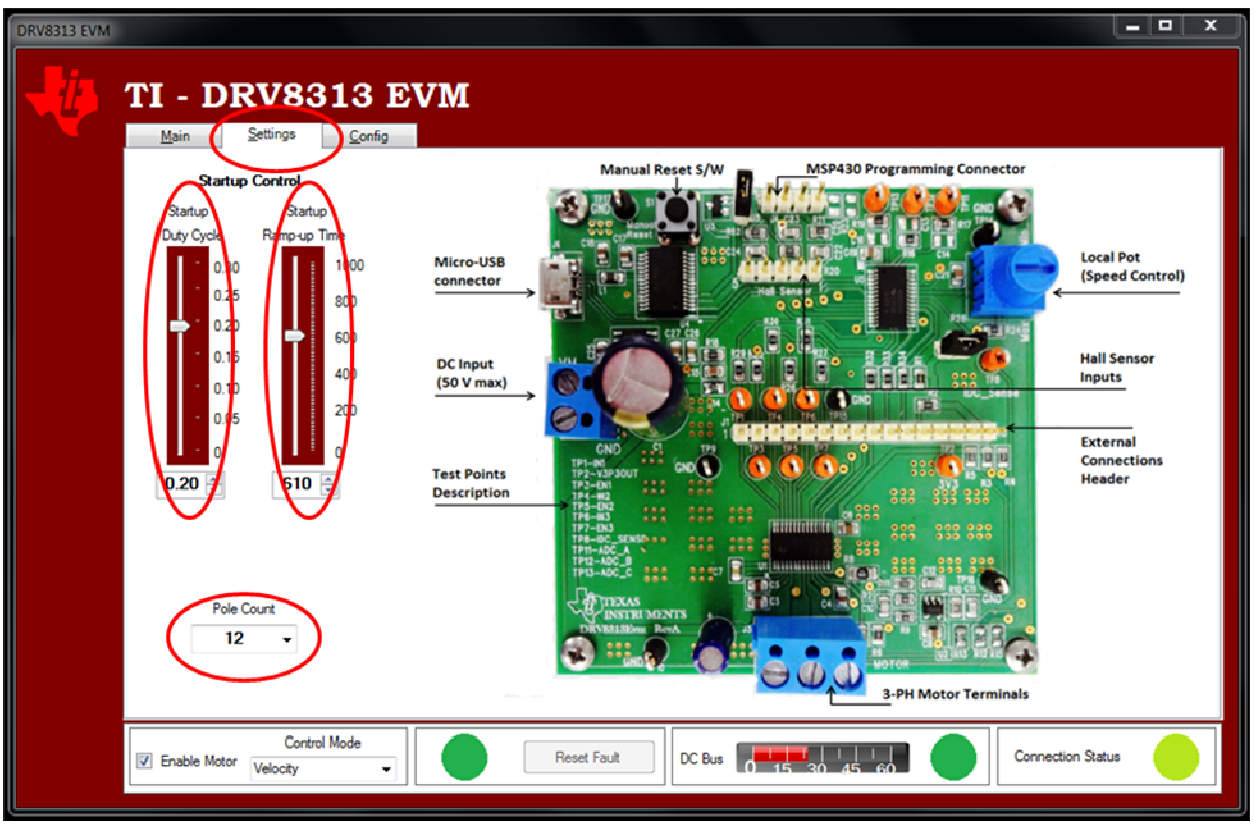
Control Mode:

The Motor Speed can be controlled in 3 control modes,

1. **Duty Cycle:** Motor runs in open loop mode in this control mode. The Control knob can enter the value from 10 to 97 in either direction which represents the % duty cycle needs to be applied to the motor. 10 and 97 are the min and max duty cycle limits applied in the code.
2. **Velocity:** Motor runs in closed loop mode in this control mode. On-Off Hysteretic control is implemented here. The Control knob can enter the speed which is tried to be matched by the hardware. The speed can be seen in the 'Motor Speed' box.
3. **Pot Control:** Motor runs in open loop mode in this control mode. The Control Knob is disabled in this case and the Pot available on the Hardware is used for the speed control purpose.

Enable Motor:

Once the parameters are set, the clicking the check-box should start the motor. Again clicking the checkbox will stop the motor.



Start-Up Process:

In order to start and run the BLDC motor, the control algorithm applies "align and go" method.

- Align State: The main aim of Alignment is to align the rotor to a known position. This is done by energizing two of motor phases for pre-fixed duty cycle to generate standstill stator magnetic field for certain duration. This known position is necessary to start rotation in the proper direction and to generate maximum torque at Start-up.
- Go State: Once the alignment is done, the control algorithm applies the next appropriate voltage vector sequence to produce the stator magnetic field, which is in 90 degree advance to the rotor flux, in order to produce maximum torque at starting. At the same instant the back e-mf integration of non excited phase is initiated and the next commutation event occurs once the back-emf integration threshold is reached and the process continues for the smooth rotation of motor.

Start-Up Duty cycle:

- Changing value of Start-up Duty Cycle will change the applied duty cycle during alignment state. Typical value can very from 10% to 30% depending upon the motor load and inertia.

Start-Up Ramp up Time:

- During alignment state of the motor, this ramp-up time defines time duration for which two of the motor phase remains active for alignment i.e. in other words its simply align state time duration. Please note that initially rotor will experience oscillation when it first moves to get aligned with stator applied field and try to get locked with it. The oscillation time depends upon the mechanical time constant of motor load and inertia. It is very important that before applying 'go state" rotor oscillation dies down sufficiently to avoid commutation failure. Start-up ramp-up time helps to change the timing of align state for different motors to make sure that rotor oscillation have died down and rotor has come very near to standstill.

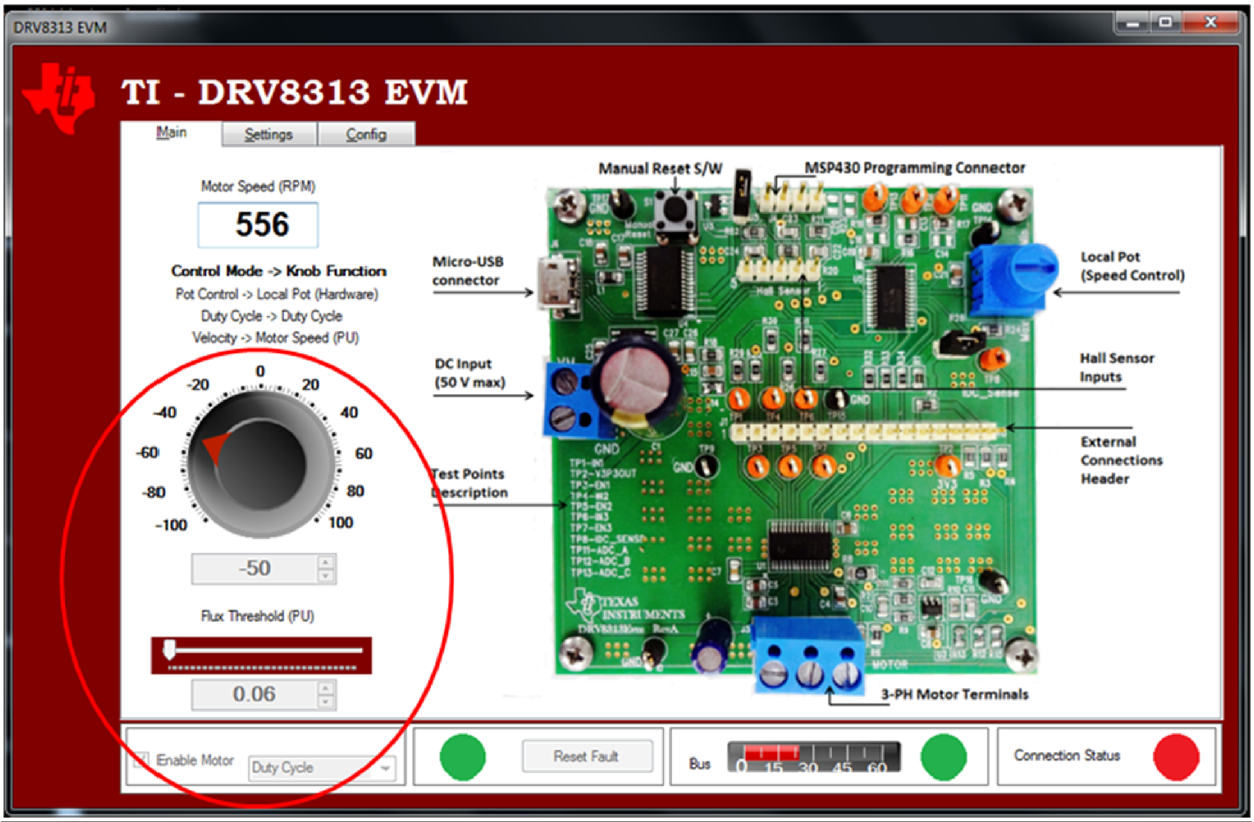
Pole Count:

The number of poles of the motor entered here will be used for the Speed Calculations.

Rotation Reversal Runtime:

The rotation of the motor can be reversed during runtime. The Hardware will try to bring motor to the halt condition first from where it will start to rotate in another direction.

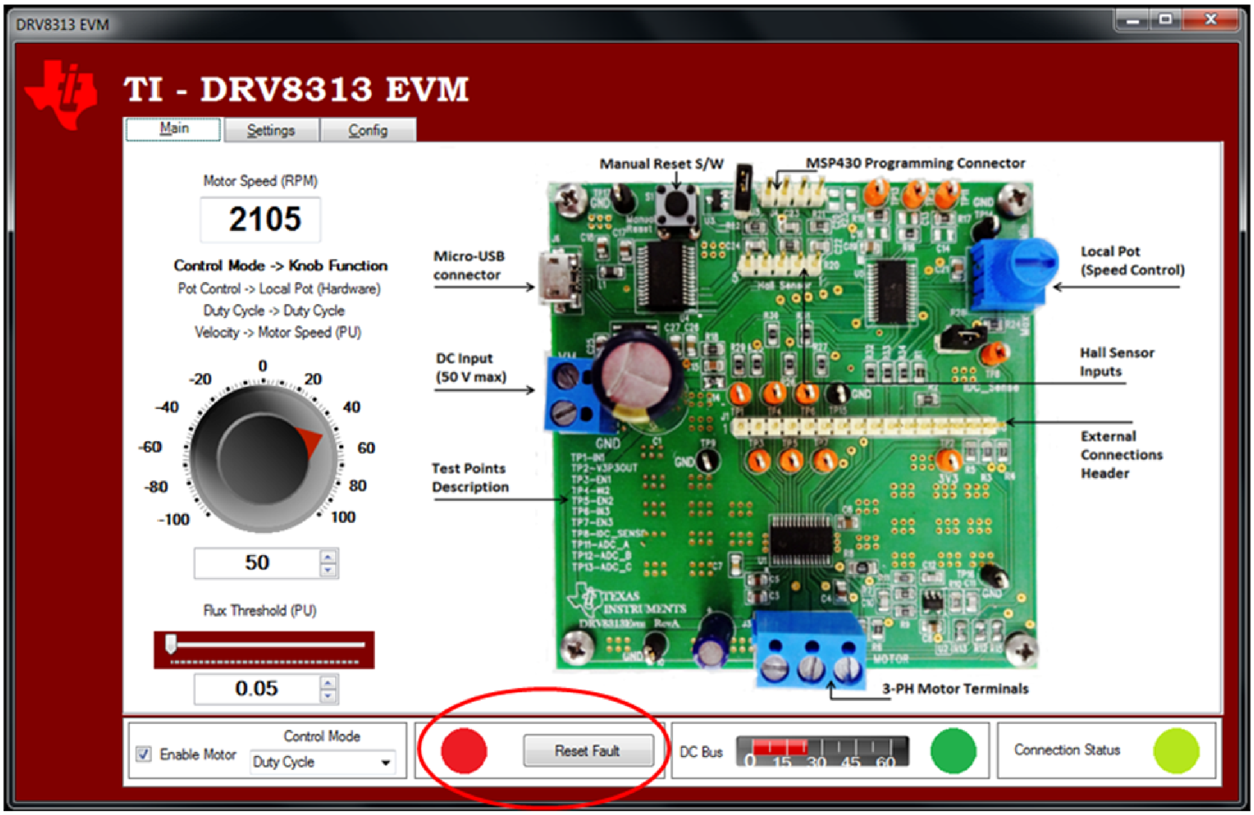
During this period of halting the motor by the hardware, the GUI is kept disabled for entering other parameters.



Please note that parameters related to reverse rotation is programmed for test motor used at factory. It may not be directly suitable for your end application. Please contact your local field application engineer to get details on how to fine tune reverse rotation parameters for your end application.

Fault Clearing:

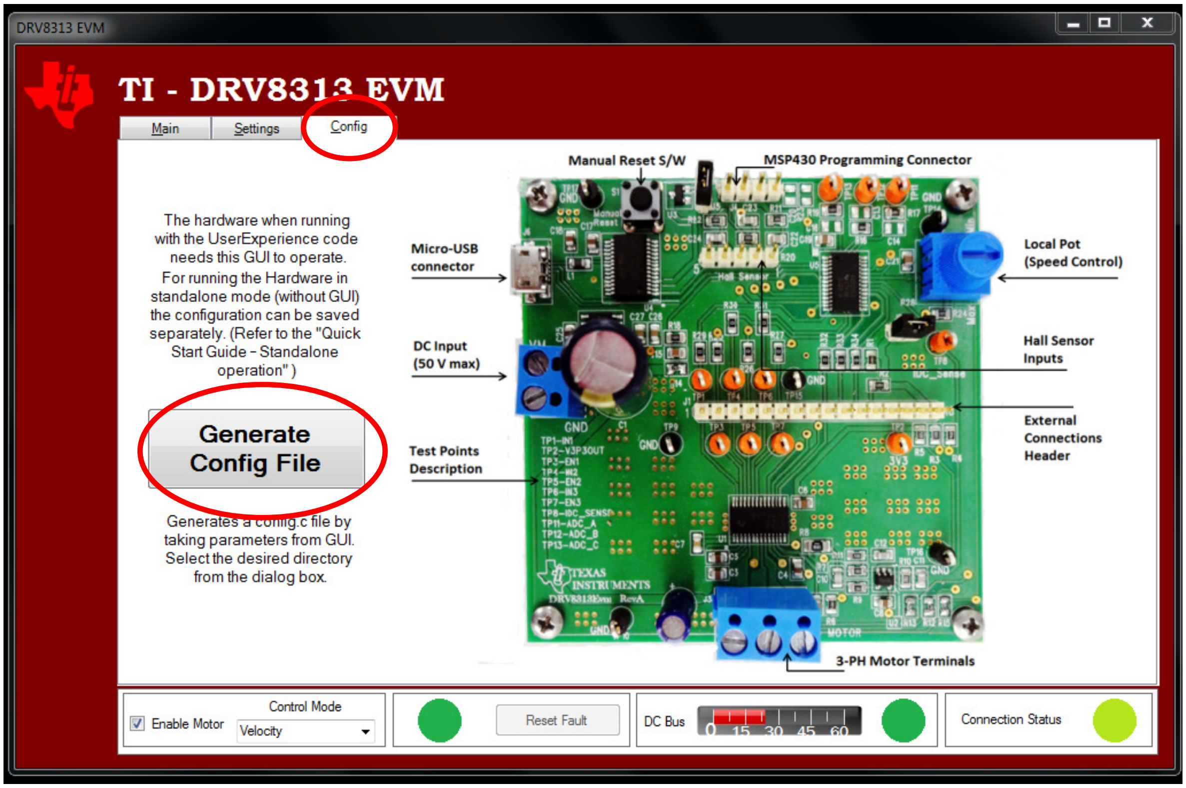
In case of Overcurrent / Short circuit and over-temperature faults at DRV8313, the fault Indicator turns Red as shown below. Clicking on 'Reset Fault' at this time will clear the fault and attempt a fresh start and run.



Running the Hardware in Stand-Alone Mode:

The firmware loaded by default in the MCU on board is a User Experience code. Using this firmware will need the GUI to run the motor.

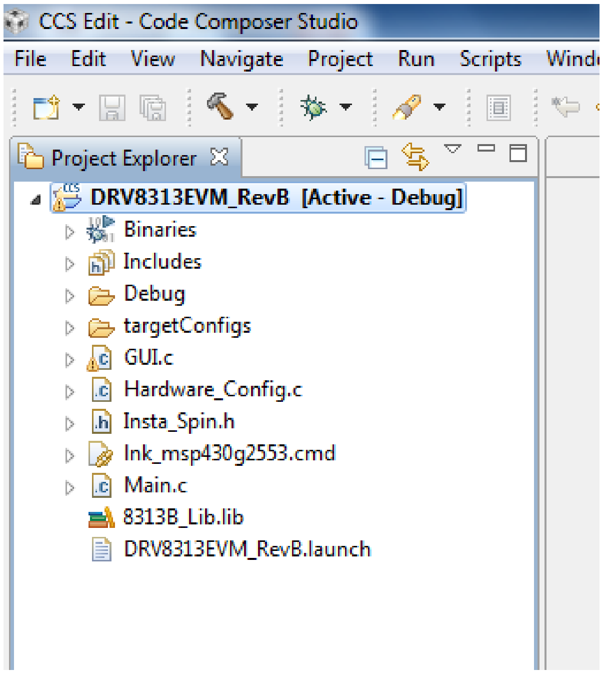
However, once motor parameters are tuned, the output of the GUI can be used to operate the hardware in a stand-alone mode (without GUI). These parameters are given by the GUI in form of the 'config.c' file which is further used to change the code inside the MCU on hardware.



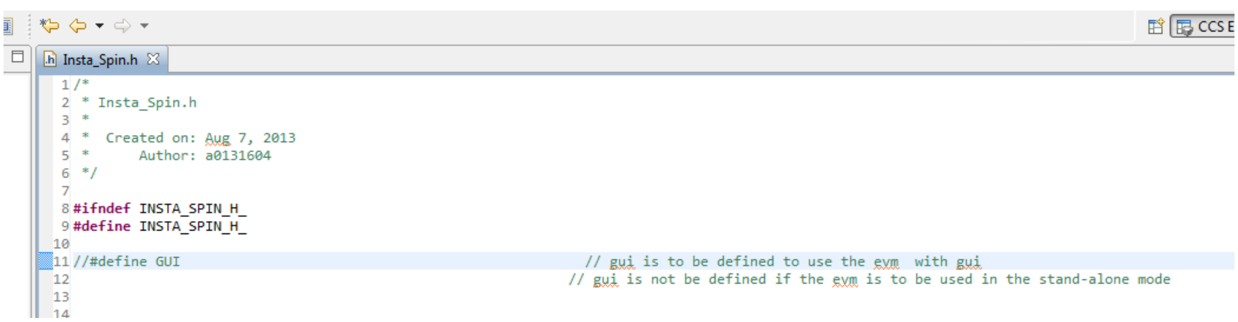
Once the parameters are tuned, clicking on 'Generate Config File' will generate a config.c file which will be used for the creating new firmware for the MSP430 device on the Hardware. This new code can be flashed via MSP430 Programming Connector using Spy-Bi Wire programming method as explained in appendix-B.

Please Note that this code and library is compatible with CCS 5.4 and above.

User Experience code can be downloaded from "Download Link". Importing this project in CCS will get us the following view.



As a first Step, The "#define GUI" needs to be commented as shown in below in the file 'Insta_Spin.h'.



Next, The function “void Init_System_Variables (void)“ needs to be replaced in the file ‘GUI.c’ by the same function which is available in the generated ‘config.c’ file.

With the above mentioned changes, the Hardware can be operated in Stand-alone mode without using GUI.

With the hardware_config.c, The hardware modification can be supported by the software.
With main.c, while (1) available, implementation of the user code shall become simpler. For user code, few more system variables also can be utilized which can be seen in GUI.c
E.g.

Speed_Control = 0 // For Pot Control

= 1 // For Duty Cycle Mode, Use with below
// ***Speed_Duty_Cycle*** = (100 – x) << 2; where x is % duty

= 2 // For Velocity Mode, Use with below
// ***Required_Speed*** = (120 / (Actual_Speed * Number_of_Poles)) * 20000;
// where Actual Speed is the expected mechanical speed.
// this is because one variable is updated every PWM Cycle. i.e 50uS and that variable is compared against variable ***Required_Speed*** for the Velocity Loop

APPENDIX – A

InstaSPIN-BLDC implementation using MSP430G2553

InstaSPIN-BLDC is TI's one of the key flagship motor control technology targeted for cost sensitive sensor-less BLDC applications. This sensor-less technique uses traditional trapezoidal or 120° commutation and monitors motor flux by integrating back-EMF of non-energized phase to determine the commutation instances. For certain market, such as fan, pumps, and blowers etc which do not require high accuracy speed control and fast dynamic torque response, InstaSPIN-BLDC implementation from TI's MSP430G2x Value line device is right way to meet low cost requirements. Some unique aspects utilized to make the solution cost-effective explained below.

For any trapezoidal control of BLDC Motor, for each 60 electrical degrees, only two inverter legs are active and deliver the power to motor while third inverter leg is kept in High impedance state by switching off both high side and low side switches. In case of Uni-polar two quadrant drive, PWM is applied only to high side switch of one active leg while low side switch of other active leg is kept ON continuously for one 60 electrical degree, we need 3 PWM outputs and 3 active signals. This Solution uses the capability of MSP430 microcontroller, routing one PWM output to multiple pins under software control. (In MSP430, this feature is generally provided to make the board layout easy). As shown in the figure A-1, two quadrants Uni-polar Drive can be implemented with only one PWM as only one is necessary at any given point of time in the drive sequence. Hence, this can be easily implemented even using a MSP430 MCU having only one timer. As MSP430 MCU with only one PWM can be used, the lowest end MSP430 is sufficient for speed control and hence it offers a very low cost solution and also makes speed control easy because only one duty cycle number is to be changed.

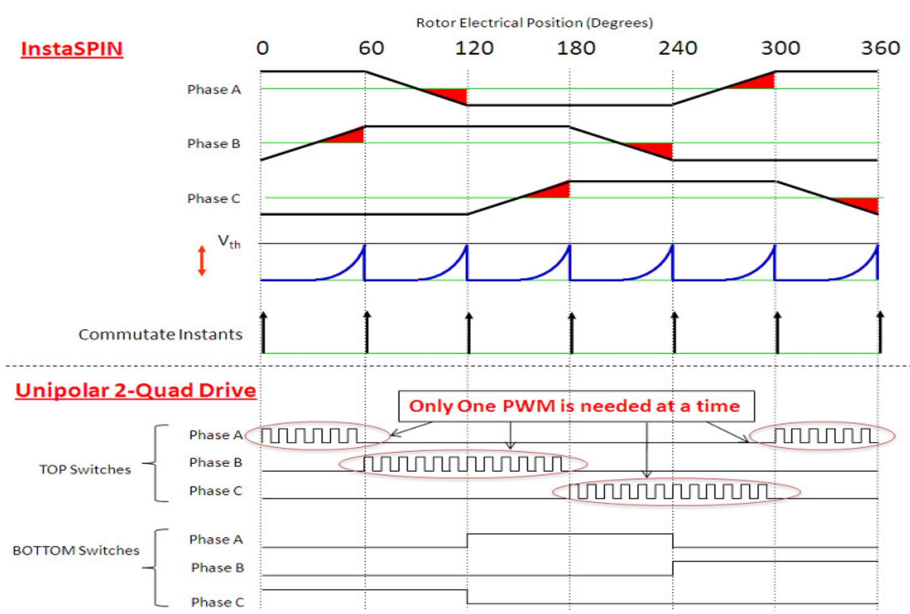


Fig A-1 - Implementation of 2-quardent Uni-polar PWM using single timer

InstaSPIN-BLDC method requires precise sensing of back-EMF of open-phase of motor to determine the commutation instant. As shown in Figure A-1, for first 0 to 60 degree interval, PWM is applied only to phase-A top switch and in phase-C bottom switch is continuously ON, back-EMF of open-phase B is rising and can be measured and integrated to determine the commutation instant. As shown in Figure A-2,During on-time of PWM pulse, with Top switch of phase-A connected to VDC, and bottom switch of phase-C connected to GND, motor neutral terminal potential rises to VDC/2 with respect to GND and back-EMF of phase B appears at VDC/2 level above GND. By capturing the absolute value of voltage across phase-B during PWM on pulse and subtracting it with VDC/2, actual value of motor back-EMF can be derived. Further to filter switching noise superimposed in back-EMF signal, symmetrical centre aligned PWM is used and ADC capture event is synchronized to timer period match as shown in Figure A-3. CCR0 decides the frequency of PWM (20KHz), CCR1 decides the PWM duty cycle for speed control and CCR2 generates the PWM whose rising edge triggers the start of conversion for ADC10.

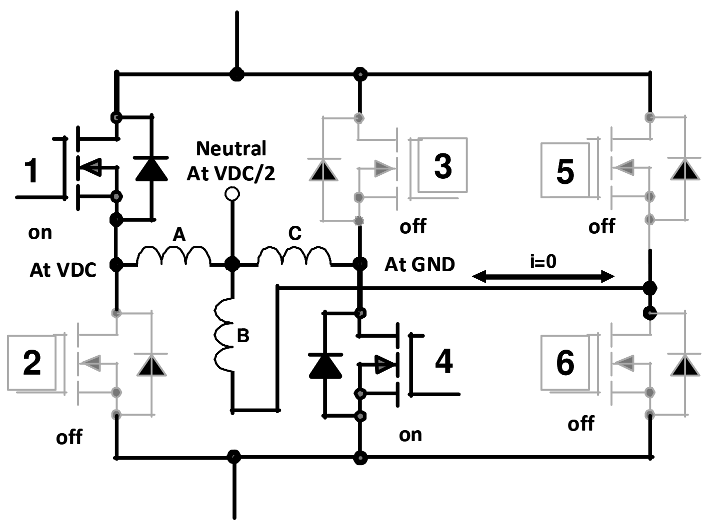


Figure A-2 – Motor Neutral voltage during PWM on period

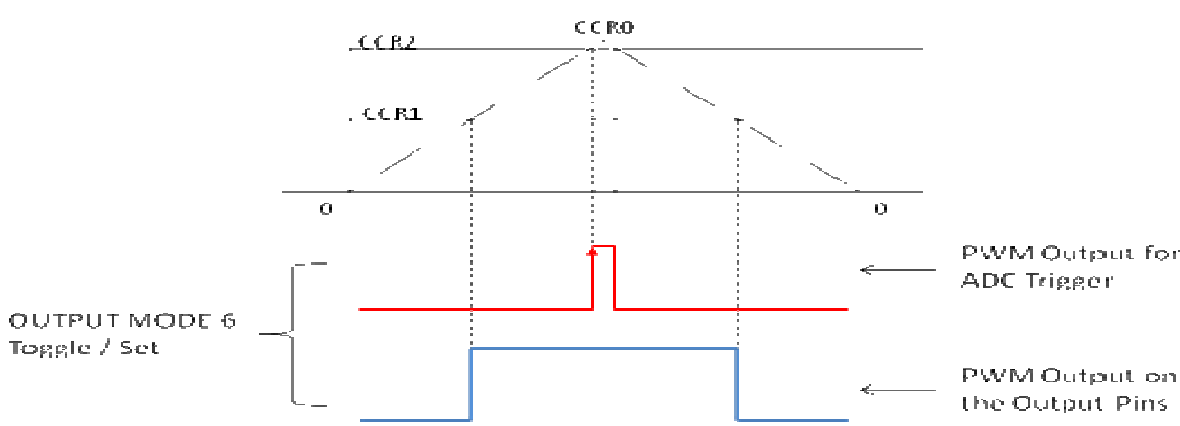


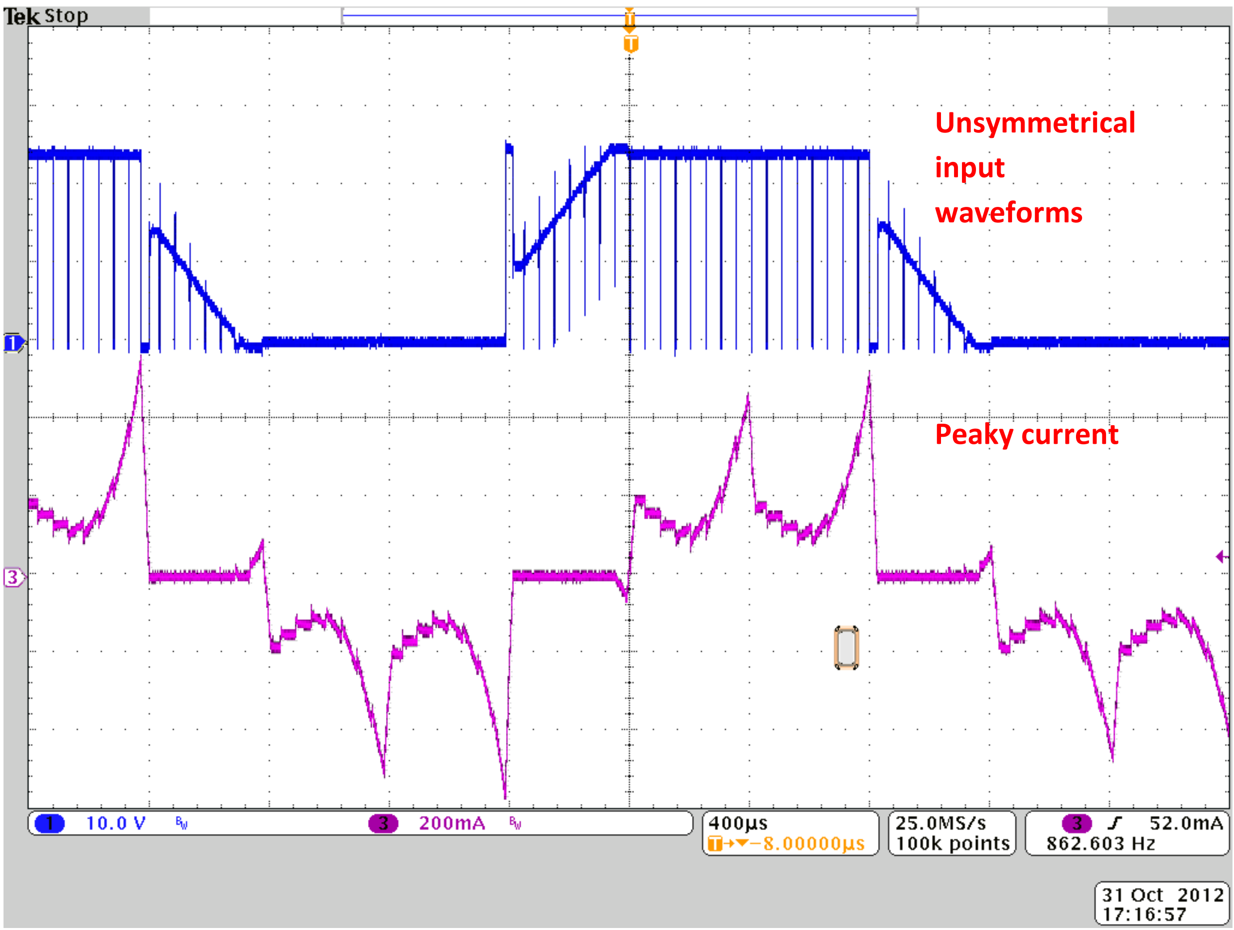
Figure A-3 - ADC capture in sync with Timer period match

Tuning the Motor with Proper Flux Threshold:

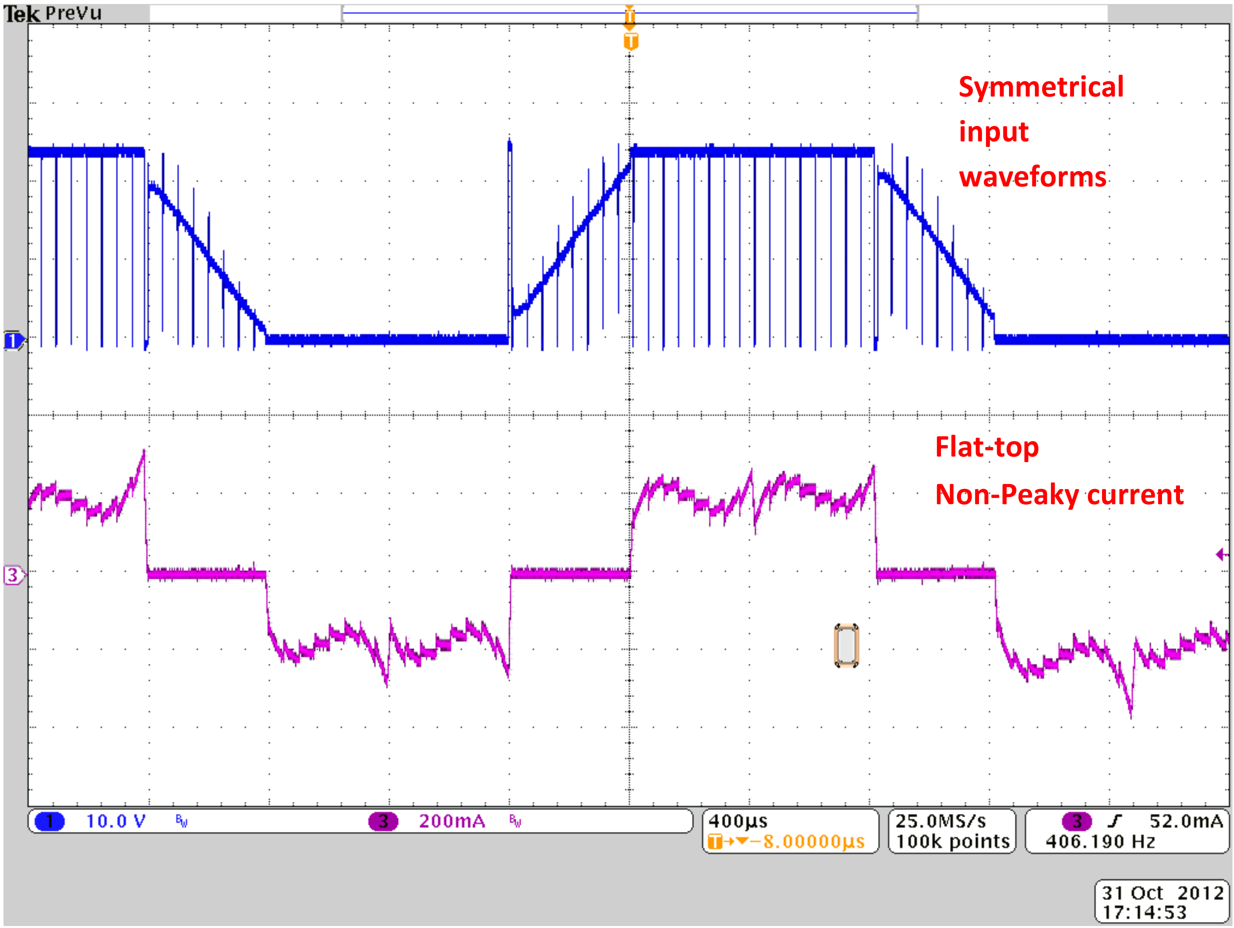
As shown in Figure A-1, the commutation instance is derived by integrating back-emf of non-energized phase (obtain during each PWM on pulse) and comparing the integrated value to pre-defined threshold. This threshold is in fact equivalent to flux of the motor because it is compared with integrated value of back-emf. Value of flux threshold depends upon motor phase resistance, inductance and back-emf constant; therefore this value needs to be tuned for the each different motor for commutation. For tuning purpose usually oscilloscope is used to see the symmetry of the motor input voltages and/or motor phase current. Please refer to the following scope-shots for the different Flux threshold values.

The scope-shots show the motor input voltage and a current waveform for one phase.

Typical Waveform for the HIGHER Flux Threshold



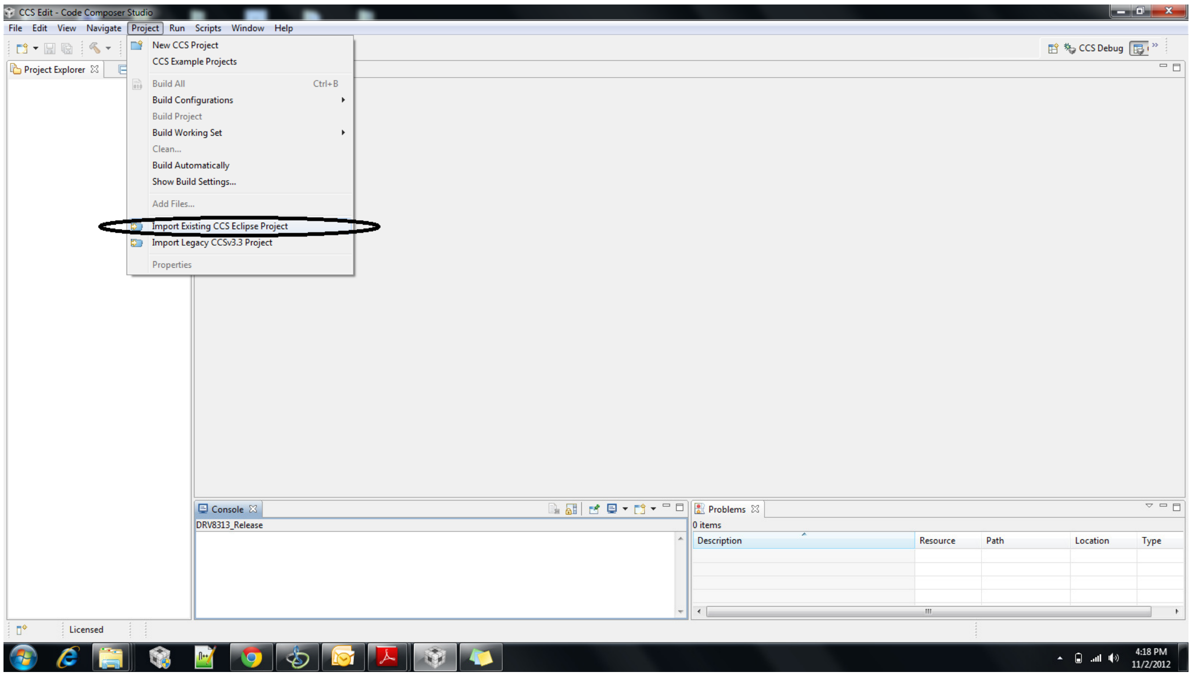
Typical Waveform for the OPTIMUM Flux Threshold



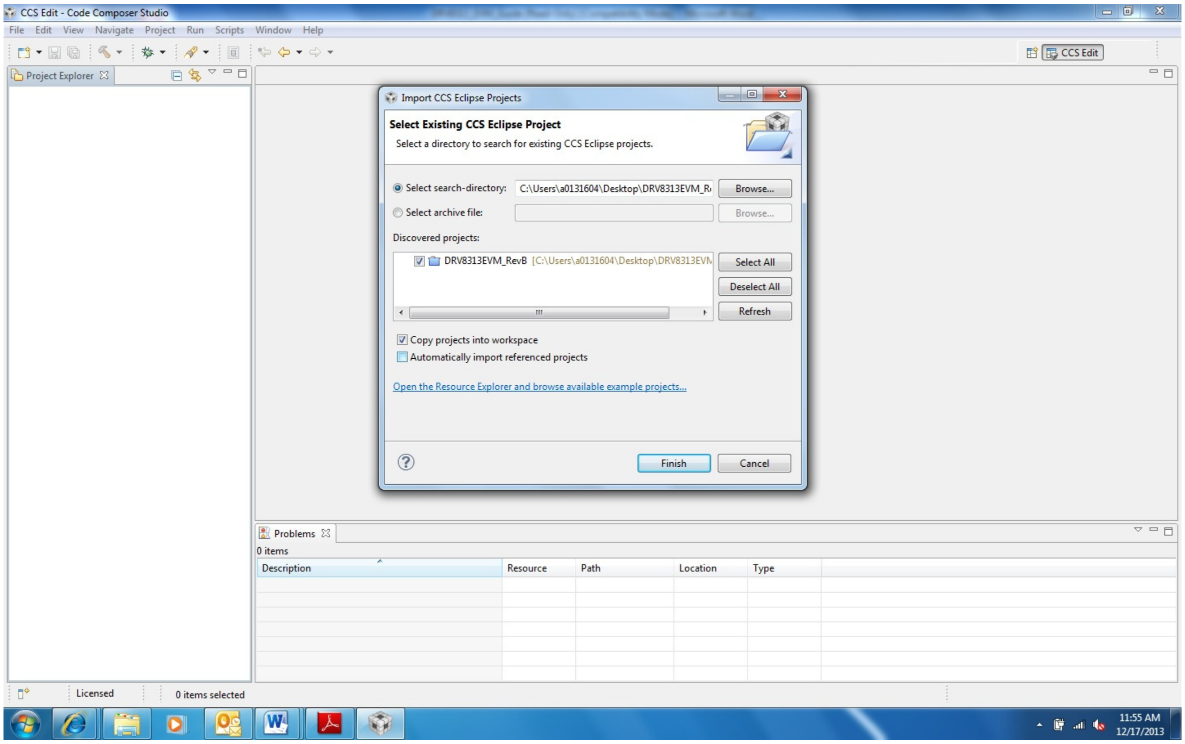
APPENDIX - B

MSP430 Programming using Code Composer Studio (V5)

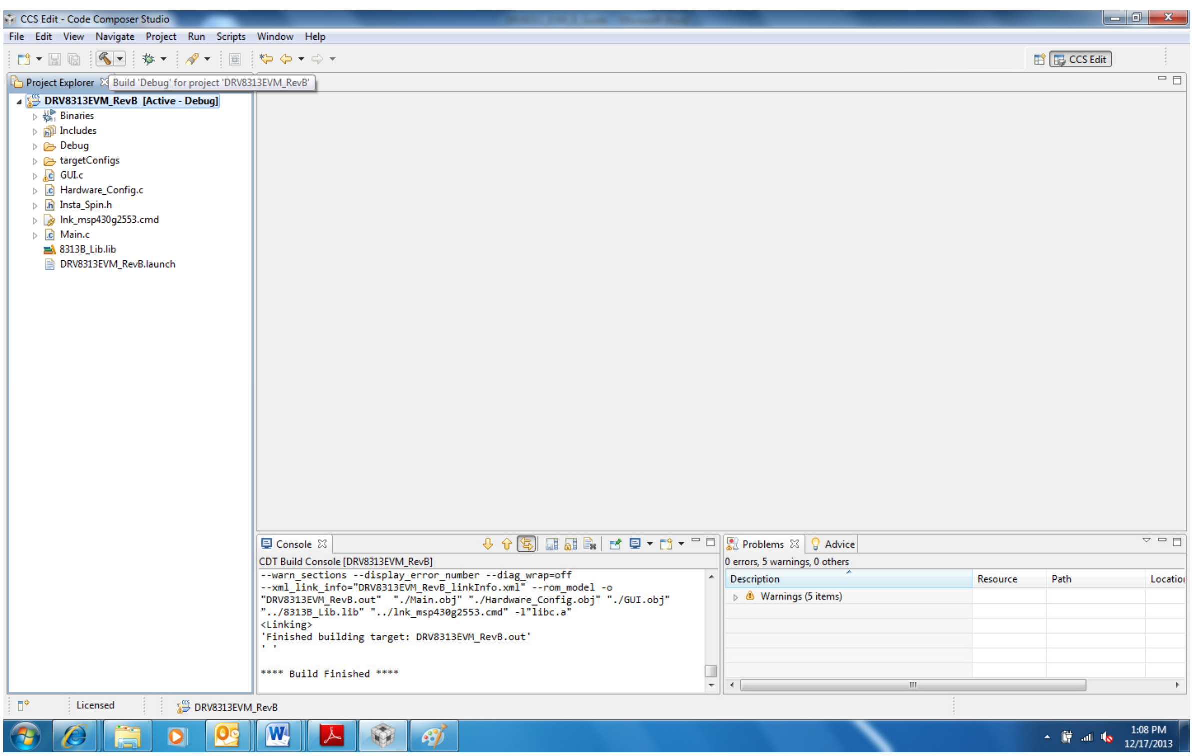
- After opening Code Composer Studio, under the **Project** menu, select **Import Existing CCS Eclipse Project**.



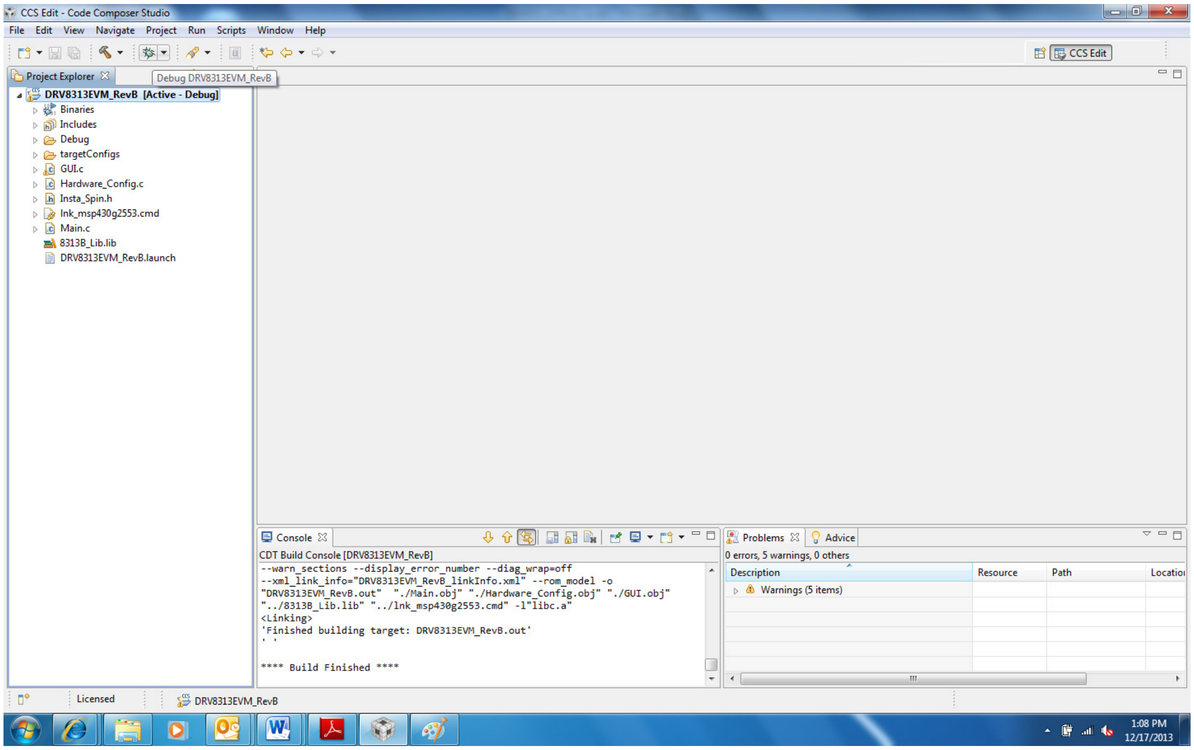
- Browse to the directory containing the firmware. Check the CCS project found and click on **Finish** button. You may choose to copy the project into workspace before this.



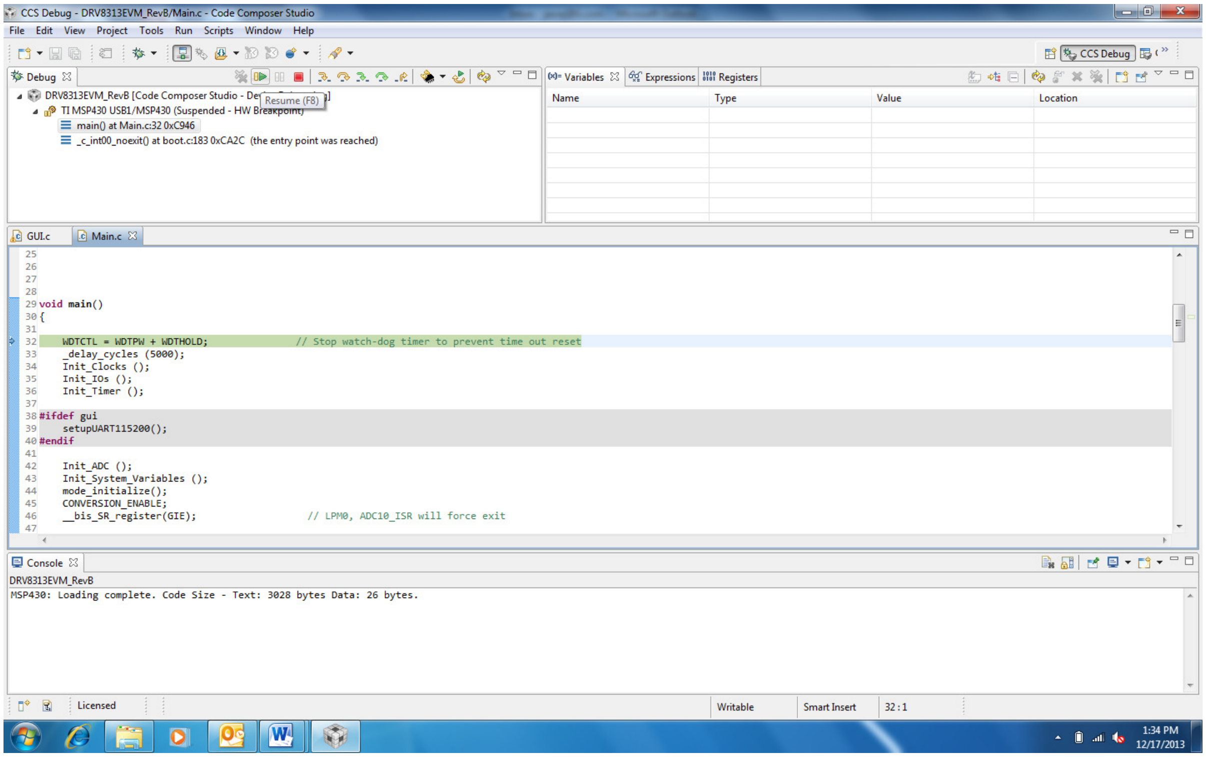
- Build the project by clicking on the title and then clicking the **Build** button as shown in the image below.



- Download the code onto the board by clicking the **Debug** button as shown in the following image.



- Once thedownload is successful, click the **Resume** button as shown in the image below.


CCS Debug - DRV8313EVM_RevB/Main.c - Code Composer Studio
File Edit View Project Tools Run Scripts Window Help
CCS Debug
Debug
DRV8313EVM_RevB [Code Composer Studio - De
Resume (F8)
TI MSP430 USB1/MSP430 (Suspended - HW Breakpoint)
main() at Main.c:32 0xC946
_c_int00_noexit() at boot.c:183 0xCA2C (the entry point was reached)
Variables
Expressions
Registers
Name
Type
Value
Location
GUI.c
Main.c
void main()
{
WDTCTL = WDTPW + WDTHOLD; // Stop watch-dog timer to prevent time out reset
_delay_cycles (5000);
Init_Clocks ();
Init_IOs ();
Init_Timer ();
#ifdef gui
setupUART115200();
#endif
Init_ADC ();
Init_System_Variables ();
mode_initialize();
CONVERSION_ENABLE;
__bis_SR_register(GIE); // LPM0, ADC10_ISR will force exit
Console
DRV8313EVM_RevB
MSP430: Loading complete. Code Size - Text: 3028 bytes Data: 26 bytes.
Licensed
Writable
Smart Insert
32 : 1
1:34 PM
12/17/2013

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