User's Guide
SLVUBC1 – January 2018

DRV8343x-Q1EVM Sensored Software User's Guide

This document is intended as a supplement to the *DRV8343H-Q1EVM and DRV8343S-Q1EVM User's Guide* to describe the functionality of the sensored BLDC motor commutation firmware used on the DRV8343x-Q1EVM. 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.

Contents

1 Overview 2
2 Sensored Control Background 2
2.1 Motor Position Detection Using Hall Sensors 2
2.2 Commutation 2
2.3 PWM Scheme 3
3 Customizing the Reference Code 4
4 Running the Project in Code Composer Studio 9

List of Figures

1 Hall Sensor Outputs 2
2 Low-Side PWM Sequence 3
3 High-Side PWM Sequence 4
4 Symmetric PWM Sequence 4
5 SPI REGISTER Page in GUI 8

List of Tables

1 Gray Coding of Hall sensors 2
2 Commutation Sequence 3

Trademarks

Code Composer Studio is a trademark of Texas Instruments.
All other trademarks are the property of their respective owners.

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 or an encoder. Three Hall sensors are placed on the motor giving a 3-bit code for the motor commutation sequence. A quadrature encoder is placed on the shaft to know the precise position of the rotor.

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 Using Hall Sensors

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.

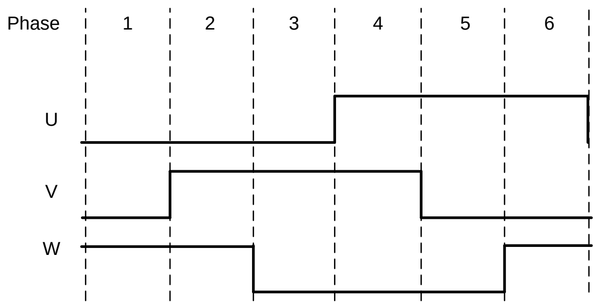


Figure 1. Hall Sensor Outputs

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	100b
2			ON	PWM			V	110b
3		ON		PWM			W	010b
4		ON				PWM	U	011b
5	ON					PWM	V	001b
6	ON				PWM		W	101b

2.3 PWM Scheme

The commutation of the driving circuit is set in three different ways: low-side PWM, high-side PWM (nonsymmetric 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.

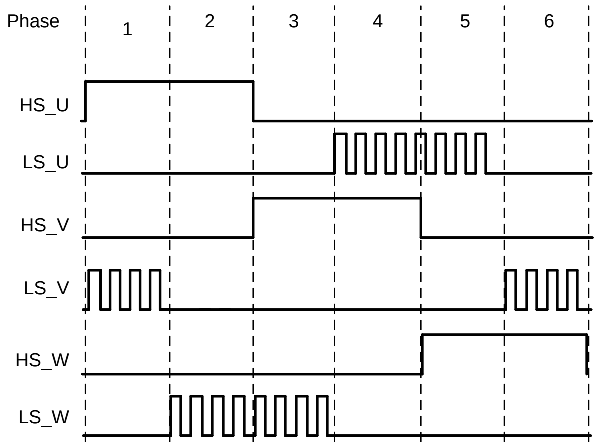


Figure 2. Low-Side PWM Sequence

2.3.2 High Side

Figure 3 shows the high-side PWM sequence.

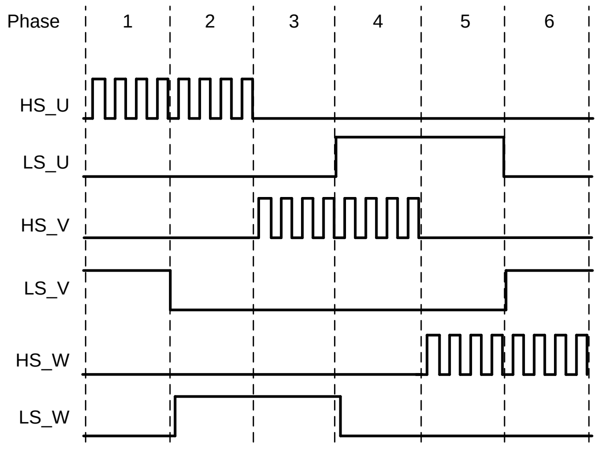


Figure 3. High-Side PWM Sequence

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 on again. 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.

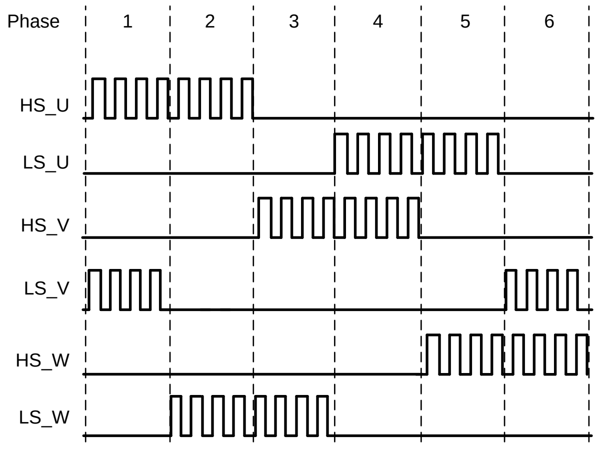


Figure 4. Symmetric PWM Sequence

3 Customizing the Reference Code

The reference code is provided as a Code Composer Studio™ (CCS) project and an Evaluation GUI.

The DRV8343x-Q1EVM GUI is a user interface (UI) to run and tune the motor on the DRV8343x-Q1EVM with the DRV8343_EVM_BLDC_FW software.

The user must install DRV8343x-Q1EVM GUI to run and modify the run time of the parameters for sensored Hall and encoder algorithm.

The user must download the Code Composer Studio (CCS) software 6.1.0 or above and install the DRV8343_MSP430F5529_Trap firmware.

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

1. Open the CCS software and import the *DRV8343_MSP430F5529_Trapezoidal_Sensored* project into the CCS workspace. The *Active Build* option from the *Build Configuration* should be set as *DRV8343_Sensored_Hall* from the folder where the demo software is located.
2. Select the file TrapSensored_Parameters_Setup.h from SensoredTrap_Hall\include. This file contains most of the parameters used to run this application code. Some parameters require modifications to properly tune the motor for different operating conditions. Step 3 describes the parameters and the detail in which they can be modified.

```
// USER DEFINES
/* Algorithm Version Settings */
#define ALGO_ID
    1       // 2 Indicates sensorless , 1 indicates sensored algorithms
#define PROD_ID                     1       // 1 Indicated AMD PL devices
#define FW_VER_MAJ                  1       // FW version major
#define FW_VER_MIN                  0       // FW version minor
#define FW_VER_PATCH                0       // FW version patch

/* Idrive & Tdrive Configuration Settings */
#define GATE_TO_DRAIN_CHARGE        8.7      // Set the gate to drain charge of the FET used
on the EVM in uC
#define RISE_TIME                   100     // Select the Maximum Rise time desired for the
FET in nano seconds
#define FALL_TIME                   50     // Select the Maximum Fall time desired for the FET
in nano Seconds


    /* System parameter setup */
#define SPEED_INPUT_SAMP_INTERVAL (127)         /* The number of PWM cycles after which change
in Speed input through ADC is Sampled */
#define PWM_FACTOR                (0)                             /* 12 bit ADC result
registers will be scaled to PWM period which is 10 bit  */
#define PWM_PERIOD (1000)                        /* PWM Period time , With a 25Mhz clock , PWM
will be generated at 25Khz*/
#define READ_VCC_PERIOD (100)                    /* TIME INTERVAL AFTER WHICH VCC IS MONITORED
(100ms) */
#define TIME_COUNT_1MS (25000)                    /* 1 ms timer count */
#define MIN_DUTY_CYCLE    (15)                     /* Minimal duty cycle applied to motor */
#define MAX_DUTY_CYCLE    (1000)                  /* Maximal duty cycle applied to motor */
#define CAL_DUTY_CYCLE    (50)                      /* Calibration duty cycle applied to motor
during Hall Calibration */
#define RAMP_RATE (1)                            /* Ramp rate for acceleration and
deceleration of the motor speed*/
#define RAMP_RATE_DELAY (100)                    /* How many PWM periods are between a change
of the speed */
#define HALL_CALIB_CYCLES (6000)                /* Defines number of PWM cycles Motor Hall
Calibration is Executed */

/*Stall Fault Detection related Parameter */
#define UNDER_VOLTAGE_LIMIT (10)                /* Under Voltage set for below the specified
volts - digital values are there by cal as (Limit*4095)/Full scale voltage  */
#define OVER_VOLTAGE_LIMIT    (20)                 /* Over Voltage set for above the specified
volts - digital values are there by cal as (Limit*4095)/Full scale voltage */
#define FULL_SCALE_VOLTAGE  (40)                /*As BEMF for above full scale volts may give
ADC ref volts exceeds nominal value 3.3V , MAX VCC is limited*/
#define MIN_POWER_SUPPLY    (5)                 /* Minimum power supply required to access the
SPI , when supply voltage is below this limit gate drivers are disabled */
#define MIN_STALLDETECT_DUTY        (125)        /* Minimum Duty Cycle above which stall will
be detected */
```

```
#define STALLDETECT_REV_THRESHOLD    (1)            /* Number of revolutions below which stall
fault will be flagged */
#define STALLDETECT_TIMER_THRESHOLD (500)        /* Time in milli seconds above which if motor
doesnt spin min revolutions specified above(STALLDETECT_REV_THRESHOLD) a stall fault is
triggered */
#define AUTO_FAULT_RECOVERY_TIME (6000)          /* Time in milli seconds after which system
reinitialises itself if fault gets cleared */
#define STALL_DETECTOIN_FLAG (1)                /* Enable motor stall fault detection*/
#define VCC_MONITOR_FLAG (1)                    /* Enable VCC Supply Voltage monitoring for
fault detection */
#define MOTOR_PHASE_CURRENT_LIMIT (434)         /* Defines the max allowed motor phase current
in digital counts. Motor phase current is monitored every electrical cycle , and when ever
current limit is reached an OC fault is Triggered */
```

3. See the description for each parameter listed in the code example:

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

GATE_TO_DRAIN_CHARGE:— This parameter defines the gate-to-drain charge or Q_{GD} in the microcontroller of the MOSFET used as an inverter. This value is used to compute the default IDRIVE and TDRIVE values as defined in the device data sheet.

RISE_TIME— This parameter defines the rise time of the MOSFET in nanoseconds used in the inverter. This value is used to compute the default IDRIVE and TDRIVE values as defined in the device data sheet.

FALL_TIME— This parameter defines the fall time of the MOSFET in nanoseconds used in the inverter. This value is used to compute the default IDRIVE and TDRIVE values as defined in the device data sheet.

CAL_DUTY_CYCLE— The calibration duty cycle sets the duty cycle for aligning the rotor with the applied vector during hall calibration. If the PWM_PERIOD is 1024 and CAL_DUTY_CYCLE is 30, the calibration duty cycle is 30 / 1024, or 3%. This value can be configured using the GUI widget during motor run time in the *Calibration Duty Cycle in Hall calibration* panel. If the motor is connected to the load, Increase the value of calibration duty along with calibration PWM cycles to detect the rotor Hall positions.

HALL_CALIB_CYCLES— The Hall calibration cycles parameter sets the number of PWM cycles for which the applied vector corresponding phases are energized. During Hall calibration, setting this parameter determines the amount of align time for a given vector. This value can be configured using the GUI widget during motor run time in the *Hall calibration Panel*.

SPEED_INPUT_SAMPLE INTERVAL — In the state machine, when the motor is in the RUN state, the speed input must be read to get the updated duty cycle inputs from the user. This parameter determines how often the speed input is read. This number times the PWM_PERIOD gives the time interval between each speed sample. This value cannot be configured and modified using the 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 the 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 the GUI widget for the 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 approximately the PWM_PERIOD, for example, the PWM_PERIOD is 1024 and the 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 the PWM_PERIOD is 1024 and the 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 and the duty cycle is increased or decreased by the 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 the 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.

$$\text{UNDER\_VOLTAGE\_LIMIT} = \frac{\text{MinVCC (V)}}{V_{FS}} \times 4096 \tag{1}$$

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.

$$\text{OVER\_VOLTAGE\_LIMIT} = \frac{\text{MinVCC (V)}}{V_{FS}} \times 4096 \tag{2}$$

FULL_SCALE_VOLTAGE— This value defines the maximum voltage that can be applied to the booster pack and can be sensed across the VSENSE resistance using the ADC (3.3 V maximum).

MIN_STALLDETECT_DUTY — Some motors 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.

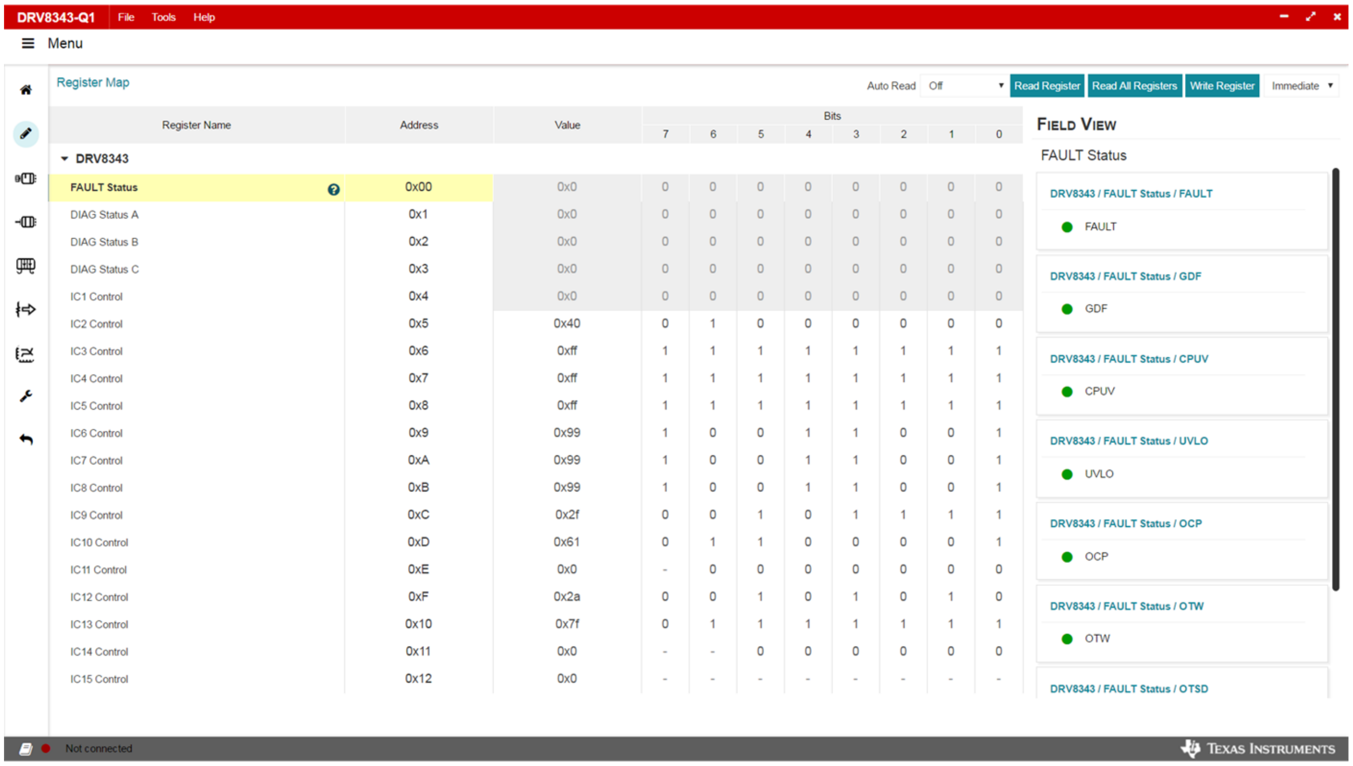
MOTOR_PHASE_CURRENT_LIMIT — The parameter defines the maximum allowed motor-phase peak current in Amperes. Motor A– phase current is monitored every electrical cycle during commutation of phase A, and whenever the current limit is reached an overcurrent (OC) fault is triggered. This value is scaled because it uses a 7-mΩ sense resistor on the DRV8343-Q1x EVM. The current sense amplifier (CSA) gain 5 V/V is set using the firmware in the Initialization 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 3 to calculate the overcurrent limit in digital counts.

$$\text{MOTOR\_PHASE\_CURRENT\_LIMIT} = \frac{\text{Amperes} \times 0.007\ \Omega \times 5\ \frac{\text{V}}{\text{V}}}{3.3\ \text{V}} \times 4096 \tag{3}$$

4. Customize the SPI REGISTER user parameters.

   For all the DRV8343s devices set the SPI register settings accordingly from the *DRV8343-Q1 Automotive 5.5 to 60-V Three-Phase Smart Gate Driver With Three Integrated Current-Shunt Amplifiers* data sheet.

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


DRV8343-Q1
File
Tools
Help
Menu
Register Map
Auto Read
Off
Read Register
Read All Registers
Write Register
Immediate
FIELD VIEW
FAULT Status
DRV8343 / FAULT Status / FAULT
FAULT
DRV8343 / FAULT Status / GDF
GDF
DRV8343 / FAULT Status / CPUV
CPUV
DRV8343 / FAULT Status / UVLO
UVLO
DRV8343 / FAULT Status / OCP
OCP
DRV8343 / FAULT Status / OTW
OTW
DRV8343 / FAULT Status / OTSD
Not connected
TEXAS INSTRUMENTS

Register Name	Address	Value	Bits							
			7	6	5	4	3	2	1	0
DRV8343										
FAULT Status	0x00	0x0	0	0	0	0	0	0	0	0
DIAG Status A	0x1	0x0	0	0	0	0	0	0	0	0
DIAG Status B	0x2	0x0	0	0	0	0	0	0	0	0
DIAG Status C	0x3	0x0	0	0	0	0	0	0	0	0
IC1 Control	0x4	0x0	0	0	0	0	0	0	0	0
IC2 Control	0x5	0x40	0	1	0	0	0	0	0	0
IC3 Control	0x6	0xff	1	1	1	1	1	1	1	1
IC4 Control	0x7	0xff	1	1	1	1	1	1	1	1
IC5 Control	0x8	0xff	1	1	1	1	1	1	1	1
IC6 Control	0x9	0x99	1	0	0	1	1	0	0	1
IC7 Control	0xA	0x99	1	0	0	1	1	0	0	1
IC8 Control	0xB	0x99	1	0	0	1	1	0	0	1
IC9 Control	0xC	0x2f	0	0	1	0	1	1	1	1
IC10 Control	0xD	0x61	0	1	1	0	0	0	0	1
IC11 Control	0xE	0x0	-	0	0	0	0	0	0	0
IC12 Control	0xF	0x2a	0	0	1	0	1	0	1	0
IC13 Control	0x10	0x7f	0	1	1	1	1	1	1	1
IC14 Control	0x11	0x0	-	-	0	0	0	0	0	0
IC15 Control	0x12	0x0	-	-	-	-	-	-	-	-

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.

The reference software was written for the 24-V Annaheim 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.