Application Note

BLDC Motor Control Using the SimpleLink™ CC2340R5 Microcontroller with Bluetooth® Low Energy

Ryan Brown and Tarek Darwiche *Connectivity LPCS*

ABSTRACT

This application note describes the capability of adding Bluetooth® Low Energy (LE) radio protocol to Brushless DC (BLDC) motor designs in a one microcontroller design. BLDC motors are commonly found in applications which can benefit from having embedded radios, including power drills, remote control (RC) cars, and electric toothbrushes. The material covered in this document shows how the CC2340R5 is capable of fulfilling this task with the DRV8329.

The implementation provided by this document uses hardware evaluation modules (EVMs) which are available on TI.com and firmware is provided for free on the SimpleLink Low Power F3 Demos GitHub. A description of both necessary hardware connections and the firmware operation are given in detail. This makes developers fully enabled to operate a demonstration and further modify the project to meet the requirements after obtaining a BLDC motor. Additional test data is also provided so that users can fully understand the operating conditions alongside options for application expansion.

Table of Contents

List of Figures

List of Tables

Trademarks

SimpleLink™, BoosterPack™, LaunchPad™, and Code Composer Studio™ are trademarks of Texas Instruments.
Bluetooth® is a registered trademark of Bluetooth SIG, Inc.
Arm® and Cortex® are registered trademarks of Arm Limited.
Wi-Fi® is a registered trademark of Wi-Fi Alliance.
Windows® is a registered trademark of Microsoft Corporation.
All trademarks are the property of their respective owners.

1 Introduction

The CC2340R5 is a powerful and low-cost microcontroller unit (MCU) with 512kB of flash and 36 or 64kB of SRAM, featuring a Arm® Cortex®-M0+ and 2.4GHz radio. This feature set is capable of achieving a multitude of end applications for a variety of radio protocols in a single-chip design. This application note highlights a single instance to prove the wider possibilities capable with this device.

Controlling a BLDC motor using a hall effect sensor trap design with the CC2340R5 through Bluetooth® LE radio communication is possible when coupled with a DRV8329. This document details the hardware and software implementations necessary to realize this application, and optional features which have been enabled. Through reading this document, users can learn more about both BLDC motor control and CC2340R5 development, and gain confidence to utilize similar concepts for designs.

1.1 CC2340R5

The CC2340R family is part of the SimpleLink™ MCU platform. This consists of Wi-Fi®, Bluetooth LE, Thread, Zigbee, Sub-1GHz MCUs, and host MCUs that all share a common, easy-to-use development environment with a single-core software development kit (SDK) and rich tool set. These devices are optimized for low-power wireless communication with Over the Air Download (OAD) support in building automation (wireless sensors, lighting control, beacons), asset tracking, medical, retail EPOS (electronic point of sale), ESL (electronic shelf), and personal electronics (toys, HID, stylus pens) markets.

The LP-EM-CC2340R5 development kit is used to speed up development with the SimpleLink Bluetooth LE MCU with support for Bluetooth 5 LE, and 2.4GHz proprietary protocols. Software support is provided by the SIMPLELINK-LOWPOWER-F3-SDK which can be built by using Code Composer Studio™ IDE. Features include access to all I/O signals with the BoosterPack™ plug-in module connectors and connecting the LaunchPad™ development kit to a smartphone using TI SimpleLink Connect. The LP-XDS110ET or LP-XDS110 debugger (sold separately) is required for programming, debugging, and RF evaluation.

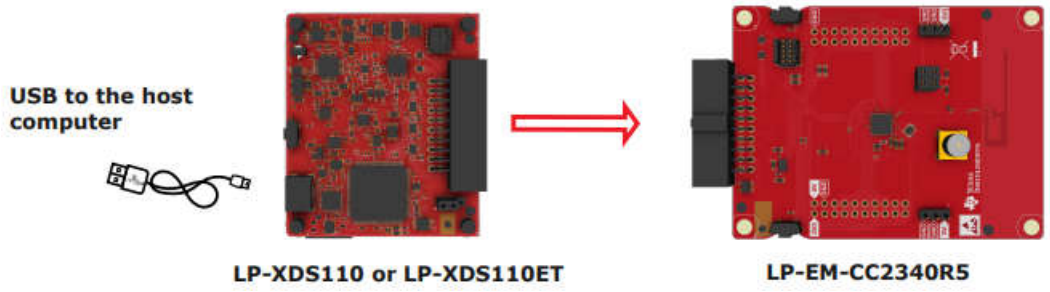

Figure 1-1. LP-XDS110ET and LP-EM-CC2340R5 Connections

1.2 DRV8329A

The DRV8329 family of devices is an integrated gate driver for three-phase applications. The devices provide three half-bridge gate drivers, each capable of driving high-side and low-side N-channel power MOSFETs. The device generates the correct gate drive voltages using an internal charge pump and enhances the high-side MOSFETs using a bootstrap circuit. A trickle charge pump is included to support 100% duty cycle. The gate drive architecture supports peak gate drive currents up to 1A source and 2A sink. The DRV8329 can operate from a single power supply and supports a wide input supply range of 4.5 to 60V (DC).

The DRV8329AEVM is a 30A, 3-phase brushless DC drive stage based on the DRV8329A gate driver for BLDC motors. The DRV8329 incorporates three diodes for bootstrap operation without the need for external diodes. The device includes a current shunt amplifier for low-side current measurement, 80mA low-dropout regulator (LDO), dead-time control pin, VDS overcurrent level pin, and gate driver shutoff pin. The EVM includes switches, potentiometers, and resistors to evaluate these settings and configurability for the A variant (6x PWM) and B variant (3x PWM) of the DRV8329 device. Up to 60V can be supplied to the EVM, and the DRV8329 integrated LDO generates the gate voltage needed for the bootstrap GVDD supply. Status LEDs and a fault LED for all power supplies are included for user feedback.

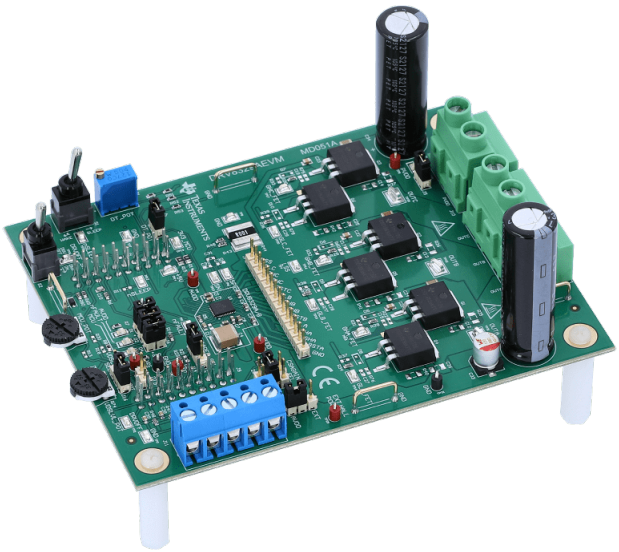

Figure 1-2. DRV8329AEVM

1.3 BLDC Motor

A brushless DC (BLDC) electric motor is a synchronous motor using a direct current (DC) power supply. These require a controller to switch DC currents to the three motor stators which produce magnetic fields to rotate the permanent magnet. In this document, the interface circuit drives six PWMs to the high and low side of three separate phases in a unipolar drive fashion. The application note focuses on using hall-effect sensor design to measure the rotor position and transitions along the commutation table accordingly. As such, a motor with hall effect connections is necessary to run the project as intended. Further modifications are required to support basic BLDC motors without hall effect sensors.

Figure 1-3. BLDC Motor With Hall-Effect Sensors

2 BLDC Hardware

2.1 Hardware Setup

The following sections describe the hardware, which must be procured, setup changes implemented on the EVMs, and the necessary connections required to run the example without any modifications to the default firmware design.

2.1.1 DRV8329AEVM Settings

Figure 2-1 is a depiction of the DRV8329AEVM board with jumpers populated in the correct places. Also provided are some comments about how the board switches and sliders are positioned to avoid issues when trying to operate the BLDC motor. Please refer to the DRV8329AEVM User's Guide for more instructions on how to interface with this hardware.

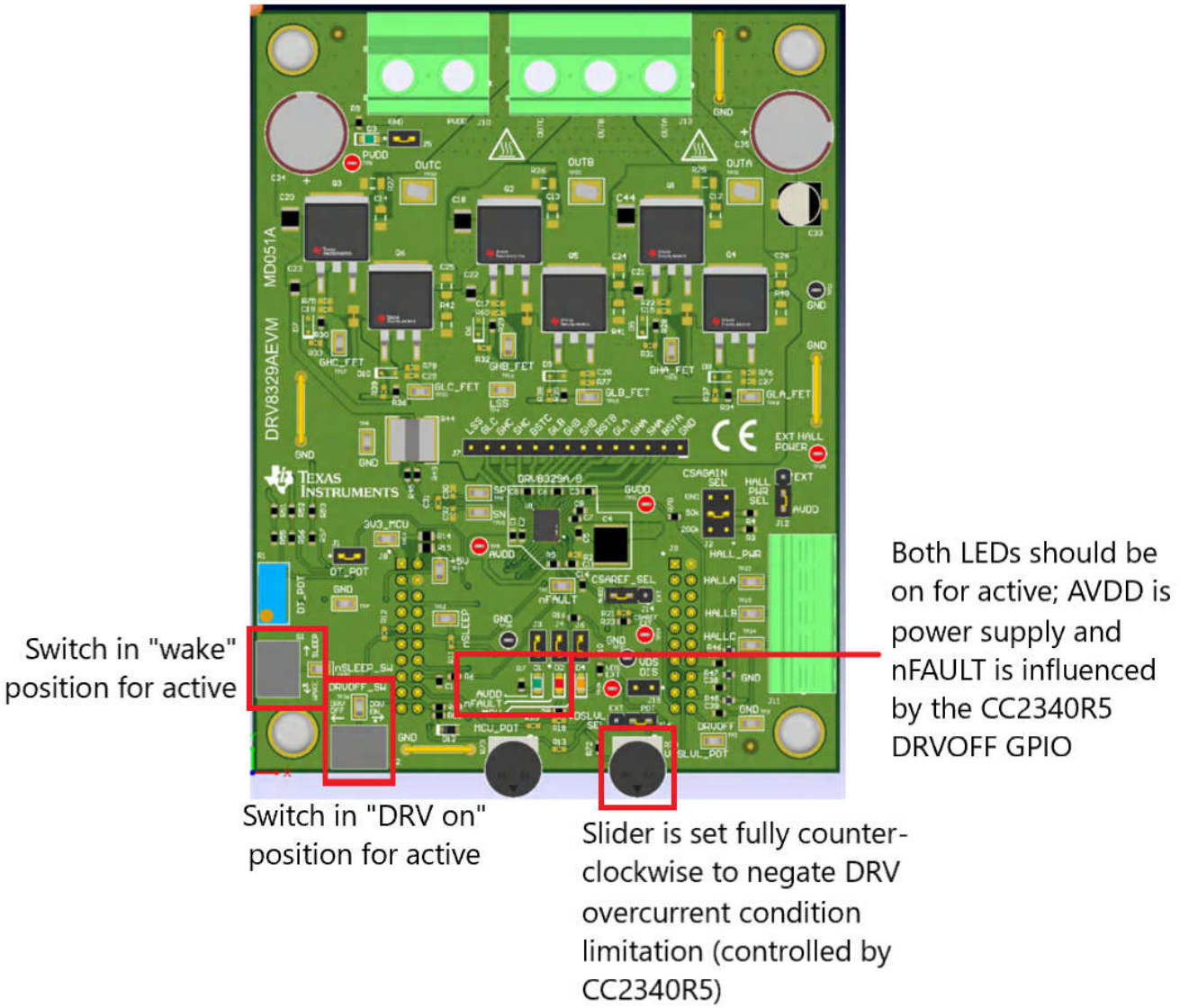

Figure 2-1. DRV8329A Hardware Setup

2.2 Connection Diagram

Table 2-1 is the connection between the DRV8329AEVM and CC2340R52 to realize the BLDC motor demonstration.

Table 2-1. Connections Between the CC2340R5 and DRV8329AEVM

Connection	CC2340R5 Function	CC2340R5 Pin	DRV8329AEVM
Phase A HS	PWM output	DIO24	INHA
Phase A LS	PMW output	DIO8	INLA
Phase B HS	PWM output	DIO12	INHB
Phase B HL	PWM output	DIO21	INLB
Phase C HS	PWM output	DIO6	INHC
Phase C LS	PWM output	DIO11	INLC
HALL A	Digital interrupt input	DIO23	HALLA
HALL B	Digital interrupt input	DIO18	HALLB
HALL C	Digital interrupt input	DIO13	HALLC
Bus voltage	ADC input	DIO7	VSENPVDD
Phase A voltage	ADC input	DIO1	VSENA
Phase B voltage	ADC input	DIO2	VSENB
Phase C voltage	ADC input	DIO5	VSENC
Shunt current	ADC input	DIO0	ISENA
DRV fault	Digital interrupt input	DIO14	nFAULT_49C
UART TX	UART transmit output	DIO20	N/A
UART RX	UART receive input	DIO22	N/A
Common GND	GND connection	GND	GND

The BLDC motor wires must be connected to the specified DRV8329AEVM pins. The necessary motor wires and the corresponding pin connections are listed in Table 2-2. For a specific motor, verify which wires implement these functions and connect them accordingly

Table 2-2. Connections Between the BLDC Motor and DRV8329AEVM

Motor Wire	DRV8329AEVM Connector	Motor Image Example Color
VCC	HALL_PWR	Red
HU/PH1	HALLA	Orange
HV/PH2	HALLB	Yellow
HW/PH3	HALLC	Blue
GND	GND	Black
U	OUTA	Orange
V	OUTB	Yellow
W	OUTC	Blue

The end result looks similar to Figure 2-2. Note that the LED header jumpers have been removed from the LP-EM-CC2340R5 as these GPIOs are used for other purposes. Once a valid power source has been supplied through J10 on the DRV8329AEVM, both toggle switches S1 and S2 must be oriented to the *wake* and *on* positions, respectively, so that the green AVDD (D1) and red nFAULT (D2) LEDs are lit.

The nFAULT LED clears whenever the CC2340R5 enables the DRV motor driver by setting the DRVOFF pin high. The nSLEEP and DRVOFF lines are inputs to the DRV8329A which is controlled by EVM toggle switch hardware, but can optionally be controlled by the CC2340R5 through further firmware development.

The CC2340R5 DIO6 pin is used as a PWM to control the BLDC motor, however the LP-EM-CC2340R5 also connects this pin to nCS of the on-board external flash device. This causes additional power consumption unless R24 and R25 on the LaunchPad are depopulated.

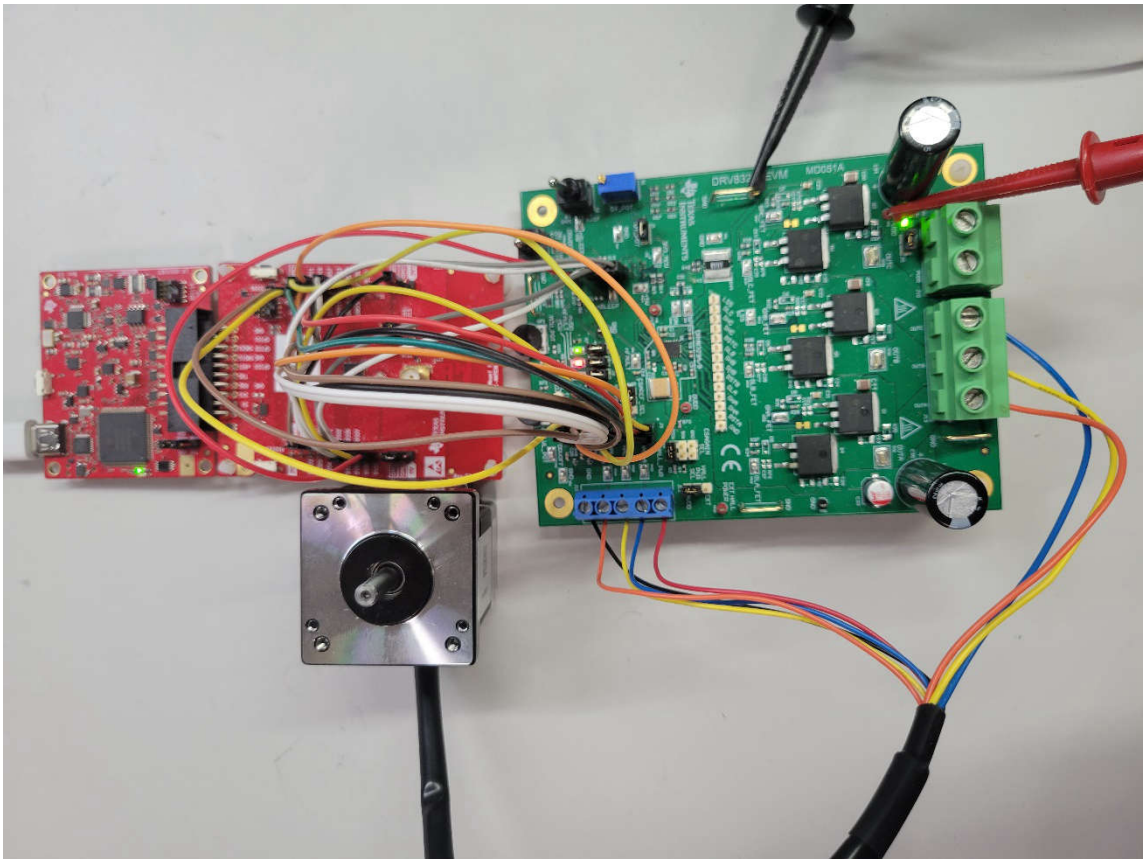

Figure 2-2. Physical Hardware Setup

3 Running the Example

The next sections describe the firmware details and how each component works towards driving the BLDC motor and collecting data.

3.1 Dependencies

The code project supplied on the SimpleLink Low Power F3 Demos GitHub uses SimpleLink F3 SDK v8.40.0.61, SYSCONFIG, and the TI CLANG v4.0.0 compiler. Make sure that all of these dependencies are installed on the machine before attempting to import the project into Code Composer Studio™ (CCS) v20 or later. For more examples for setting up an environment, refer to SimpleLink Academy for CC23xx. Note, that users are responsible for migrating and supporting any dependency versions not listed above.

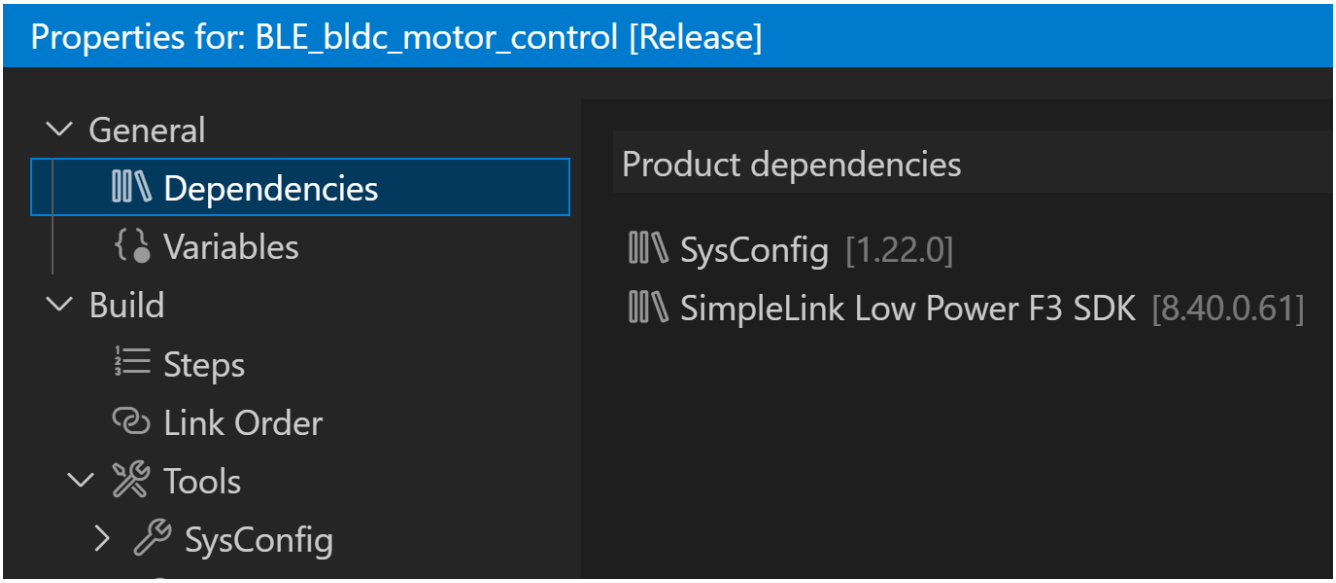

Figure 3-1. CCS Properties

3.2 Loading Firmware

Projects built inside of CCS can be loaded either directly in this IDE by selecting Run -> Flash Project (Ctrl + F5) or Debug Project (F5). TI recommends to exit Debug Mode to allow free-running if the project is not actively being debugged. Consider using the UNIFLASH software tool to load binary images.

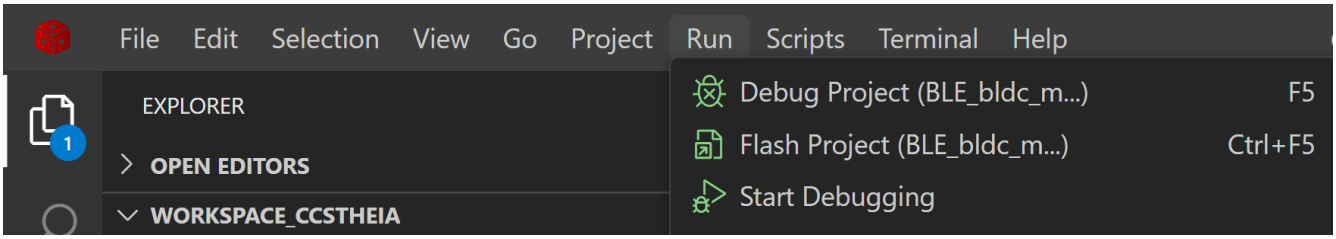

Figure 3-2. CCS Load Options

3.3 Motor Connection Test

A test firmware setup is provided with the project to test the orientation of the motor phase connections on the DRV8329AEVM (OUTA, OUTB, OUTC) and monitor the hall effect sensor interrupts (HALLA, HALLB, HALLC) to determine whether the connection orientation is correct. This happens by outputting both the phase table and hall sensor positions through a UART terminal which uses the following settings: 921600 baud, 8 data bits, 1 stop bit, no parity, and no flow control. The COM Port is the same as the Application UART displayed in the computer system. PuTTY software is used on a Windows® operating system in the example below, but any UART terminal or operating system is acceptable so long as the correct settings are used.

Note that UART must be fully disabled in the application to allow for standby power consumption while the motor is not spinning, otherwise active mode is required to constantly monitor the RX pin. UART can be removed from application code by undefining USE_UART in the *app_bldc_motor_control.c* file.

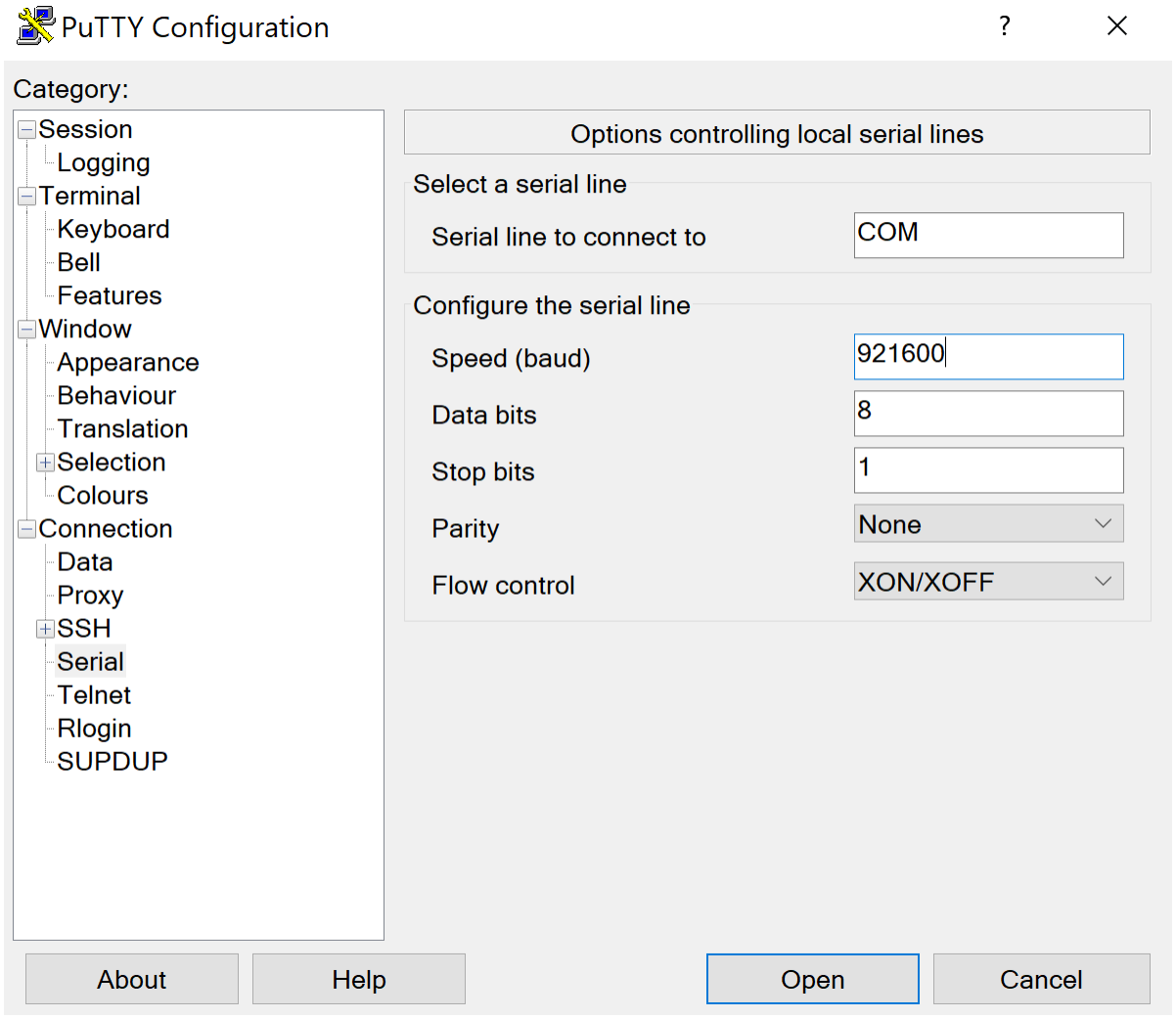

Figure 3-3. PuTTY UART Options

To enable this option, add TEST_NO_HALL inside of the Project Properties -> Build -> Tools -> Arm Compiler -> Predefined Symbols. Then, rebuild and load the project.

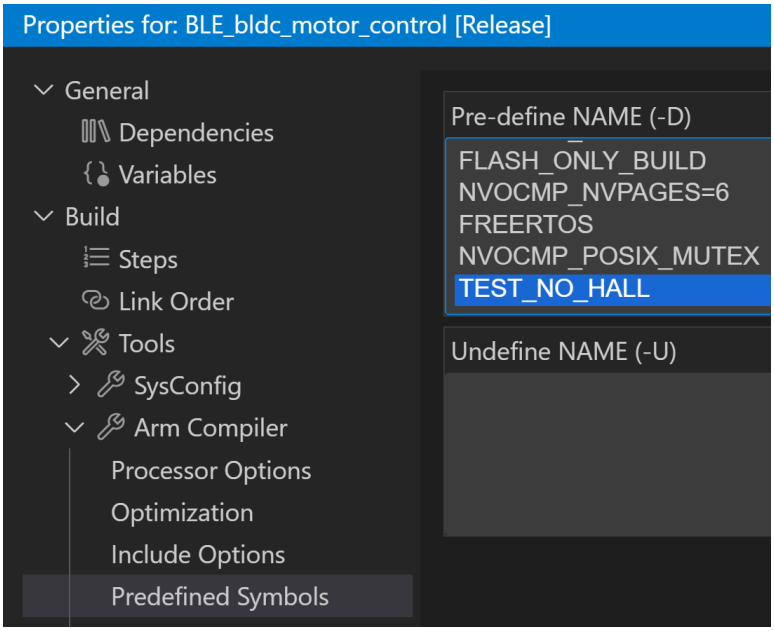

Figure 3-4. Predefined Symbols

Once the BLDC motor main menu is displayed, press *s* on the keyboard (make sure the UART terminal window is actively selected) to start the motor at a 10% duty cycle in the forward direction (forward is considered to be counter-clockwise when looking directly at the rotor shaft). If the motor direction is reversed from expected, then consider swapping two of the OUT motor phases. Otherwise, press *s* once more to stop the motor (or reset the LaunchPad), then iterate changes to the HALL inputs on the DRV8329AEVM until the UART output follows the exact same sequence as shown. This is to make sure that the BLDC motor runs as expected when the hall sensors are enabled.

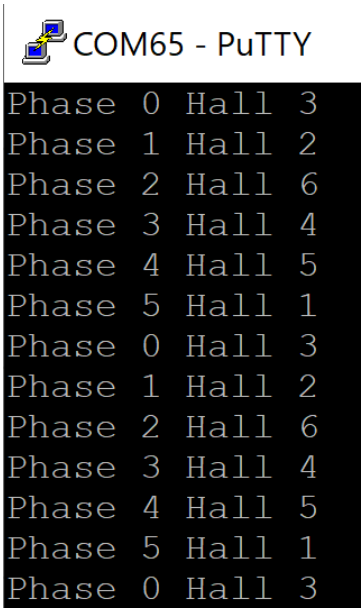

Figure 3-5. BLDC Motor No Hall Test

3.4 BLDC Motor Hall Sensored Trap Operation With Bluetooth® LE

After all of the correct motor connections have been confirmed, users can run the default example. Make sure that the TEST_NO_HALL predefinition has been removed before rebuilding and loading the project.

3.4.1 SimpleLink™ Connect Phone Application

This demonstration uses the out-of-box SimpleLink Connect phone app (on either SimpleLink™ Apple Store or SimpleLink™ Play Store) to interact with the BLDC motor firmware. Once the CC2340R5 device is running, users can use the *Scanner* feature to locate and connect to the *Bluetooth LE/BLDC MOTOR* device. Then, select the *TI Simple Peripheral Service* to open access to the pertinent characteristics.

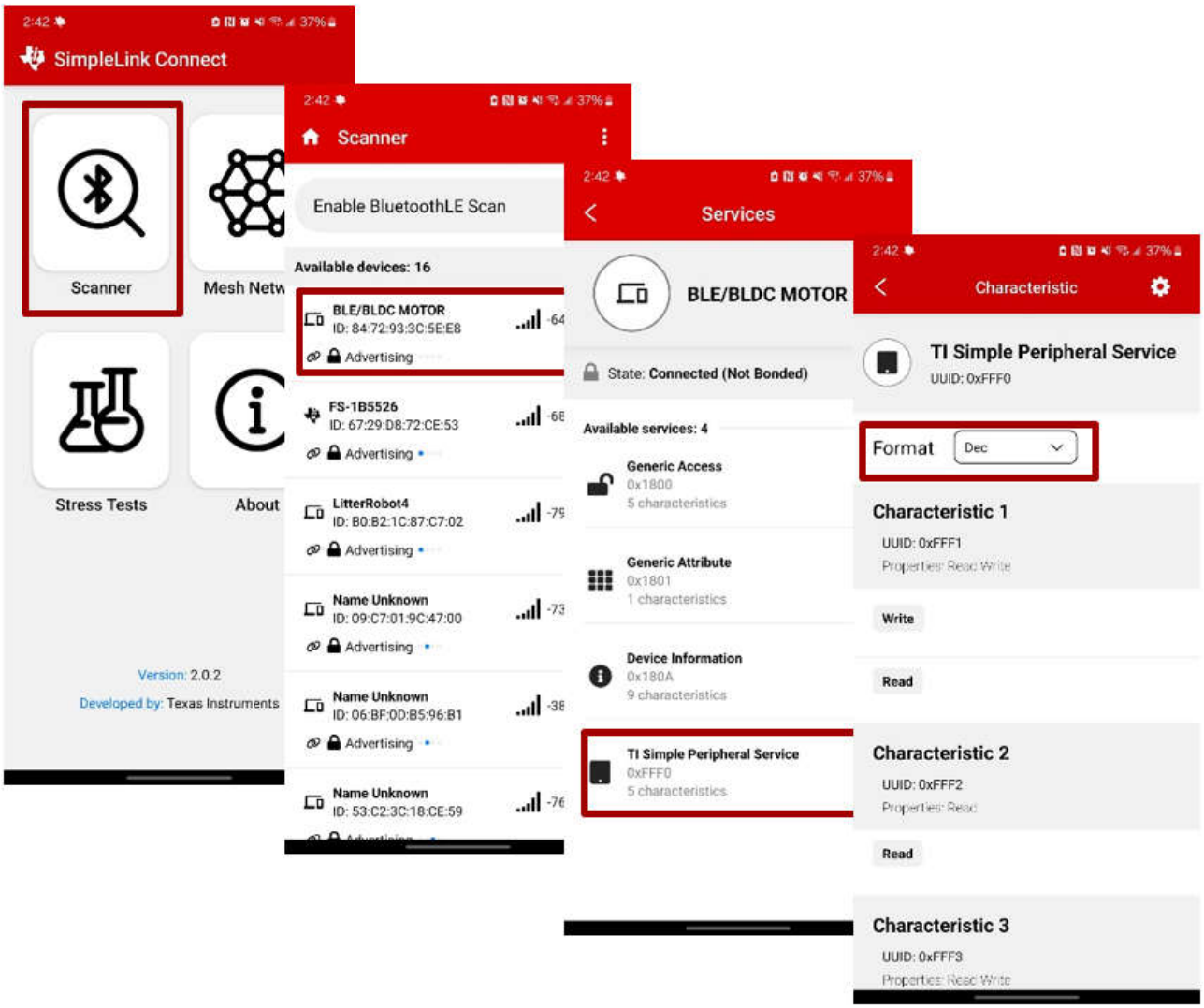

Figure 3-6. SimpleLink Connect Application

A custom Bluetooth LE profile was made for this project. The profile was based off of the *Simple GATT* profile found in the basic Bluetooth LE example (*simple_gatt.c*) in the SDK, with significant modifications to the properties of each characteristic, to achieve the desired functionality of this project. In the *SimpleLink™ Connectivity* phone application, this shows up as *TI Simple Peripheral Service*. Table 3-1 reflects all of the available Bluetooth LE options to interact with the motor, including the Bluetooth LE characteristics and the UART interface described in the prior motor connection test section.

Table 3-1. SimpleLink™ Connect App and UART Interface Actions

Action	SimpleLink Connect Interface	UART Keyboard Entry
Start the motor	Write *1* to characteristic 1	*s* (if motor is stopped)
Stop the motor	Write *0* to characteristic 1	*s* (if motor is running)
View RPM	Read characteristic 2 (Hex format)	*l* (the letter)
Forward direction	Write *1* to characteristic 3	*f*
Reverse direction	Write *0* to characteristic 4	*r*
Duty cycle percentage	Write characteristic 5 (Dec format)	N/A
Increase duty cycle	N/A (see characteristic 5)	*u*
Decrease duty cycle	N/A (see characteristic 5)	*d*

4 Firmware Design

Now that the default BLDC motor project is running as intended, the firmware is further analyzed.

4.1 Code Flow Description

Figure 4-1 shows a simple code block diagram of the processes used inside the CC2340R5 code. The functionality is realized in the *app_bldc_motor_control.c* file.

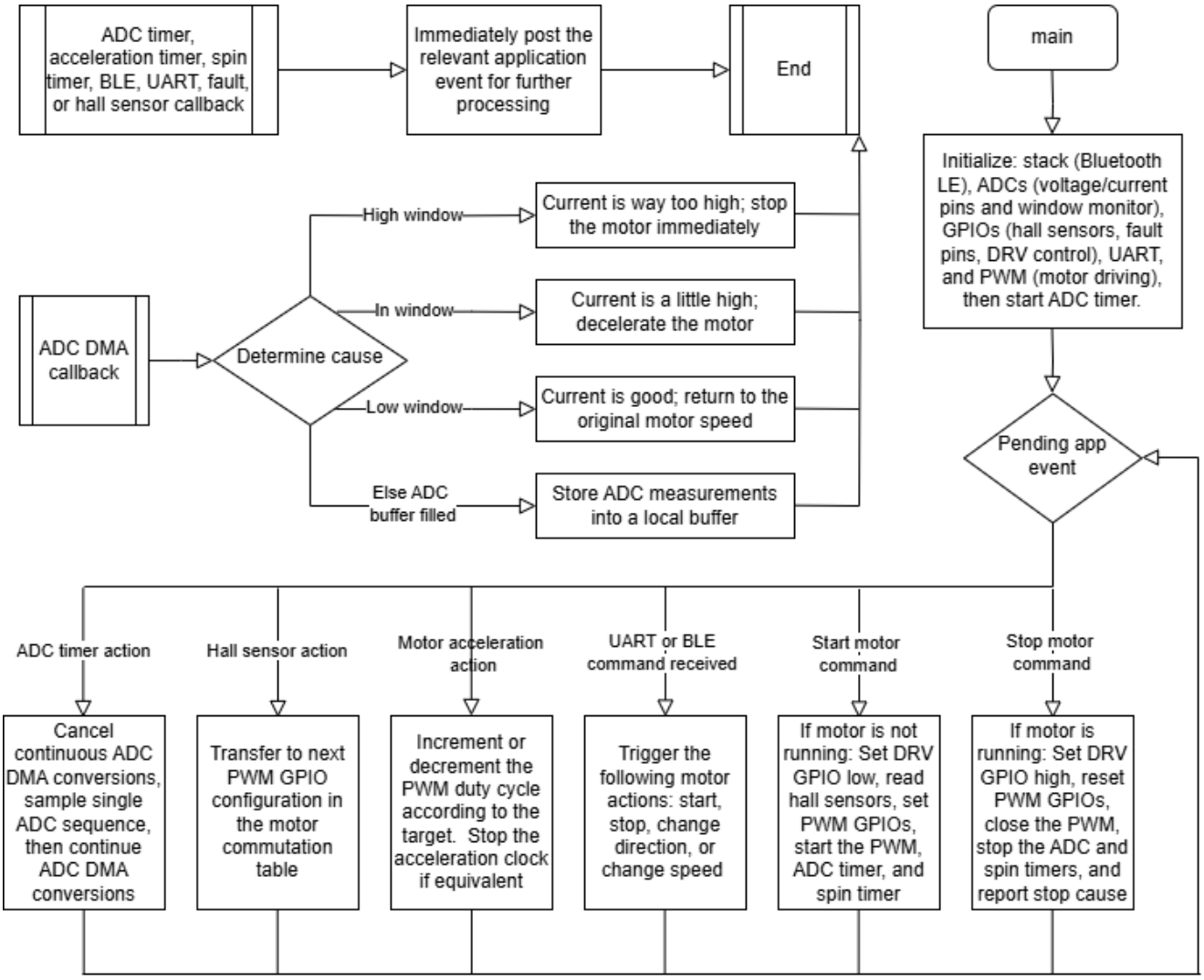

Figure 4-1. BLDC Motor Code Diagram

The main function initializes all TI drivers and timers necessary for the BLDC motor example to operate. Once entering the main while loop, this pends further action on an event being set by hardware callbacks. After servicing the corresponding action through a subroutine, the events are reset and the process repeats.

All hardware callbacks, with a few exceptions, simply post an event for the main application to process. An exception is the ADCBuf callback which processes the status immediately and does not call for any further action from the main application loop unless a window comparator threshold has been exceeded.

There are several definitions implemented which define the motor behavior and are referenced in the following sections. Table 4-1 shows the configurable definitions located in app_bldc_motor_control.h.

Table 4-1. BLDC Motor Application Definitions

Definition	Default	Units	Function
PWM_PERIOD	40000	Hz	PWM period divided by two (for up or down mode), thus 20kHz
DUTY_MAX	80	%	Maximum PWM duty cycle allowed
DUTY_MIN	10	%	Maximum PWM duty cycle allowed
DUTY_INC	5	%	PWM duty cycle increment from UART command
MAX_STRING	20	Integer	Maximum characters in a single UART string
SPIN_TIMEOUT	1000000	µs	The motor driver stops if the motor has not rotated once during this span of time
RPM_INTERVAL	500000	µs	Time interval for which the RPM, CPU load, and ADC measurement count are reported
ACCEL_INTERVAL	10000	µs	Time delay before increasing or decreasing the PWM duty cycle
CHANGE_DELAY	1000000	µs	Delay after stopping the motor before changing direction
ADC_COUNT	5	Integer	Number of ADC channels enabled
ADC_INTERVAL	100000	µs	Time interval to sample single ADC sequences (requires stopping and then restarting the ADC DMA buffer)
VSEN_THRESHOLD	2000	Integer	Raw power supply ADC value required to trigger motor stop
WINDOW_LOW	1500	Integer	Raw current ADC value of the low window monitor
WINDOW_HIGH	3000	Integer	Raw current ADC value of the high window monitor
ADCSAMPLESIZE	128	Integer	Size of ADC buffer transferred by DMA

4.2 Customized TI Drivers

There are a few TI Drivers which have been customized to function in the way intended for BLDC motor operation. Modified driver application files have been included in the project directory so that the files are used in place of the originals packaged with the SDK release.

4.2.1 PWM

The PWM TI Driver is originally designed to drive just one channel output from a timer in up mode. For the purposes of the BLDC motor, *PWMTimerLPF3.c* has been modified to drive all three channel outputs (representing the three motor phases) from a single timer. The driver also utilizes up or down mode as is important for triggering motor current measurements halfway through a duty cycle. The PWM_PERIOD (20kHz after dividing by two for up or down mode) is a typical PWM frequency when driving BLDC motors and TI does not recommend to change this value unless acutely familiar with the motor design.

4.2.2 ADCBuf

The default ADCBuf operates on a single channel in repeated single mode. Since two continuous channel conversions are warranted for this application, current (ISEN) and voltage (VSENPVDD), a repeated sequence mode is preferable. *ADCBufLPF3.c* is, therefore, also configured to transfer data through the ADC peripheral's FIFO instead of a single memory register.

The original TI Driver also chooses to begin subsequent ADC conversions automatically. This is not a good choice for the purposes of the motor design which must measure current halfway through a PWM duty cycle. Therefore, the ADC is set to trigger conversions of the first ADC channel (ISEN) upon the LGPT of the PWM TI Driver reaching the target value in up or down mode. The next ADC channel (VSENPVDD) is measured immediately afterward, and the process continues until the buffer is filled.

The ADCBuf callback operation also includes status processing for the high, in, low interrupts of the window monitor which is initialized in the *app_bldc_motor_control.c* file. This way, the application is notified of either a window monitor change or completed ADC buffer through DMA.

4.2.3 Power

A custom policy function is added to *app_bldc_motor_control.c* to count the amount of time spent idling when the CPU is not performing any actions. This is then reported to the user as a measurement of the application total CPU usage. The *bldc_motor.syscfg* SysConfig file power module directly refers to this custom policy function so that the function is used. Consider this feature for test and evaluation only.

4.3 Application Events

When a user operation or hardware callback sets an event, the following routines are processed.

Table 4-2. Application Routines and Function

Routine	Function
ACTION_UART	Process either Bluetooth LE or UART functionality received through user input which was described earlier. Developers can modify this section to determine functionality and user interface.
ACTION_CONTINUE	Start the motor by driving to the DRV pin low, setting the minimum duty cycle, initializing the correct motor GPIO states, starting application timers, and opening the PWM TI Driver. This routine is not performed if the PWM TI Driver is already opened.
ACTION_STOP	Stop the motor by driving the DRV pin high, resetting the motor GPIO pins, stopping relevant application clocks, and stopping the PWM TI Driver. This also reports out the cause of the stop through UART.
ACTION_HALL	After a hall sensor pin callback, reads all hall sensor states and proceeds along the motor commutation table accordingly.
ACTION_PWM	Serviced whenever the acceleration clock expires to increment or decrement the duty cycle value until this matches the goal, and then stop the acceleration clock once achieved.

4.4 Commutation Table

Each time a hall sensor GPIO transitions from high to low or low to high, the motor commutation tables are used to select the next motor phase state. Thus the *reversePhaseTable* or *forwardPhaseTable* arrays take the total bitmask value of all three hall sensors and determine the next case to be serviced by *forwardDirection* or *reverseDirection* functions. These functions simply set each motor phase to the correct and expected configuration accordingly using direct hardware register commands.

4.5 Motor Acceleration

As the motor cannot achieve a significant instantaneous duty cycle change, for example from 20% to 80%, the acceleration clock is used to increment or decrement the duty cycle by 1% every ACCEL_INTERVAL µs. This clock is stopped once the duty cycle output matches the intended goal. Precautions are implemented by the application to make sure that the duty cycle remains within DUTY_MIN and DUTY_MAX. Decreasing ACCEL_INTERVAL makes the motor transition duty cycles quicker but can increase current spikes by doing so.

4.6 ADC Operations

Measuring all pertinent ADC channels with the one CC2340R5 ADC peripheral is a challenge considering that there are more ADC inputs (ADC_COUNT is five) than memory registers (four) and the need to measure the ISEN pin (motor current) at each PWM duty cycle output (thus at a rate of PWM_PERIOD/2). The implementation devised is to have the modified ADCBuf TI Driver, which utilizes ADC and DMA to quickly and efficiently fill an ADC buffer of size ADCSAMPLESIZE without interrupting the CPU, measure both ISEN and VSENPVDD (system voltage) continuously while occasionally interrupting this process to measure the remaining VSENA, VSENB, and VSENC ADC pins at a rate of ADC_INTERVAL. As a result, the number of ADC measurements per channel using ADCBuf is slightly lower than the PWM_PERIOD/2 due to the time and frequency of the single ADC measurements triggered every time the ADC_INTERVAL timer expires. As the ADCBuf driver has been configured to trigger the ADC from PWM LGPT activity, the driver does not take measurements while the motor is stopped. By contrast, the single ADC measurements are valid each time ADC_INTERVAL occurs regardless of motor activity.

A window monitor is also enabled on the ADC peripheral so that immediate action can be taken if the value of the latest ISEN measurement goes within or above the window limits to be further processed in adcBufCallback. If the value of WINDOW_LOW is surpassed, then an ADC callback status of ADC_INT_INIFG reduces the PWM duty cycle until the cycle goes lower than the window value. At which point, the ADC_INT_LOWIFG status

returns the duty cycle to the original value. If WINDOW_HIGH is surpassed, then the ADC_INT_HIGHIFG status immediately stops the motor. Any ADC memory register can trigger the window monitor, but only one value for the low and high window thresholds can be set. Only ISEN is configured to use memory register two for which the window monitor is enabled, thus, only ISEN can trigger the window monitor. As ISEN switches between two ADC memory registers (zero and two), the window monitor evaluates ISEN at a rate of approximately 10kHz.

Whenever the ADC callback is triggered and not caused by the window monitor status, the ADC buffer has been filled. The application populates the outputBuffer array with the results of the ADC conversions and evaluates whether the motor needs to be stopped due to a received VSENPVDD ADC conversion surpassing VSEN_THRESHOLD. All conversions are 12-bit resolution (maximum value of 4096) and left in the raw format. Although additional processing APIs are available in the TI Drivers to further adjust these values and convert them to a µV format. Caution must be taken so that these processes do not interfere with the motor drive functions.

4.7 Spin Detect Feature

An application timer is begun every time the motor starts and expires if the motor does not complete one entire revolution every SPIN_TIMEOUT µs. This is to prevent having an open PWM TI Driver actively outputting to the motor phases if the motor is not actively spinning. The timer restarts every full iteration of the commutation table, and the timer is stopped when the motor is commanded to stop by any other application method.

4.8 Reporting Statistics

Each RPM_INTERVAL, µs, statistics are updated based on the counts of the respective subprocesses. The statistics currently enabled include the CPU load as a percentage (cpuLoad) and the RPM in units of revolutions per minute (rpmValue). Another option is the total ADCBuf measurements taken per second (adcTotal). These are reported to the user as determined by the UART or Bluetooth LE interfaces, but can also be viewed inside the Expressions Window of an active debug session. When using a motor capable reaching higher than around 25k RPM, take caution about how statistics are reported as can interrupt the motor drive commutations as this can create a fault (indicated by the DRV8329A shutting down motor motion and driving low the nFAULT pin input to the CC2340R5). Thus, TI recommends that the data being reported is as short and concise through Bluetooth LE for the application.

4.9 Bluetooth® LE Stack

This example is based on the out-of-box Basic Bluetooth® LE Project and Data Stream Bluetooth LE examples found in the SimpleLink Low Power F3 SDK v8.40. However, the Bluetooth LE profile was heavily modified to accomplish the goals of the project described in this application note. All project functionality not pertaining to the Bluetooth LE stack was handled outside the Bluetooth LE task context, as to not starve the task or risk impacting the Bluetooth LE connection. The device functions as a Bluetooth LE peripheral and advertises using the following parameters, however these are highly configurable. For more information on scanning and advertising in Bluetooth LE, refer to the SimpleLink Academy Scanning and Advertising lab.

Table 4-3. Bluetooth® LE Stack Configurations

Bluetooth® LE Component	Value
Advertisement Type	Connect Bluetooth LE and scan Bluetooth LE Undirected Advertisements
Primary PHY Interval Min and Max	100ms, 100ms
Primary PHY	1M PHY

Upon receiving a connection established event in the app_connection.c file, the CC2340R5 sends an exchange MTU request to the Bluetooth LE central to increase the maximum size of data that can be sent when interacting with the Bluetooth LE characteristics of the Bluetooth LE profile. To perform this, the max size of PDUs was increased to 255 in the general configurations of the Bluetooth LE stack in the Syscfg file. This was heavily utilized in the throughput test that is discussed in Section 5.

5 Tests and Results

CPU load and RPM measurements with an active Bluetooth LE connection have been recorded for various duty cycle outputs and are provided in Table 5-1. Keep in mind that RPMs relative to the duty cycle will be greatly dependent on the physical properties and characteristics of the BLDC motor used.

Table 5-1. BLDC Motor Application Performance

Duty Cycle (%)	RPM	CPU Load (%)
10	1080	3
30	5040	5
50	10800	8
70	15120	11
90	23520	15

Using a logic analyzer is easier to visualize the BLDC motor being driven using the high and low sides of the three phases, along with the transition between hall effect sensor level transitions

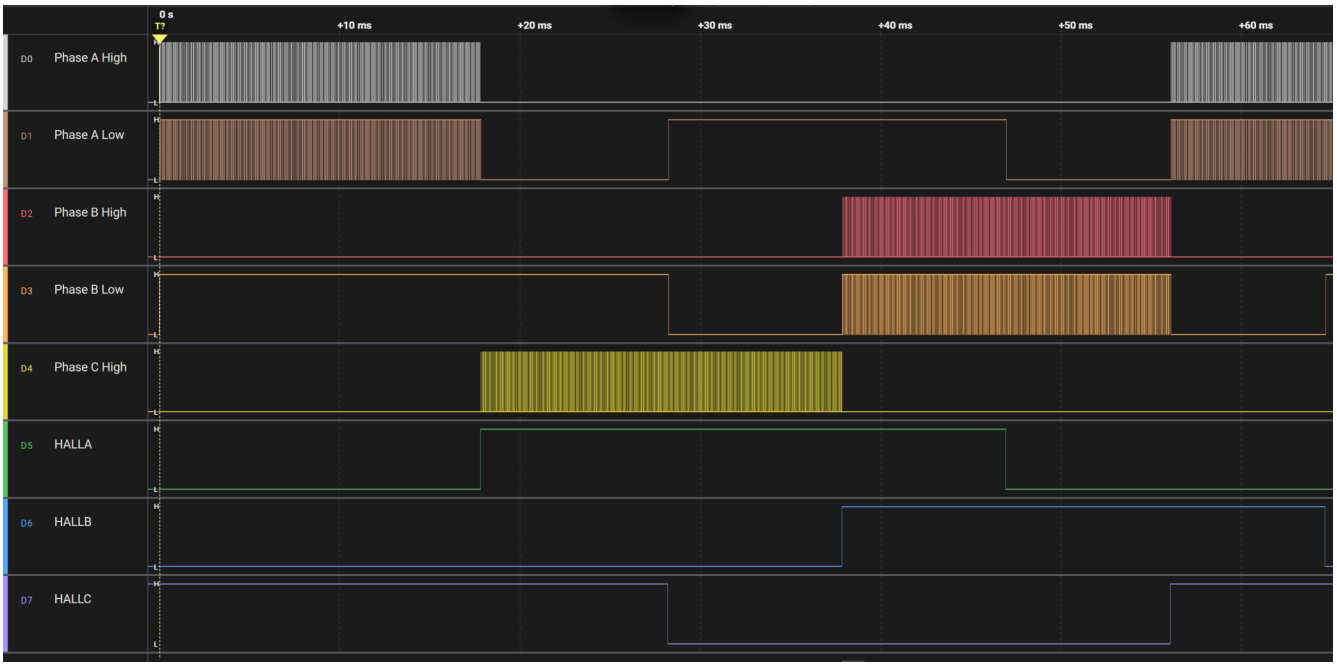

Figure 5-1. BLDC Motor Oscilloscope Screenshots

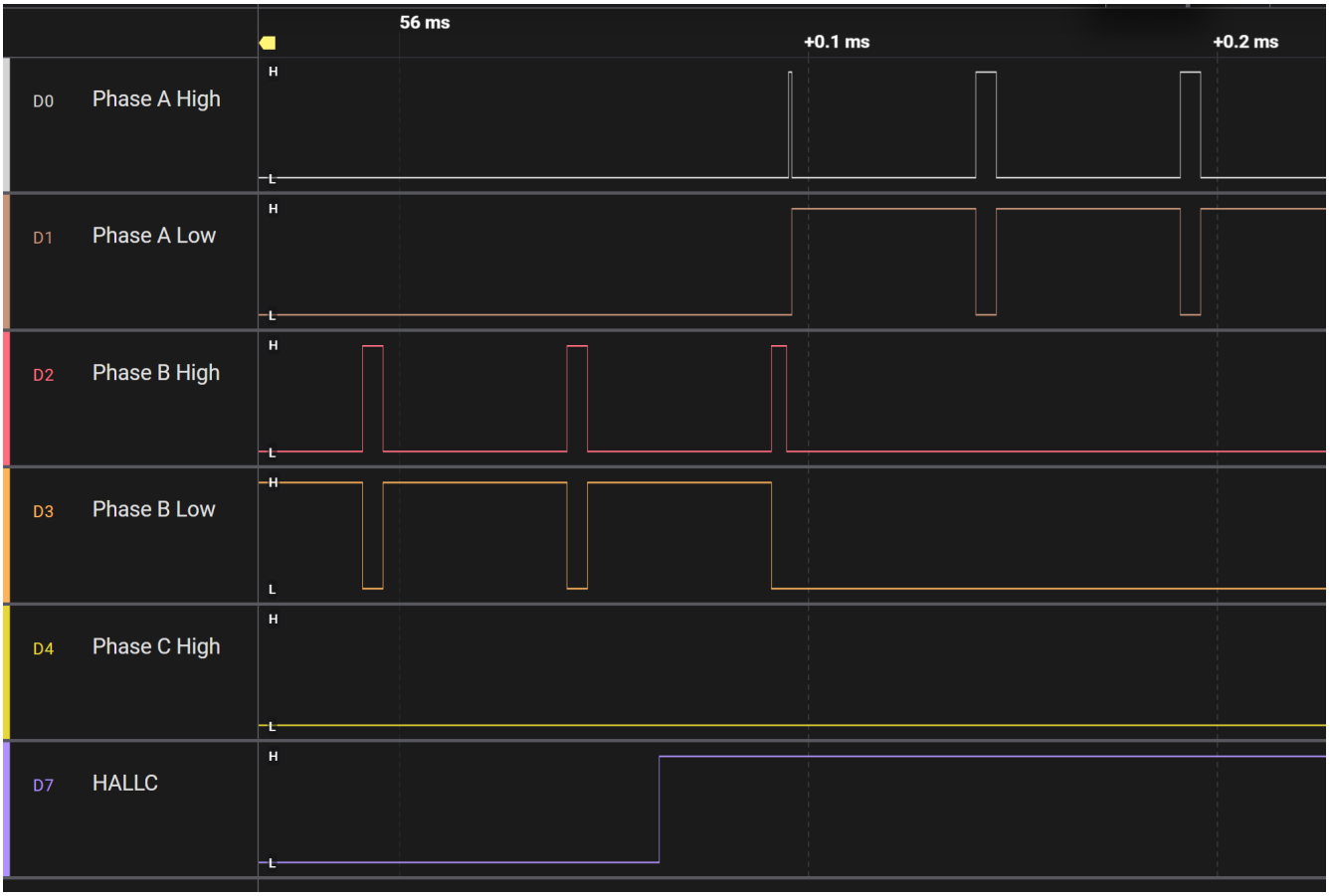

Figure 5-2. BLDC Motor Transition

When disabling the UART peripheral, not enabling Bluetooth LE behavior, and not actively spinning the BLDC motor, the CC2340R5 LaunchPad operates with a power consumption of less than 1µA.

As determined through the Memory Allocation view inside of CCS, the default project requires 184 kB of flash (not including the 16 kB reserved for non-volatile memory) and 32 kB of SRAM. Disabling the UART save 3 kB of flash, and a project built without using the Bluetooth LE stack only consumes 22 kB of flash and 15 kB of SRAM in total.

A throughput test was added to get a rough estimate of the maximum amount of data that can be sent out over characteristic 5 of the Bluetooth LE profile. To perform this test, a clock instance was initialized to send out 247 bytes of data over notifications. The device was connected to a central, whose connection interval was 45 ms. Across this connection, and while maintaining a duty cycle of 80%, 380 Kbps throughput was achieved. Throughput tests depend on a number of factors, such as connection intervals, and CPU overhead (motor duty cycle in this case), therefore these numbers are just a rough estimate and could be improved with the right modifications. The stress test is disabled by default but can be easily added by setting the start flag of the ClockP instance found in *app_peripheral.c* to True.

6 Summary

This application note has fully defined a BLDC motor hall effect sensor trap implementation using the SimpleLink CC2340R5 and DRV8329A. A description of the necessary hardware connections and MCU programming instructions have been provided so that users are empowered to operate the out-of-box demonstration. Test results have been provided to confirm the stability and robustness of the implementation. The source code is freely accessible and the code flow has been detailed so that developers are familiar with how the project works and are enabled to further modify the project to fit unique application requirements. Readers are encouraged to post to the E2E forum concerning any additional questions Texas Instruments, or support needs that pertain to these resources provided.

7 References

1. Texas Instruments, CC2340R SimpleLink™ Family of 2.4GHz Wireless MCUs, data sheet
2. Texas Instruments, Meet the CC2340R5 LaunchPad Development Kit, quick start guide
3. Texas Instruments, DRV8329 4.5 to 60V Three-phase BLDC Gate Driver, data sheet
4. Texas Instruments, DRV8329AEVM User's Guide
5. Texas Instruments, BLE5-Stack User's Guide
6. Texas Instruments, SimpleLink Academy for CC23xx
7. Texas Instruments, Measuring CC13xx and CC26xx Current Consumption, application note
8. GitHub, SimpleLink Low Power F3 Demos, example code

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATA SHEETS), 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, regulatory 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 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.

TI objects to and rejects any additional or different terms you may have proposed.

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