DRV832XX EVM Sensorless Software User's Guide

This document is intended as a supplement to the *BOOSTXL-DRV8320x User's Guide*, *BOOSTXL-DRV8323RX User's Guide*, and *DRV832xx GUI User's Guide* to describe the functionality of the sensorless BLDC motor commutation firmware used to on the BOOSTXL-DRV832x EVM. This user's guide outlines the different considerations for motor commutation as well as how to adjust the different code parameters provided in the sensorless firmware.

Contents

1 Overview 2
2 Sensorless Control Background and Implementation 2
2.1 Initial Speed Control 2
2.2 Initial Position Detection 2
2.3 Open-Loop Control 3
2.4 Closed-Loop Control 4
3 Customizing the Reference Code 5
3.1 Customizing ISC User Parameters 5
3.2 Customizing IPD User Parameters 6
3.3 Customizing ALIGN User Parameters 7
3.4 Customizing OPEN LOOP ACCELERATION User Parameters 7
3.5 Customizing CLOSED LOOP User Parameters 8
3.6 Customizing FAULT HANDLING User Parameters 12
3.7 Customizing SPI REGISTER User Parameters 13
4 Running the Project in Code Composer Studio 13

List of Figures

1 Six Pulses of Initial Position Detection 2
2 IPD 6 Current Pulses Across Phases 3
3 ADC Sampling Delay in Reading Phase Current 3
4 Open-Loop to Closed-Loop Threshold 4
5 ISC Brake Time 5
6 IPD_PULSE_TIME and IPD_ADD_BRAKE 6
7 IPD Coast Time 6
8 Open-Loop Acceleration 8
9 Observation of per Phase BEMF Waveform of the Motor When Motor is in Generating Mode 9
10 Calculation of BEMF Threshold Limit Using Trapezoidal BEMF Integration Method 9
11 Commutation Blank Time 10
12 PWM BLANK COUNTS 11
13 SPI REGISTER Setting Page in GUI 13

Trademarks

MSP430, Code Composer Studio, BoosterPack are trademarks of Texas Instruments.
All other trademarks are the property of their respective owners.

Copyright © 2017, Texas Instruments Incorporated

1 Overview

Driving a BLDC motor involves electronically commutating the phases. The windings must be energized in a sequence which makes knowing the rotor position important. In a sensorless control solution, the rotor position is initially detected using the properties of the motor. Then, after start-up, the rotor position is detected using the back electromotive force (BEMF) generated by the running motor. The DRV832XX EVM sensorless software uses a BEMF integration technique for sensorless control.

This user's guide is designed to show how sensorless control works and to enable users to modify the application software for a specific system. This document has two major sections. The first section is an introduction to sensorless control. The second section is an explanation of how to customize the reference code.

2 Sensorless Control Background and Implementation

2.1 Initial Speed Control

One feature of the reference code is to enable the firmware to stop the motor if the motor is already spinning. This is called initial speed control (ISC). ISC is used to check if a motor is already spinning and generating BEMF. If the observed BEMF is very high and if brakes are applied instantly, huge current spikes occur and potentially damage the switches. Therefore the motor is allowed to coast until the speed or BEMF is less than the switching on all the low-side gates so that the motor slows down and eventually stops without a huge current spike.

2.2 Initial Position Detection

To start up a motor, the rotor position must be known to enable the control to drive the correct phases without unwanted backspin. Traditionally using Hall Effect sensors, the rotor position can simply be read, setting the control sequence and spin up the motor. Removing the sensors from the control leads to the issue of not knowing where the rotor is. The remedy for this is initial position detection (IPD). IPD uses characteristics of the motor to detect where the rotor is before starting to spin.

The magnetic field created by the magnets on the rotor interacts with the electrical field generated by any current in the phases on the stator. Using this feature, measuring the time it takes for a current pulse to reach a specified level in all six drive states will indicate which phase has the lowest inductance, which is the position of the rotor. Figure 1 shows the six drive states. This method uses six pulses of current, one for each sequence, and measures the rise time of that current pulse.

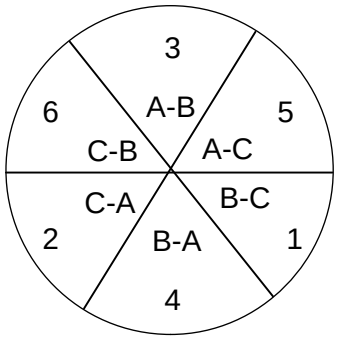


Figure 1. Six Pulses of Initial Position Detection

When the motor is energized with the current pulse, careful tuning is required to reduce any rotation of the motor. This tuning sets the current level to a value large enough to detect the change in rising times yet low enough to not begin to turn the rotor. Another step taken to reduce the movement of the rotor during IPD is to vary the pulse sequence to oppose any rotation. The numbers in Figure 1 specify the order of the IPD routine. Some audible noise may occur in the motor because of vibrations from the sudden current pulses.

Copyright © 2017, Texas Instruments Incorporated

As shown in Figure 1, each electrical rotation of the motor is split into six 60-degree drive states. The maximum torque is produced during drive if the excited phase is 90-degrees ahead of the rotor. Based on the minimum rise time, the rotor position is known to be in a 60-degree window. To reduce this window to 30 degrees, the two adjacent phases can be compared. For example, if sequence A-C was found to have the minimum rise time, comparing A-B and B-C rise times shows in which 30 degrees of A-C the rotor resides. From this, the drive state can be easily selected 90 degrees ahead of the rotor position.

To implement the IPD feature, phase-current sensing of the DRV832x device and the ADC unit of the MSP430F5529 device is used. A current impulse is given to a drive state for a predefined amount of time. After the pulse is withdrawn, the current in the phase is measured by using ADC channels. Timer A1 is used to count for the specified amount of time and sets an interrupt to start the ADC conversion. This time also turns off the high-side FETs and leaves the low-side FET on.

Leaving the low side on recirculates the current on the low side through the FET and one body diode, which is the motor braking. After braking for an established amount of time, the low-side FET is turned off and causes the rest of the energy to return to the supply through the high-side body diode, which is coasting. When the coasting period is complete and the energy is expelled, the next pulse begins repeating this sequence. Finally, the rise times are compared, and the initial drive state is handed off to open-loop control.

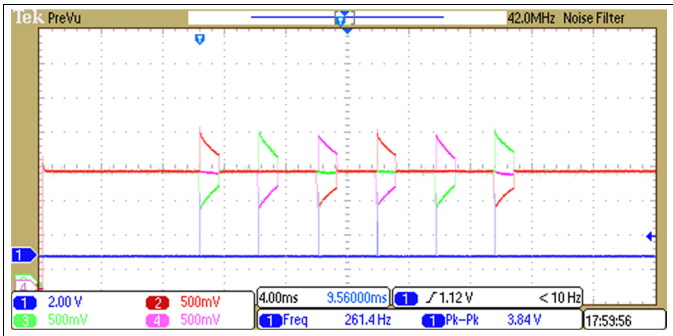


Figure 2. IPD 6 Current Pulses Across Phases

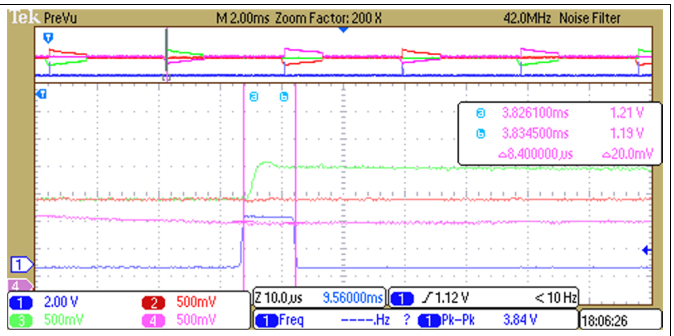


Figure 3. ADC Sampling Delay in Reading Phase Current

A slight delay occurs between turning off the phases and measuring the current values because of a delay in ADC sampling. This delay is negligible, and the amount of current drop can be ignored as shown in Figure 3. The phase current in the low-side switch represented by channel 4 (pink); the drop in the current value during sampling is observed to be negligible.

For some motors and some hardware setups, IPD is not a feasible start-up technique. In this case, a start-up method known as *align* is used. This start-up routine energizes two phases, one high and one low. This routine aligns the rotor to those phases. The phases are turned off after an established amount of time, therefore allowing any oscillations to settle. The rotor position is known at this time, and the drive state can move to a start-up.

2.3 Open-Loop Control

With the rotor position known, the next step is to begin driving the motor. However, the same problem persists; without sensors, the true position of the rotor is not known when it starts spinning. The IPD or align method handed off the initial drive state, and continuing to drive from there without feedback from the rotor position is open-loop control. When the motor reaches a certain speed, enough BEMF is available to use that measurement as feedback for the rotor position and move to closed-loop control.

Different methods of open-loop control are available, such as commutating a set number of times before handing off to a closed loop or increasing the established duty cycle to a fixed percentage before handing off. To make this system more robust, the method of open-loop control used in the reference code estimates the speed of the motor and from that, the distance traveled. Assuming the rotor follows the drive state, increasing the velocity at a fixed rate enables the control to estimate how far the rotor has traveled since the last commutation. When the distance calculation shows the rotor travels 60 degrees, the drive state is changed. When the speed reaches an established threshold, the control is switched to closed loop. To keep an accurate estimation of the distance and the speed, a timer is used to increase the velocity and calculate the distance. The same timer generates the PWM signal for the FETs.

Copyright © 2017, Texas Instruments Incorporated

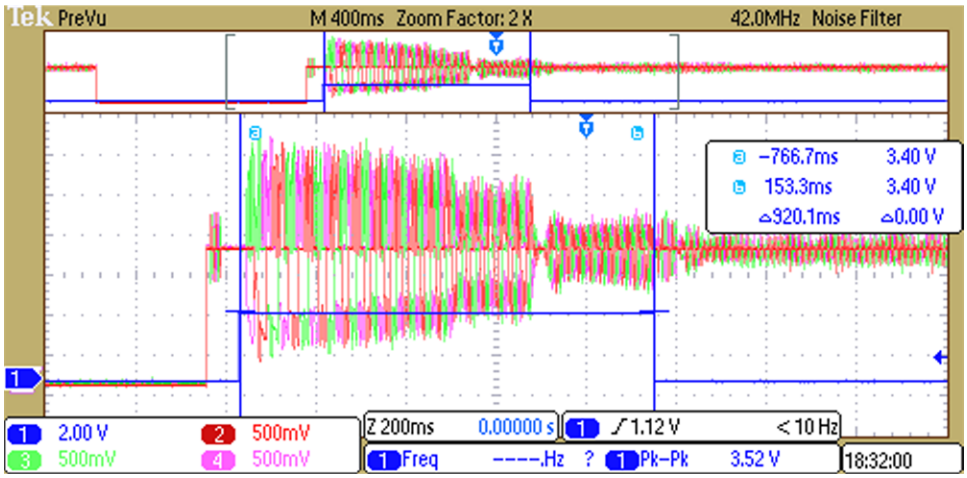


Figure 4. Open-Loop to Closed-Loop Threshold

2.4 Closed-Loop Control

When the motor spins fast enough, the BEMF can be read and used as position information. In this reference software, the BEMF integration method is used to control the motor. When the motor is driven, one high-side FET, and one low-side FET is on. One phase is floating with the high-side and low-side FET off. This phase is monitored with an ADC.

At the floating phases, the BEMF is either increasing or decreasing and eventually crosses the center tap value. To simplify the motor connections, the center tap is approximated as VCC / 2. BEMF integration begins to sample the BEMF after this crossing and integrates it. As the BEMF continues to increase or decrease, the integrated value continues to increase. When the integrated value reaches an established threshold, the drive state is switched.

The integration threshold is established based on the velocity constant of the motor in units of volts per hertz (V/Hz). As the motor is spinning faster, the BEMF is larger, and when the motor is slow, the BEMF is smaller. Because the speed of the motor, or the speed at which the drive state changes, relates directly to the BEMF level, the established integration threshold is constant across motor speed. See Section 3.5.1 to calculate the BEMF threshold for a specific motor.

To vary the speed of the motor, the duty cycle is adjusted based on the user input. As the duty cycle changes, the time of the high-side FET being turned on changes. The ADC reading the BEMF must be sampled on the high-side PWM pulse because, at this point, the center tap is closest to VCC / 2. The exact sample point of the ADC is set just before the falling edge of the PWM. Also, when a motor is switched, depending on the motor, some oscillation of the floating phase voltage occurs. To avoid sampling any of the oscillations, a settling time must occur before sampling the ADC. This window, after the oscillations but before the falling PWM edge, results in a minimum duty cycle allowed by this drive method. However, if the motor has very small oscillations in the floating phase, the settling time can be short, and the minimum duty cycle can be decreased.

A feature of a switching motor is that only a short time passes after the drive state changes until the current in the floating phase must be depleted. This feature results in conduction through the body diode of one of the FETs. To avoiding sampling the ADC during this time, a blanking time must occur after commutation. The reference code takes advantage of this blanking time by reading VCC and the duty cycle input.

The external DC supply of 0 to 3.3 V to ADC channel 6 is used as a duty cycle input. As previously mentioned, the duty cycle input is sampled during the commutation blanking time along with the supply being sampled. Because the ADC is on the MSP430™ microcontroller and settings must be changed before each sample, it takes advantage of the blanking time by using this time as a period to change the sampled ADC channel to the supply and the duty Input. Figure 4 shows two examples of closed-loop control with different duty cycle commands, channel 3 (the red trace) is the analog input for the duty cycle.

Copyright © 2017, Texas Instruments Incorporated

The timer that generates the PWM is also used to trigger the BEMF samples; a trigger point is a fixed number of clock cycles before the PWM falling edge. During a PWM event, an interrupt occurs which monitors all of the time-sensitive controls. The state machine takes advantage of flags set by the timer interrupt.

3 Customizing the Reference Code

To modify the sensorless code, the end-user must download the Code Composer Studio™ (CCS) software, BOOSTXL-DRV832X EVM_GUI, and DRV832XX_EVM_BLDC_FW software.

The following steps go through the process of modifying some parameters for sensorless control.

1. Open the CCS software and import the project: *DRV832XX_EVM_Trapezoidal_sensored_BLDC* from the folder where the demo software is located.
2. Select the file *TrapSensored_Parameters_Setup.h*. This folder contains most of the parameters used to run this application code. Some parameters require modifications to properly tune for different operating conditions. The sections that follow describe the parameters and the detail in which they can be modified. All of these parameters, except for the ISC parameters, can be varied using the GUI during run time for accurate tuning.

3.1 Customizing ISC User Parameters

The ISC user parameters follow:

```
//ISC User Parameters
#define ISC_MIN_BEMF            70
#define ISC_BRAKE_TIME          30
```

3.1.1 ISC_MIN_BEMF

This parameter is used to set the minimum phase BEMF above which motor is allowed to coast at the motor start. During start-up, if the motor is already spinning at high speeds, high BEMF is detected along the phases. Therefore, braking or applying the commutation state at such a point could lead to huge current spikes. So, the motor is allowed to coast until the motor-generated BEMF is less than ISC_MIN_BEMF. The speed of the motor is directly proportional to the amount of BEMF generated. Therefore setting a minimum BEMF-threshold value decides the speed below which the motor brakes are applied, and the motor will brake until the value set by ISC_BRAKE_TIME.

3.1.2 ISC_BRAKE_TIME

This parameter sets how long the motor brakes are applied if it is moving during motor start-up. If the ISC control determines if the motor is spinning, it applies motor brakes by turning on the low-side FETs. The ISC_BRAKE_TIME parameter sets the amount of time in milliseconds that motor brakes are applied. For motors with greater inertia, this number must be set to a higher value to allow the motor to stop completely. Smaller inertia can reduce this number for faster start-ups.

$$\text{ISC_BRAKE_TIME} = \text{ISC_BRAKE_TIME} \times 1\ \text{ms} \quad (1)$$

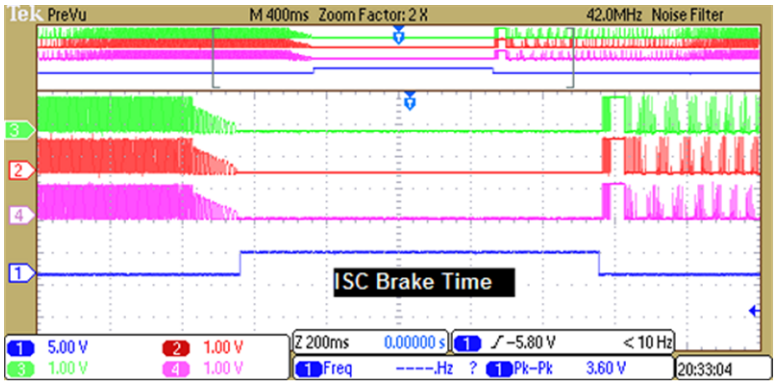


Figure 5. ISC Brake Time

Copyright © 2017, Texas Instruments Incorporated

3.2 Customizing IPD User Parameters

The IPD user parameters follow:

```
//IPD User Parameters
#define IPD_ADD_BRAKE            30
#define IPD_PULSE_TIME           3000
#define IPD_DECAY_CONSTANT       3
```

3.2.1 IPD_ADD_BRAKE

After the control energizes two phases with a pulse during IPD, that energy must be exhausted. This control method uses two modes to deplete the energy: braking the motor or turning on the low-side FET, and coasting the motor or turning off all the FETs. The IPD_ADD_BRAKE parameter specifies what additional length of time to the rise time is used to brake the motor. This number can be tuned to optimize the IPD control for a specific motor.

$$\text{ISC BRAKE TIME} = \text{ISC_BRAKE_TIME} \times 40\ \mu s \quad (2)$$

3.2.2 IPD_PULSE_TIME

This parameter sets the number of clock pulses for which a phase is excited and creates a rise in the phase current. This phase current is used for the current measurements in the IPD six step.

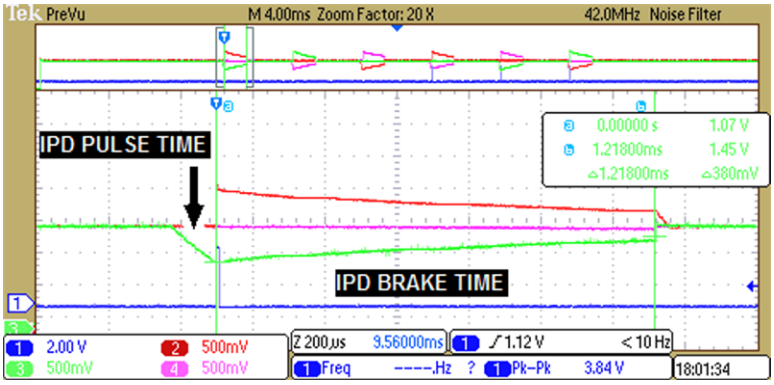


Figure 6. IPD_PULSE_TIME and IPD_ADD_BRAKE

3.2.3 IPD_DECAY_CONSTANT

After braking during IPD, the FETs are turned off; this is referred to as coasting. When the FETs are turned off, any remaining energy pumps back into the supply and the control begins the next IPD pulse. To allow some time for the remaining energy to escape before the next pulse, the IPD_DELAY_CONSTANT parameter sets the coast time to a multiple of the brake time.

$$\text{IPD Coast Time} = \text{IPD_DECAY_CONSTANT} \times \text{IPDBrakeTime} \quad (3)$$

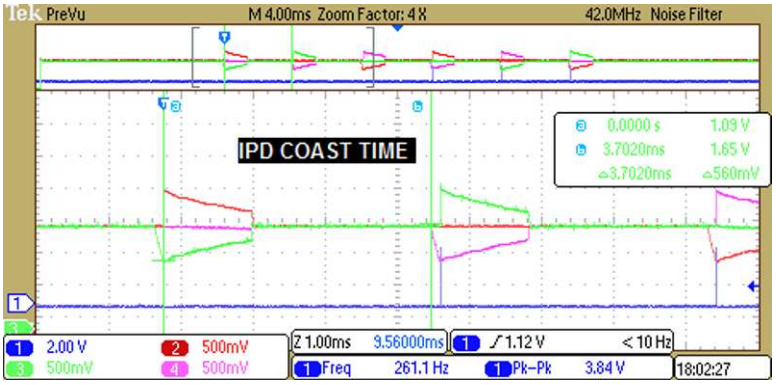


Figure 7. IPD Coast Time

Copyright © 2017, Texas Instruments Incorporated

3.3 Customizing ALIGN User Parameters

The ALIGN user parameters follow:

```
//Align User Parameters
#define ALIGN_SECTOR            1
#define ALIGN_WAIT_TIME         50
```

3.3.1 ALIGN_SECTOR

If the align function is used instead of initial position detection, the control energizes the commutation sequence associated to this ALIGN_SECTOR parameter. The rotor will start the align sector position when the control moves to open-loop acceleration.

3.3.2 ALIGN_WAIT_TIME

The ALIGN_WAIT_TIME parameter is how long the control will turn on the phases and hold it there for settling time. For a motor with larger inertia, this number must be larger to allow the rotor to settle in the specified location. Motors with smaller inertia that move to one position quickly and not oscillate back and forth can use a smaller number here. If the ALIGN_WAIT_TIME parameter is set too low the start-up can become unreliable because the rotor position is not correctly aligned.

3.4 Customizing OPEN LOOP ACCELERATION User Parameters

The OPEN LOOP ACCELERATION user parameters follow:

```
//Open Loop Acceleration User Parameters
#define ACCEL_RATE              40
#define ACCEL_STOP              50000
#define ACCEL_VELOCITY_INIT     10000
```

3.4.1 ACCEL_RATE

The ACCEL_RATE parameter defines how fast the motor accelerates during open-loop control. Specifically, this number is used to increase the velocity and distance calculation during open-loop control. The unit associated with this parameter is hertz per second (Hz/s) where hertz is the electrical speed of the motor. For a motor with greater inertia or that requires a longer time to accelerate, set this number to a small value such as 1. Motors that can ramp up quickly can use a larger value for ACCEL_RATE to decrease the start-up time.

3.4.2 ACCEL_STOP

The ACCEL_STOP parameter defines the velocity when the control transitions from open-loop control to closed-loop control. In open-loop control, each time the motor switches the commutation state, the control compares the calculated velocity with this number. The value is specified in millihertz (mHz) which is the electrical speed of the motor. For motors that produce a larger BEMF at low speeds, this number can be set low to decrease the start-up time. Conversely, if a motor must spin faster to produce sufficient BEMF for control, this number must be set higher.

3.4.3 ACCEL_VELOCITY_INIT

The ACCEL_VELOCITY_INIT parameter is used to set the initial speed during open-loop start-up. For some motors, the initial speed can be set high because the motor can spin up fast enough and is not required to start from 0. However, for motors with high inertia or that are difficult to start up, this number can be set to 0.

Copyright © 2017, Texas Instruments Incorporated

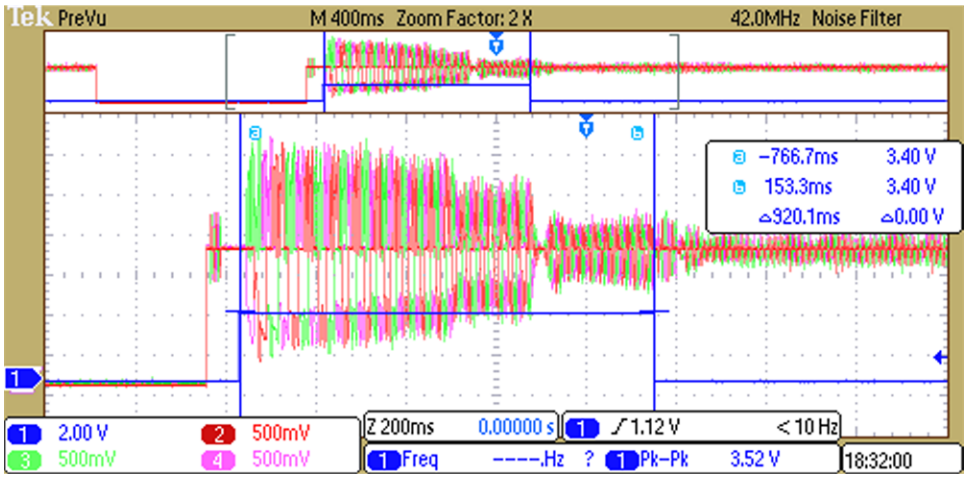


Figure 8. Open-Loop Acceleration

3.5 Customizing CLOSED LOOP User Parameters

The CLOSED LOOP user parameters follow:

```
//Closed Loop User Parameters
#define BEMF_THRESHOLD            1488
#define RAMP_RATE_DELAY           20
#define RAMP_RATE                 1
#define COMMUTATION_BLANK_TIME    2
#define PWM_BLANK_COUNTS          25

#define MAX_DUTY_CYCLE            1000
#define MIN_OFF_DUTY              250
#define MIN_ON_DUTY               260
#define START_UP_DUTY_CYCLE       250
#define PWM_FACTOR                0
```

3.5.1 BEMF_THRESHOLD

The motor is connected to a DRV832xx BoosterPack™ plug-in module and the phase voltage pins on the board are observed as shown in Figure 12. When the BLDC motor is run by the trapezoidal algorithm, the phase waveforms can be similar to waveforms in Figure 12. The BEMF of the floating phase is either increasing or decreasing and is available for sampling by the ADC. This BEMF of the floating phase is periodically sampled and added (a digital form of integration) until it reaches the commutation point. Because this commutation point is not known, it is reverse-identified by using the BEMF threshold limit.

Copyright © 2017, Texas Instruments Incorporated

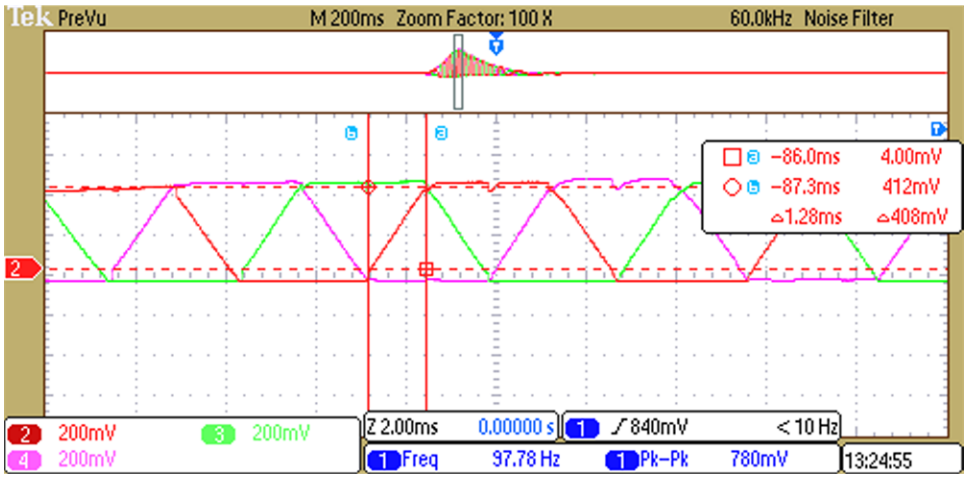


Figure 9. Observation of per Phase BEMF Waveform of the Motor When Motor is in Generating Mode

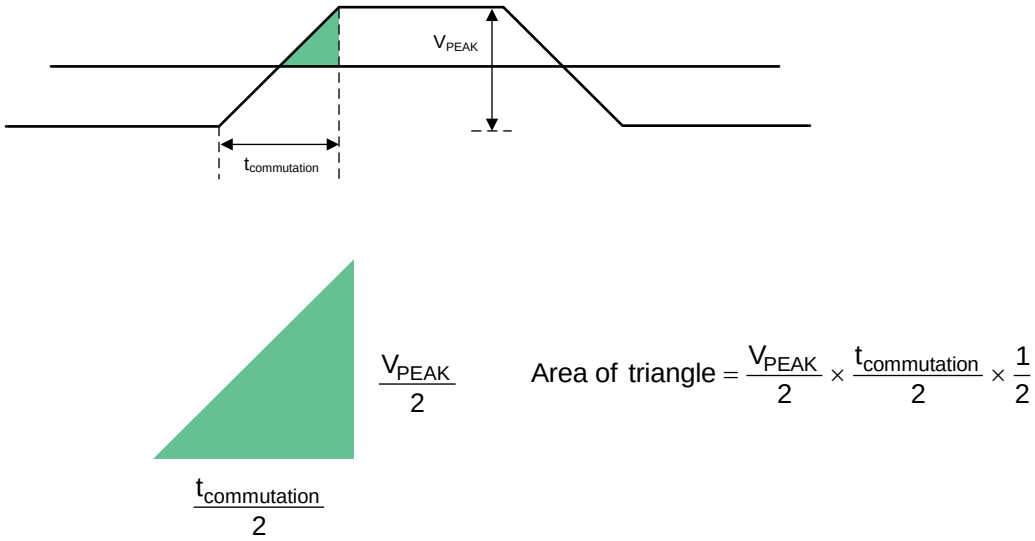


Figure 10. Calculation of BEMF Threshold Limit Using Trapezoidal BEMF Integration Method

Whenever BEMF is greater than VCC /2, the BEMF is sampled and summed up, which is equivalent to the area of the triangle shown in Figure 10. Therefore, whenever the summed up value crosses the area of the triangle (BEMF threshold limit), commutation of phases occurs.

From Figure 12, V_{PEAK} = 408 mV in digital counts 3.3 V, which is approximately 4096 counts; therefore 0.408 V = 506 counts with $t_{commutation}$ = 1.28 ms. As the ADC is sampled every 25 kHz or 41 µs (approximately), $t_{commutation}$ = 1.28 ms / 41 µs ≈ 31 samples. According to the FAULT HANDLING user parameters (see Section 3.6), the area of the triangle, or the BEMF threshold, equals (506 × 31) / 8 ≈ 1960.

3.5.2 RAMP_RATE_DELAY

The RAMP_RATE_DELAY parameter sets how many PWM_PERIOD interrupts must occur before adjusting the duty cycle. Changing this value changes how fast the duty cycle is adjusted. For example, if the PWM_PERIOD is 1024 or 40.96 µs and the RAMP_RATE_DELAY is 24, the duty cycle is adjusted every 983 µs. This parameter controls the acceleration and deceleration of the motor.

Copyright © 2017, Texas Instruments Incorporated

3.5.3 RAMP_RATE

The RAMP_RATE parameter indicates the amount of increase or decrease in the duty cycle for every PWM_PERIOD interrupt. If the system is either ramping up or down, and the acceleration count is reached, the duty cycle is increased or decreased by the RAMP_RATE parameter.

3.5.4 COMMUTATION_BLANK_TIME

When the closed-loop control switches the active phases of the motor, a short time must occur when the previous phase must deplete the energy built up in it. This depletion occurs as conduction through one of the body diodes of the FETs. To avoid this conduction being sampled in the control, a blanking time is available where the control does not monitor the BEMF. The COMMUTATION_BLANK_TIME parameter specifies the number of PWMs the control skips monitoring the BEMF. Figure 11 shows the brief period where the BEMF is blanked, indicated by the blue line going low. The blue trace is zero for a short period after every commutation.

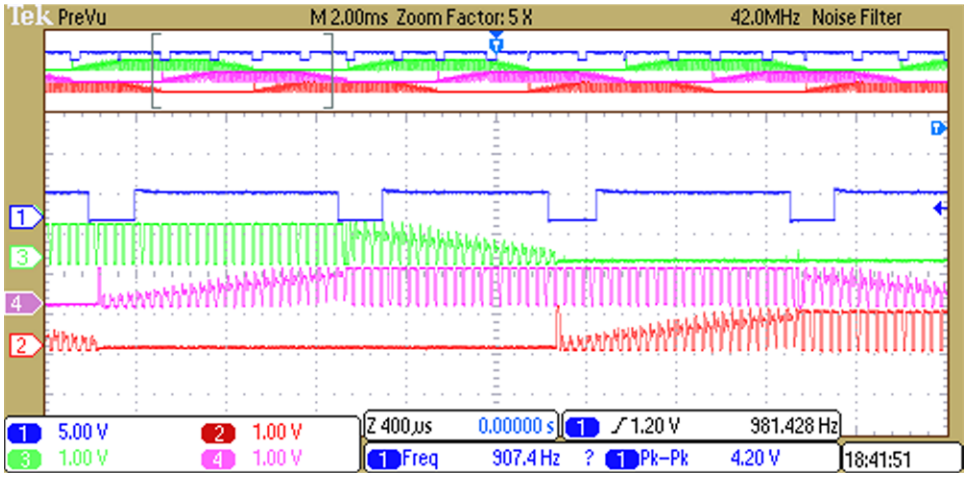


Figure 11. Commutation Blank Time

3.5.5 PWM_BLANK_COUNTS

The PWM_BLANK_COUNTS parameter sets a number of clock cycles before the center of the PWM edge that the BEMF ADC is sampled. The BEMF of some motors require a longer time to settle to the final value. This number is a trade-off between the minimum duty cycle allowed and the settling time of the BEMF. Because this number specifies clock cycles, use Equation 4 to calculate the time.

$$\text{Time before PWM edge} = \frac{\text{PWM_BLANK_COUNTS}}{25\ \text{MHz}} \quad (4)$$

In Figure 12 the PWM_BLANK_COUNT is 80 which is equivalent to 3.2 µs.

As shown in Figure 12, when the duty cycle is around 15 µs / 40 µs = 35%, until the BEMF value starts falling.

Copyright © 2017, Texas Instruments Incorporated

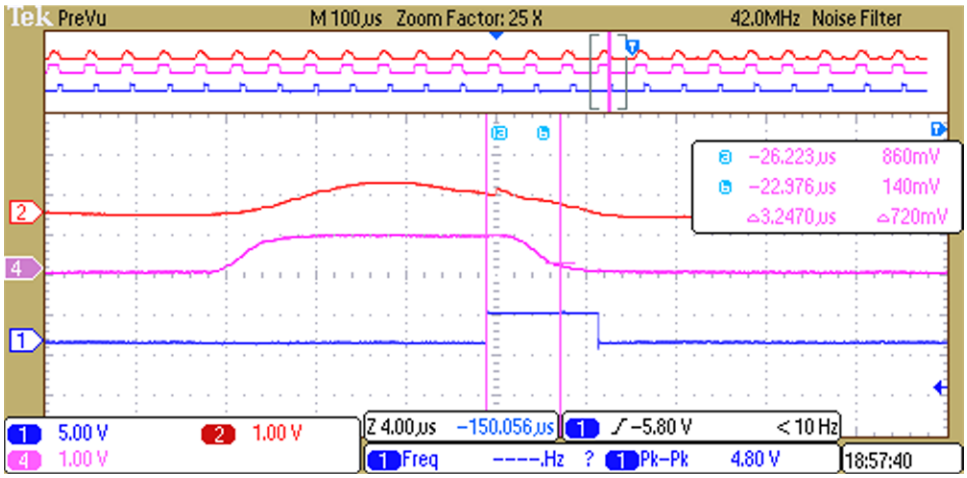


Figure 12. PWM BLANK COUNTS

3.5.6 MAX_DUTY_CYCLE

The MAX_DUTY_CYCLE parameter sets the maximum threshold that the input duty-cycle command can reach. Every time the input is read, the duty-cycle input command is compared to the MAX_DUTY_CYCLE parameter. If the duty-cycle command exceeds this parameter, the target duty cycle is set to the MAX_DUTY_CYCLE value.

$$\text{Maximum Duty Cycle (\%)} = \frac{\text{MAX_DUTY_CYCLE}}{\text{PWM_PERIOD}} \quad (5)$$

3.5.7 MIN_OFF_DUTY_CYCLE

The MIN_OFF_DUTY_CYCLE parameter sets the minimum threshold that the input duty-cycle command is allowed to reach after already starting. Every time the input is read, the duty-cycle input command is compared to the MIN_OFF_DUTY_CYCLE parameter. If the duty-cycle input command is below this parameter, the target duty cycle is set to 0.

$$\text{Minimum Off Duty Cycle (\%)} = \frac{\text{MIN_OFF_DUTY_CYCLE}}{\text{PWM_PERIOD}} \quad (6)$$

3.5.8 MIN_ON_DUTY_CYCLE

The MIN_ON_DUTY_CYCLE parameter sets the threshold that the input duty-cycle command must reach before starting the control. After initialization, the input duty-cycle command is read and compared to the MIN_ON_DUTY_CYCLE parameter. The control waits to continue until the input is greater than the value of MIN_ON_DUTY_CYCLE.

$$\text{Minimum On Duty Cycle (\%)} = \frac{\text{MIN_ON_DUTY_CYCLE}}{\text{PWM_PERIOD}} \quad (7)$$

3.5.9 START_UP_DUTY_CYCLE

During open-loop control, the duty cycle is fixed to the START_UP_DUTY_CYCLE parameter. This number can be adjusted for a specific motor to control how much current is used during start-up. This number is related to the PWM_PERIOD.

$$\text{Start-up Duty Cycle (\%)} = \frac{\text{START_UP_DUTY_CYCLE}}{\text{PWM_PERIOD}} \quad (8)$$

Copyright © 2017, Texas Instruments Incorporated

3.6 Customizing FAULT HANDLING User Parameters

The FAULT HANDLING user parameters follow:

```
/* Fault handling setup */
#define UNDER_VOLTAGE_LIMIT    (712)
#define OVER_VOLTAGE_LIMIT    (1424)
#define STALLDETECT_REV_THRESHOLD    (1)
#define STALLDETECT_TIMER_THRESHOLD    (200)
#define MOTOR_PHASE_CURRENT_LIMIT    (300)
#define AUTO_FAULT_RECOVERY_TIME    (3000)
```

3.6.1 UNDER_VOLTAGE_LIMIT

Voltage monitoring measures the VCC applied through the internal ADC and compares the ADC measurement with the specified limits. If the voltage is less than the specified UNDER_VOLTAGE_LIMIT value, the code shuts off the predrivers and the device goes into the FAULT state.

$$\text{UNDER_VOLTAGE_LIMIT} = \frac{\text{MinVCC (V)}}{60\ \text{V}} \times 4096 \tag{9}$$

3.6.2 OVER_VOLTAGE_LIMIT

Coupled with the UNDER_VOLTAGE_LIMIT parameter, if the voltage is found to be above the specified OVER_VOLTAGE_LIMIT value, the code shuts off the predrivers and goes into the FAULT state.

$$\text{OVER_VOLTAGE_LIMIT} = \frac{\text{MinVCC (V)}}{60\ \text{V}} \times 4096 \tag{10}$$

3.6.3 STALLDETECT_REV_THRESHOLD

In a certain amount of time, the motor should be spinning at least an established amount of revolutions. The number of revolutions is fixed by this parameter. In the set amount of time specified by the STALLDETECT_TIMER_THRESHOLD parameter, if the motor has not spun at least the count specified by this value, then the motor is assumed to have stalled.

3.6.4 STALLDETECT_TIMER_THRESHOLD

TimerB0 generates an interrupt service routine (ISR) every 1 ms and each ISR has a count that is increased. When the count reaches the STALLDETECT_TIMER_THRESHOLD value, if the current revolution count is less than the STALLDETECT_REV_THRESHOLD, the motor is stalled and the state machine goes into the FAULT state.

3.6.5 MOTOR_PHASE_CURRENT_LIMIT

The MOTOR_PHASE_CURRENT_LIMIT parameter defines the maximum allowed motor-phase peak current in amperes. The phase current of Motor A– is monitored every electrical cycle during commutation of phase A, and whenever the current limit is reached, an overcurrent (OC) fault is triggered. This value is scaled because it uses a 7-mΩ sense resistor on the DRV832XXEVM. The current-sense amplifier (CSA) gain 5 V/V is set using the firmware in the init section for better sensitivity. This value can be modified accordingly when higher values of current sensing are required. With an ADC of 3.3-V full-scale value and 12-bit resolution, use Equation 11 to calculate the overcurrent limit in digital counts.

$$\text{MOTOR_PHASE_CURRENT_LIMIT} = \frac{\text{Ampheres} \times 0.007\ \Omega \times 5\ \frac{\text{V}}{\text{V}}}{3.3\ \text{V}} \times 4096 \tag{11}$$

3.6.6 AUTO_FAULT_RECOVERY_TIME

TimerB0 is used to recover after a fault has been detected. FAULT_RECOVERY_TIME is the value used for how many timer interrupts must occur before reinitializing the system. For example, if the TimerB0 interrupt is set to occur every 1 ms and FAULT_RECOVERY_TIME is set to 3000, the system will reinitialize 3 s after a fault was detected.

Copyright © 2017, Texas Instruments Incorporated

3.7 Customizing SPI REGISTER User Parameters

For all the DRV832xS devices set the SPI register settings accordingly from the DRV832xS device data sheet (*DRV832x 6 to 60-V Three-Phase Smart Gate Driver*). Modify the register settings using register page found in the GUI.

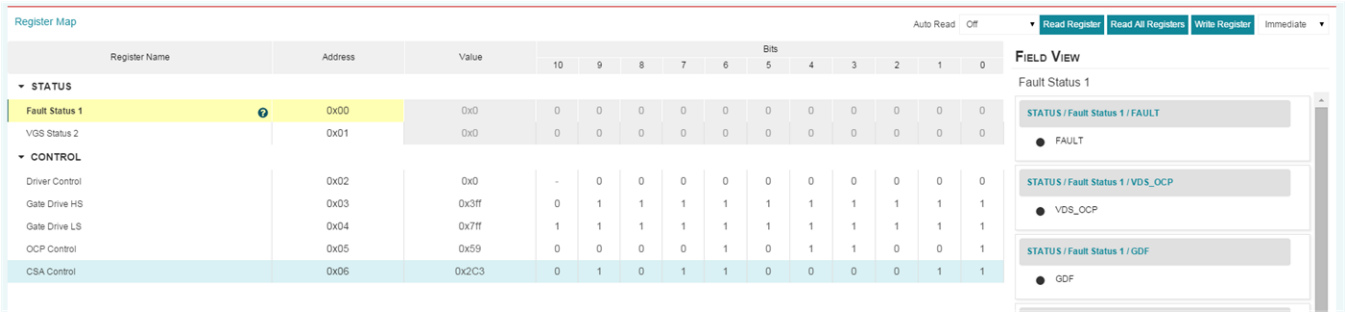


Figure 13. SPI REGISTER Setting Page in GUI

4 Running the Project in Code Composer Studio

To run the project in CCS, perform the steps that follow:

1. Install CCS software V6.1 or above.
2. Read through how to customize user parameters to tune the control for the specific motor.
3. Compile the modified project.
4. Connect the MSP430F5529 device to download and run the modified program.

The reference software was written for the Telco motor. If a different motor is used and the reference code is unable to spin the motor, the motor was most likely improperly tuned. To properly tune the motor parameters, see Section 3.

Copyright © 2017, Texas Instruments Incorporated

IMPORTANT NOTICE FOR TI DESIGN INFORMATION AND RESOURCES

Texas Instruments Incorporated ('TI") technical, application or other design advice, services or information, including, but not limited to, reference designs and materials relating to evaluation modules, (collectively, "TI Resources") are intended to assist designers who are developing applications that incorporate TI products; by downloading, accessing or using any particular TI Resource in any way, you (individually or, if you are acting on behalf of a company, your company) agree to use it solely for this purpose and subject to the terms of this Notice.

TI's provision of TI Resources does not expand or otherwise alter TI's applicable published warranties or warranty disclaimers for TI products, and no additional obligations or liabilities arise from TI providing such TI Resources. TI reserves the right to make corrections, enhancements, improvements and other changes to its TI Resources.

You understand and agree that you remain responsible for using your independent analysis, evaluation and judgment in designing your applications and that you have full and exclusive responsibility to assure the safety of your applications and compliance of your applications (and of all TI products used in or for your applications) with all applicable regulations, laws and other applicable requirements. You represent that, with respect to your applications, you have all the necessary expertise to create and implement safeguards that (1) anticipate dangerous consequences of failures, (2) monitor failures and their consequences, and (3) lessen the likelihood of failures that might cause harm and take appropriate actions. You agree that prior to using or distributing any applications that include TI products, you will thoroughly test such applications and the functionality of such TI products as used in such applications. TI has not conducted any testing other than that specifically described in the published documentation for a particular TI Resource.

You are authorized to use, copy and modify any individual TI Resource only in connection with the development of applications that include the TI product(s) identified in such TI Resource. NO OTHER LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE TO ANY OTHER TI INTELLECTUAL PROPERTY RIGHT, AND NO LICENSE TO ANY TECHNOLOGY OR INTELLECTUAL PROPERTY RIGHT OF TI OR ANY THIRD PARTY 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 products or services are used. Information regarding or referencing third-party products or services does not constitute a license to use such products or services, or a warranty or endorsement thereof. Use of TI Resources 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 RESOURCES ARE PROVIDED "AS IS" AND WITH ALL FAULTS. TI DISCLAIMS ALL OTHER WARRANTIES OR REPRESENTATIONS, EXPRESS OR IMPLIED, REGARDING TI RESOURCES OR USE THEREOF, INCLUDING BUT NOT LIMITED TO ACCURACY OR COMPLETENESS, TITLE, ANY EPIDEMIC FAILURE WARRANTY AND ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT OF ANY THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

TI SHALL NOT BE LIABLE FOR AND SHALL NOT DEFEND OR INDEMNIFY YOU AGAINST ANY CLAIM, INCLUDING BUT NOT LIMITED TO ANY INFRINGEMENT CLAIM THAT RELATES TO OR IS BASED ON ANY COMBINATION OF PRODUCTS EVEN IF DESCRIBED IN TI RESOURCES OR OTHERWISE. IN NO EVENT SHALL TI BE LIABLE FOR ANY ACTUAL, DIRECT, SPECIAL, COLLATERAL, INDIRECT, PUNITIVE, INCIDENTAL, CONSEQUENTIAL OR EXEMPLARY DAMAGES IN CONNECTION WITH OR ARISING OUT OF TI RESOURCES OR USE THEREOF, AND REGARDLESS OF WHETHER TI HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

You agree to fully indemnify TI and its representatives against any damages, costs, losses, and/or liabilities arising out of your non-compliance with the terms and provisions of this Notice.

This Notice applies to TI Resources. Additional terms apply to the use and purchase of certain types of materials, TI products and services. These include; without limitation, TI's standard terms for semiconductor products http://www.ti.com/sc/docs/stdterms.htm), evaluation modules, and samples (http://www.ti.com/sc/docs/sampterms.htm).

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265
Copyright © 2017, Texas Instruments Incorporated