User's Guide
SLVUB82–July 2017

DRV8350x-EVM Sensored Software User's Guide

This document is intended as a supplement to the *DRV8350x-EVM User's Guide*, and *DRV8350x-EVM GUI User's Guide* to describe the functionality of the sensored BLDC motor commutation firmware used to on the DRV8350X-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 sensored firmware.

NOTE: The scope captures on this document are from the BOOSTXL-DRV832x EVMs.

Contents

List of Figures

List of Tables

Trademarks

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 sensored control solution, the rotor position is detected by the outputs of Hall-effect sensors. Three sensors are placed on the motor giving a 3-bit code for the motor commutation sequence.

This user's guide is designed to show how sensored 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 the description of how sensored motor control works. The second section is an explanation of the reference code.

2 Sensored Control Background

2.1 Motor Position Detection

In applications equipped with Hall Effect sensors, the shaft position detection is simple. Each sensor output is directly wired to a GPIO pin of the microcontroller. The sensors give three 180° overlapping signals offset by 60° and therefore providing the six mandatory commutation points. The rising and falling edges of each sensor output represent a change in the drive state is needed. After the controller determines which edge has been detected, it computes the time elapsed since the last detected edge and commutates the respective phase.

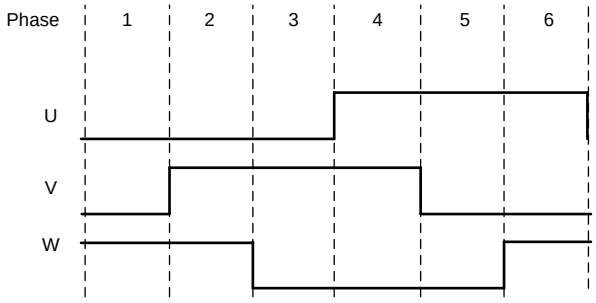

Figure 1. Hall Sensor Outputs

The sensor output is gray coded (see Table 1), so the sensor signal has only one edge change for each state change which reduces the noise sensitivity.

Table 1. Gray Coding of Hall sensors

Phase	Hall Inputs CW (WVU)	Active FETs
1	100	HS_W, LS_V
2	110	HS_W, LS_U
3	010	HS_V, LS_U
4	011	HS_V, LS_W
5	001	HS_U, LS_W
6	101	HS_U, LS_V

2.2 Commutation

With the position of the motor known, the next control state for the external FETs can be determined from the commutation sequence. The commutation sequence defines which windings of the motor the current flows through and which one is open. Commutating the phases occurs by the integrated predriver turning on and off certain low-side and high-side FETs. The PWM duty cycle is used to control the amount of current through the power FETs and motor windings. Adjusting the current to the motor, in turn, adjusts the speed of the motor.

Table 2 provides an example low-side commutation table. Rotating motor steps through this table during normal operation according to the Hall sensor signal that was received. If the motor is to spin in reverse, the commutation sequence is simply reversed.

Table 2. Commutation Sequence

Commutation Phase	Gate Drive Outputs						Open Phase	Hall Sensor State
	High-Side			Low-Side				
	U	V	W	U	V	W		
1			ON		PWM		U	100
2			ON	PWM			V	110
3		ON		PWM			W	010
4		ON				PWM	U	011
5	ON					PWM	V	001
6	ON				PWM		W	101

2.3 PWM Scheme

The commutation of the driving circuit is set in three different ways: low-side PWM, high-side PWM (non-symmetric PWM), and complementary PWM (using the built-in function called the dead time generator).

With the low-side PWM, only one high-side transistor is in the continuous ON state, and only one low-side transistor is driven by the PWM signal. With the high-side PWM scheme, the high-side transistor is driven by the PWM signal.

2.3.1 Low Side

Figure 2 shows the low-side PWM sequence.

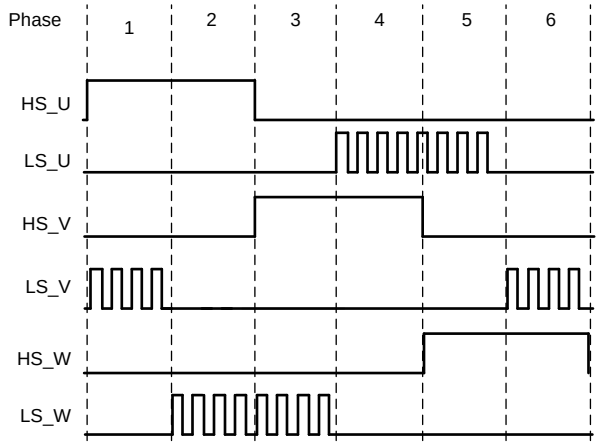

Figure 2. Low-Side PWM Sequence

2.3.2 High Side

Figure 3 shows the high-side PWM sequence.

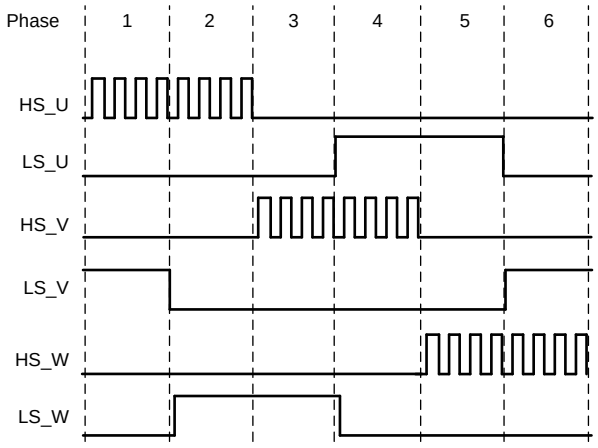

Figure 3. High-Side PWM Sequence

When the phase is driven with the PWM signal and when the PWM is in the off state of the duty cycle, the current still must flow in the motor. Typically the current can flow through the body diode on the FET until the PWM signal turns back on. To avoid any unnecessary power loss, the FET can be turned on instead of just letting the body diode conduct. For example in phase 1 in Figure 3, when the U high-side FET sees the PWM signal when the PWM is temporarily off the U low-side FET can be temporarily turned on. This method of alternating the PWM on the low side and high side of the same phase is known as synchronous PWM. A small amount of time must be inserted between when the high side turns on or off and when the low side turns off or on, respectively. This inserted time is known as dead-time.

2.3.3 Symmetric (Complementary)

Figure 4 shows the symmetric PWM sequence.

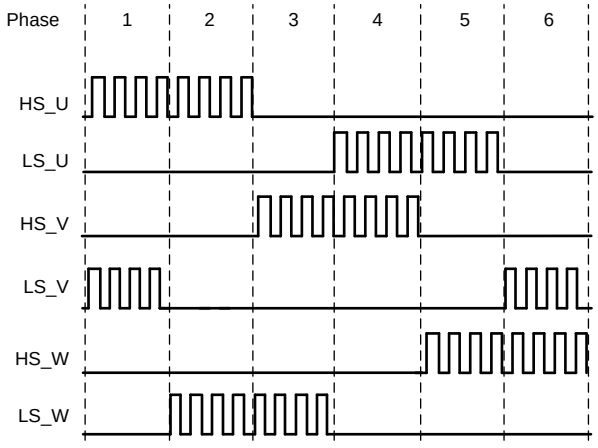

Figure 4. Symmetric PWM Sequence

3 Customizing the Reference Code

To modify the sensored code, the end-user must download the Code Composer Studio™ (CCS) software and the DRV8350x_EVM_BLDC_FW software. To modify the run time of the parameters, the user must install DRV8350x-EVM GUI.

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

1. Open the CCS software and import the project: *DRV8350x_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. Step 3 describes the parameters and the detail in which they can be modified.

```
/* System parameter setup */
#define ALGO_ID                       (0)
#define SPEED_INPUT_SAMPLE INTERVAL   (0)
#define PWM_FACTOR                    (0)
#define PWM_PERIOD                    (0)
#define READ_VCC_PERIOD               (0)
#define TIME_COUNT_1MS                (0)
#define MIN_DUTY_CYCLE                (0)
#define MAX_DUTY_CYCLE                (0)
#define RAMP_RATE                     (0)
#define RAMP_RATE_DELAY               (0)
/* Fault handling setup */
#define UNDER_VOLTAGE_LIMIT           (0)
#define OVER_VOLTAGE_LIMIT            (0)

#define MIN_STALLDETECT_DUTY          (0)
#define STALLDETECT_REV_THRESHOLD     (0)
#define STALLDETECT_TIMER_THRESHOLD   (0)
#define AUTO_FAULT_RECOVERY_TIME      (0)
#define STALL_DETECTION_FLAG          (0)
#define VCC_MONITOR_FLAG              (0)
#define MOTOR_PHASE_CURRENT_LIMIT     (0)
```

3. See the description for each parameter shown in Figure 5 in the following code comments:

ALGO_ID — This Parameter is used to identify the type of algorithm (sensored or sensorless) by the GUI and this value should not be changed.

SPEED_INPUT_SAMPLE INTERVAL — In the state machine, when the motor is in the RUN state, the speed Input must be read to get updated duty cycle inputs from the user. This parameter determines how often the speed input is read. This number × PWM_PERIOD gives the time Interval between each speed sample. This value cannot be configured and modified using GUI.

PWM_FACTOR — This parameter is the ratio of the ADC full-scale value to the PWM_PERIOD value. This value cannot be configured and modified using GUI.

PWM_PERIOD — This value sets the frequency of the PWM pulse train used in switching the FETs. The PWM frequency is calculated as the ratio of MCLK to PWM_PERIOD. As the master clock is operated at 25 MHz, if PWM_PERIOD = 1024, then PWM frequency = ~ 25 Khz (40 µs). This value also serves as the maximum comparator value that can be loaded that sets the duty cycle. This parameter can be configured using GUI widget PWM switching frequency.

READ_VCC_PERIOD — This number sets the time in milliseconds after which the supply voltage is periodically monitored for any voltage faults. This parameter cannot be configured through the GUI widget.

TIME_COUNT_1MS — This parameter is the equivalent clock counts for 1 ms at 25-MHz clock. This parameter cannot be configured through the GUI widget.

MIN_DUTY_CYCLE — This parameter sets the minimum threshold for the system to start spinning the motor and at what duty cycle. After the system initializes, it waits for an input command greater than this specified value before ramping up the motor. This value is about the PWM_PERIOD, for example, PWM_PERIOD is 1024 and MIN_DUTY_CYCLE is 128 decimal, then the minimum allowed duty cycle is 128 / 1024, or 12.5%.This value can be configured using the GUI widget during motor run time.

MAX_DUTY_CYCLE — This parameter sets the maximum value that the system uses as the duty cycle, therefore even if the input command is greater than this value, the duty cycle will not exceed this threshold. This value is also related to the PWM_PERIOD, for example, if PWM_PERIOD is 1024 and MAX_DUTY_CYCLE is 1000, then the maximum duty cycle is 1000 / 1024, or 97.6%. This value can be configured using the GUI widget during motor run time.

RAMP_RATE — This 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 RAMP_RATE. This value cannot be configured by the GUI widget.

RAMP_RATE_DELAY — This 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 motor. This parameter can be configured by the GUI widget.

UNDER_VOLTAGE_LIMIT — One feature of this code is voltage monitoring. 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.

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

MIN_STALLDETECT_DUTY — Some motors will spin very slowly at a low duty cycle. To prevent this condition from triggering a stall fault, a minimum duty cycle is required for the stall detection to be enabled. This parameter, MIN_STALLDETECT_DUTY, sets the threshold for the minimum duty cycle where the stall detection feature will be enabled.

STALLDETECT_REV_THRESHOLD — In a certain amount of time, the motor should be spinning at least a set amount of revolutions. This parameter sets the number of revolutions. 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.

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.

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.

STALL_DETECTION_FLAG — Setting the STALL_DETECTION_FLAG parameter to 1 enables the stall detection fault logic. If this parameter is set to 0, the stall fault detection is disabled.

VCC_MONITOR_FLAG — Setting the VCC_MONITOR_FLAG parameter to 1 enables the VCC undervoltage and overvoltage fault logic. If this parameter is set to 0, the VCC fault detection is disabled.

MOTOR_PHASE_CURRENT_LIMIT — The parameter defines the maximum allowed motor-phase peak current in Amperes. Motor A– phase current 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 DRV835xxEVM. 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 1 to calculate the overcurrent limit in digital counts.

$$\text{MOTOR\_PHASE\_CURRENT\_LIMIT} = \frac{\text{Amperes} \times 0.007\ \Omega \times 5\ \frac{\text{V}}{\text{V}}}{3.3\ \text{V}} \times 4096 \quad (1)$$

4. Customize the SPI REGISTER user parameters.

   For all the DRV8350S devices set the SPI register settings accordingly from the *DRV835x 100-V Three-Phase Smart Gate Driver* data sheet.

5. Modify the register settings using register page found in the GUI.

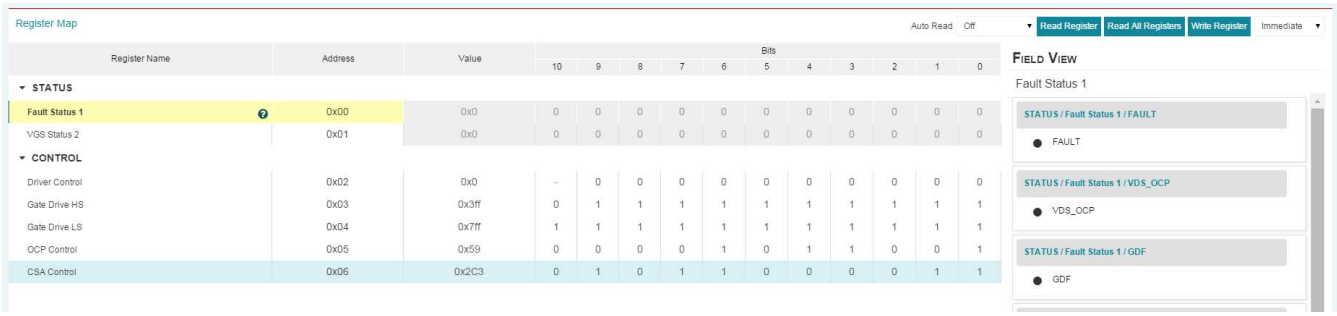

Figure 5. SPI REGISTER Page in GUI

4 Running the Project in Code Composer Studio

To run the project in CCS, use 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 micro-USB to download and run the modified program.
5. 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.

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale (www.ti.com/legal/termsofsale.html) or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

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