Application Note

Stepper Motor Control Using the SimpleLink CC2340R5 MCU With Proprietary RF

Ryan Brown

ABSTRACT

This application report explores the capability of adding Proprietary RF protocol to stepper motor designs in a one-MCU solution. Stepper motors are commonly found in applications which may benefit from having embedded radios, including radiator valves, 3D printers, and robotics. The material covered in this document will prove how the SimpleLink CC2340R5 MCU is capable of fulfilling this task with assistance of the DRV8411 motor driver.

The solution provided by this document uses hardware EVMs which are available for purchase on TI.com and firmware freely provided on the SimpleLink Low Power F3 Demos GitHub. A description of both necessary hardware connections and the firmware's operation are given in detail so that, after obtaining a stepper motor, developers can be fully enabled to operate their own demonstration and further modify the project to meet their requirements. Additional test data is also provided so that readers may fully understand the operating conditions alongside options for application expansion.

Table of Contents

1 Introduction....2
1.1 CC2340R5....2
1.2 Stepper motor....2
2 Stepper Motor Hardware....4
2.1 Hardware setup....4
2.2 DRV8411EVM settings....4
2.3 Connection Diagram....4
3 Running the Example....6
3.1 Dependencies....6
3.2 Loading Firmware....6
3.3 Local Stepper Motor Control....6
3.4 Remote Control Using Proprietary RF....7
4 Firmware Design....8
4.1 Code Flow Description....8
4.2 ADCBuf....8
4.3 Power....9
4.4 Application Events....9
4.5 Step Table....9
4.6 Fault Detection Pin....9
5 Tests and Results....10
6 Summary....13
7 References....14

Trademarks

1 Introduction

The SimpleLink™ 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. Given this feature set it is capable of achieving a multitude of end applications for a variety of radio protocols in a single-chip design. This application report highlights a single instance to prove the wider possibilities capable with this device.

Coupled with a DRV8411 motor driver, this becomes possible to control a stepper motor using different stepping modes with the CC2340R5 through Proprietary RF radio communication, among others. This guide closely details the hardware and software implementations necessary to realize this application, and comment towards optional features which have been enabled. Through reading this report users can become knowledgeable in both stepper motor control and CC2340R5 development, and gain confidence to utilize similar concepts for their own designs.

1.1 CC2340R5

The CC2340R family is part of the SimpleLink™ MCU platform, which consists of Wi-Fi®, Bluetooth Low Energy, 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 Low Energy MCU with support for Bluetooth 5 Low Energy, and 2.4-GHz proprietary protocols. Software support is provided by the SIMPLELINK-LOWPOWER-F3-SDK which can be built using the freely offered / CCSTUDIO IDE. Features include access to all I/O signals with the BoosterPack™ plug-in module connectors and connecting the LaunchPad development kit to your smart phone using TI SimpleLink Connect. The LP-XDS110ET or LP-XDS110 debugger (sold separately) is required for programming, debugging, and RF evaluation.

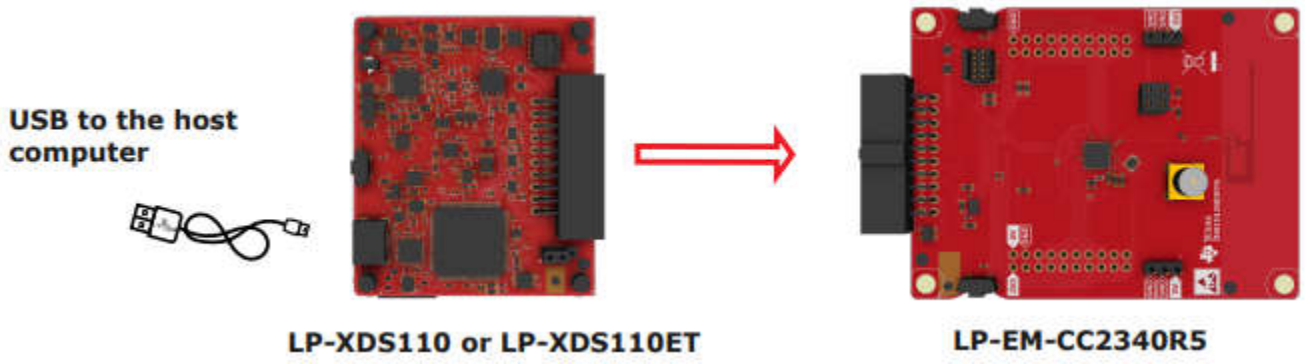

Figure 1-1. LP-XDS110ET and LP-EM-CC2340R5 Connection

1.2 Stepper motor

A stepper motor is a brushless DC motor which uses precise steps to allow for accurate positioning. It accomplishes this by aligning the motor's rotor with the stator's poles through a sequence of energizing electromagnetic coils in a prepared sequence (i.e. step). Each unique stepper motor has a step angle determined through its characteristics, alongside the recommended voltage and phase amperage which should be used to achieve the expected torque.

Figure 1-2. Stepper Motor

2 Stepper Motor Hardware

2.1 Hardware setup

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

2.2 DRV8411EVM settings

Figure 2-1 is a depiction of the DRV8411EVM board with jumpers populated in the correct places. Please refer to the DRV8411EVM User's Guide for more instructions on how to interface with this hardware.

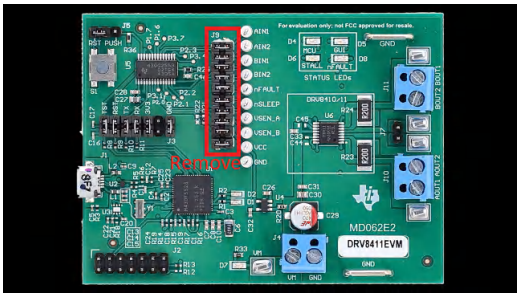

Figure 2-1. DRV8411EVM Hardware Setup

2.3 Connection Diagram

Table 2-1 is the connection between the DRV8411EVM and CC2340R52 to realize the stepper motor demonstration.

Table 2-1. Table Connections Between the CC2340R5 and DRV8411EVM

Connection	CC2340R5 function	CC2340R5 pin	DRV8411EVM
Phase A1	digital output	DIO23	AIN1
Phase A2	digital output	DIO2	AIN2
Phase B1	digital output	DIO5	BIN1
Phase B2	digital output	DIO25	BIN2
Fault indicator	digital interrupt input	DIO18	nFAULT
Sleep mode	digital output	DIO1	nSLEEP
Phase A voltage	ADC input	DIO24	VSEN_A
Phase B voltage	ADC input	DIO0	VSEN_B
Common GND	N/A	GND	GND

The stepper motor wires will need to be connected to the DRV8411EVM outputs in the orientation specified by the motor manufacturer. The DRV8411EVM VM/VCC power should supplied separately of the LP-EM-CC2340R5, as the LP-XDS110(ET) cannot sufficiently supply the current necessary to supply the stepper motor. The end result looks similar to Figure 2-2.

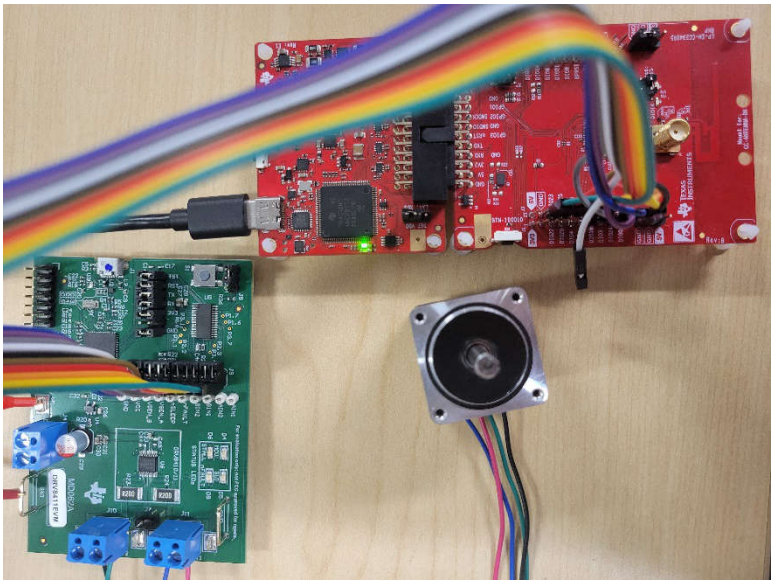

Figure 2-2. Physical Hardware Setup

3 Running the Example

The next sections includes the firmware details and how each component works towards driving the stepper motor and controlling with a remote radio-enabled device.

3.1 Dependencies

The code projects supplied on the SimpleLink Low Power F3 Demos GitHub uses SimpleLink F3 SDK v9.11.0.18, SYSCONFIG Configuration Tool v1.22, and the TI CLANG v4.0.3 compiler. Please make sure that all of these dependencies are installed on your machine before attempting to import the project into Code Composer Studio (CCS) v20 or later. For more examples for setting up your environment, please refer to these SimpleLink Academy Labs: Resource Explorer. Please note that users are responsible for migrating and supporting any dependency versions not listed above.

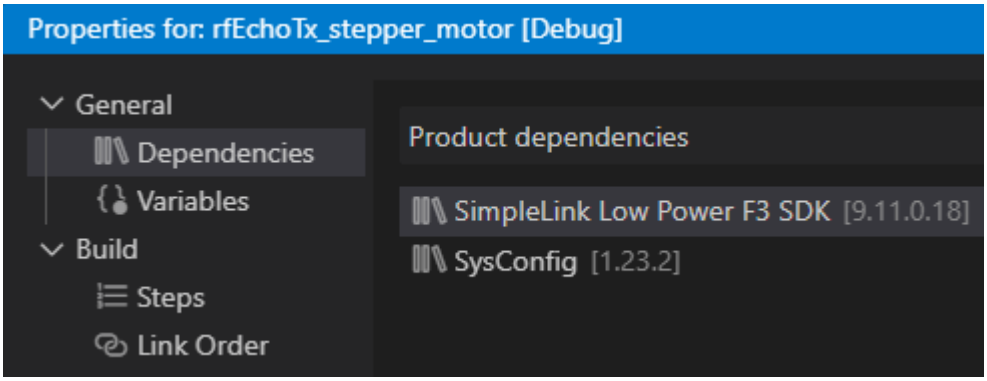

Figure 3-1. CCS Properties

Two projects are included in the following

- rfEchoTx_stepper_motor: This is for the CC2340R5 device controlling the stepper motor and is the central focus of this document
- rfEchoRx: This is a companion CC2340R5 project for communicating stepper motor commands to the rfEchoTx_stepper_motor over Proprietary RF

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). The recommendation is to exit Debug Mode to allow free-running if the project is not actively being debugged. Also consider using the Uniflash software tool to load binary images.

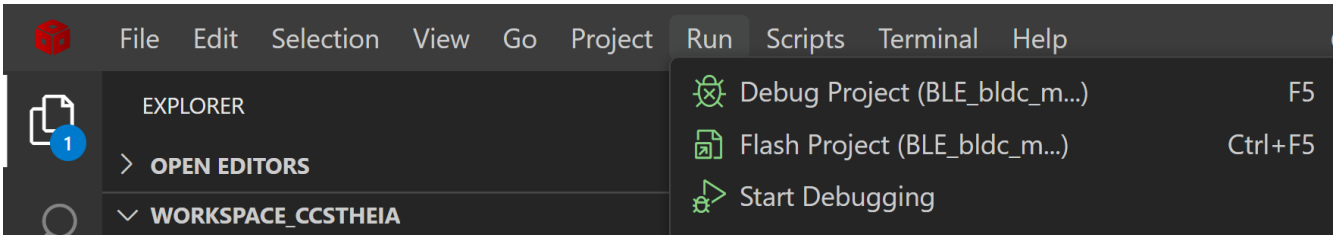

Figure 3-2. CCS Load Options

3.3 Local Stepper Motor Control

To test the stepper motor operation locally using just rfEchoTx_stepper_motor, the LaunchPad push buttons are configured to move the stepper motor counter-clockwise when pushing BTN-1 and clockwise when pushing BTN-2. Make sure that both the LP-EM-CC2340R5 and DRV8411EVM boards are powered, and that the CC2340R5 firmware has been programmed. The speed and number of steps per action are dependent on the following definitions within *stepper_motor.c*

Table 3-1. Stepper Motor Application Definitions

Definition	Default	Units	Function
FULL_STEP	Not defined	N/A	Firmware produces full step waveforms
HALF_STEP_SLOW	Not defined	N/A	Firmware produces half step waveforms for slow decay
HALF_STEP_FAST	Defined	N/A	Firmware produces half step waveforms for fast decay
STEP_FREQUENCY	400	Hz	Number of steps per second
NUMBER_STEPS	400	Integer	Number of steps per action
ADC_SAMPLE_SIZE	100	Integer	Size of the ADC buffer
ADC_PER_STEP	50	µs	Number of ADC measurements taken per step
WINDOW_HIGH	1500	Integer	12-bit threshold of the ADC window comparator

In this example, using a stepper motor with a 1.8° step angle (200 steps/revolution), the motor can complete two rotations (400 steps) during a second of operation (400 steps per second).

3.4 Remote Control Using Proprietary RF

After the correct motor functionality has been confirmed, this is possible to add a remote LP-EM-CC2340R5 device running a Proprietary RF project. Included in this example is a slightly modified version of rfEchoRx, where similar button functionality has been implemented to be communicated over the Proprietary RF radio every time rfEchoTx_stepper_motor requests a packet. To communicate with the rfEchoTx portion of the stepper motor code, the *.syscfg files must have the exact same SysConfig -> RF Stacks -> Custom configuration.

Table 3-2. Proprietary RF Application Definitions

Definition	Default	Units	Project	Function
MAX_LENGTH	10	Integer	Both	Data bytes per packet
PACKET_INTERVAL	20000000	¼ ms	rfEchoTx	Interval between packet events
TX_DELAY	40000	¼ ms	rfEchoRx	Delay before transmitting response
RX_TIMEOUT	80000	¼ ms	rfEchoTx	Timeout waiting for response
FREQUENCY	2412000000	Hz	Both	Proprietary RF frequency

No modifications to the rfEchoRx CC2340R5 LaunchPad are necessary. If done properly, with both LaunchPads programmed and powered on, you are able to push BTN-1 or BTN-2 on the rfEchoRx LaunchPad and, after the PACKET_INTERVAL expires, observe the stepper motor moving counter-clockwise or clockwise, respectively, after the packet is received by the rfEchoTx_stepper_motor LaunchPad.

4 Firmware Design

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

4.1 Code Flow Description

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

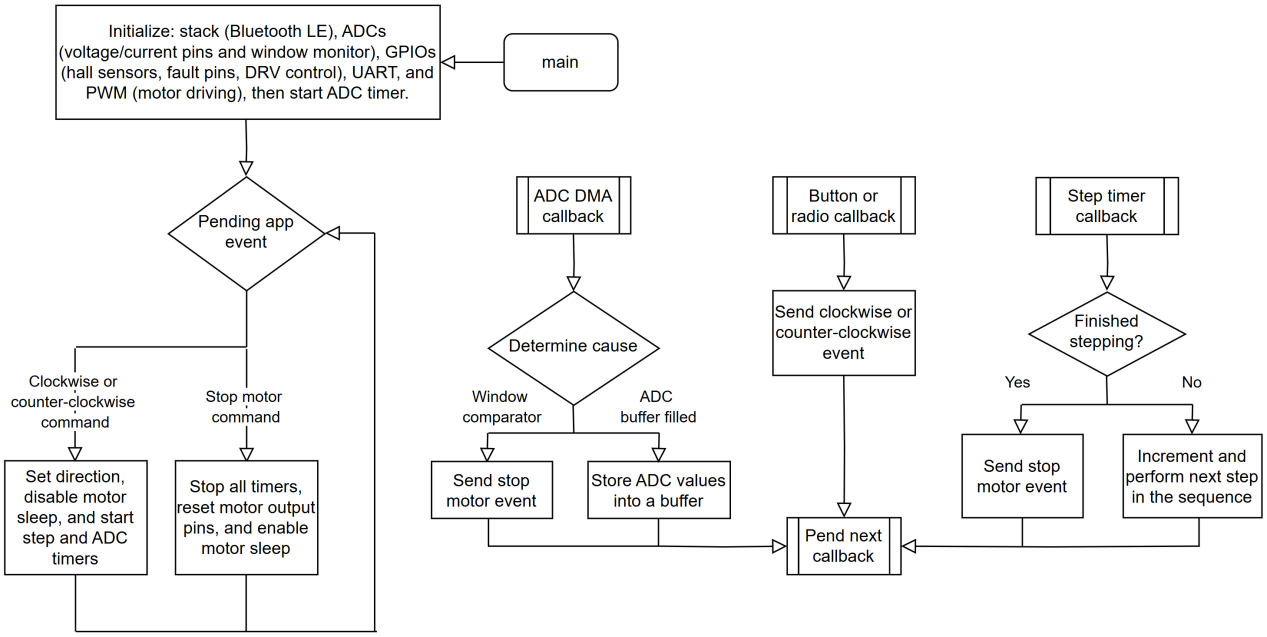

Figure 4-1. Stepper Motor Code Diagram

The main function initializes all TI drivers and timers necessary for the stepper motor example to operate. Once entering the main while loop, this can pend 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 itself.

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.

4.2 ADCBuf

The default ADCBuf operates on a single channel in repeated single mode. Since two continuous channel conversions are warranted for this application, both VSEN_A and VSEN_B, a repeated sequence mode is used by adding a custom *ADCBufLPF3.c* to the project's main directory. This transfers 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 suitable for the purposes of the stepper motor design which should take measurements a defined number of times per step. Therefore, the ADC is set to trigger conversions based on a LGPT trigger. The rate of ADC measurements per step is determined by the ADC_PER_STEP definition. As two measurements are taken each interval, *ADC_SAMPLE_SIZE* divided by two represents the frequency of ADC callbacks. This also is dependent on the amount of RAM available for increasing the ADC buffer.

The ADCBuf callback operation also includes status processing for the high interrupts of the window monitor which is initialized in the *stepper_motor.c* file. This way the application is notified of a high current event and can stop the motor immediately if triggered. The values have been adjusted and converted to microVolts by default using APIs included in the TI Driver.

4.3 Power

A custom policy function is added to *stepper_motor.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 *stepper_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.

4.4 Application Events

Once a user operation or hardware callback sets an event, this is processed by the following routines.

Table 4-1. Application Routines and Functions

Routine	Function
ACTION_CCLOCK	If a motor movement is not ongoing, then set the direction and action variables accordingly and set ACTION_START
ACTION_CLOCK	If a motor movement is not ongoing, then set the direction and action variables accordingly and set ACTION_START
ACTION_START	Start the motor by driving to the sleep pin high, starting the motor step clock, beginning ADC conversions, and initializing the LGPT sync timer for ADC measurements
ACTION_STOP	Stop the motor by closing all peripherals commenced in ACTION_START and driving the sleep pin low

4.5 Step Table

Each time the motor timer expires, the step count is incremented and evaluated. If there are no more steps then the ACTION_STOP application event is called. Otherwise, a case statement is used to determine the next iteration to complete the current step for the motor output pins. The number of iterations is dependent on the stepping mode, whether that is FULL_STEP (four iterations) or HALF_STEP_SLOW/ HALF_STEP_FAST (eight iterations). Only one stepping mode can be selected at any time.

4.6 Fault Detection Pin

If the DRV8411 detects an error with the stepper motor performance then this forces the nFault pin low. The CC2340R5 has configured the corresponding pin as an interrupt so that the ACTION_STOP event can be immediately called to discontinue the motor output pin waveforms.

5 Tests and Results

A logic analyzer is used to validate the waveform being driven from the CC2340R5 to the DRV8411. An example of this is provided below for half-stepping with fast decay mode.

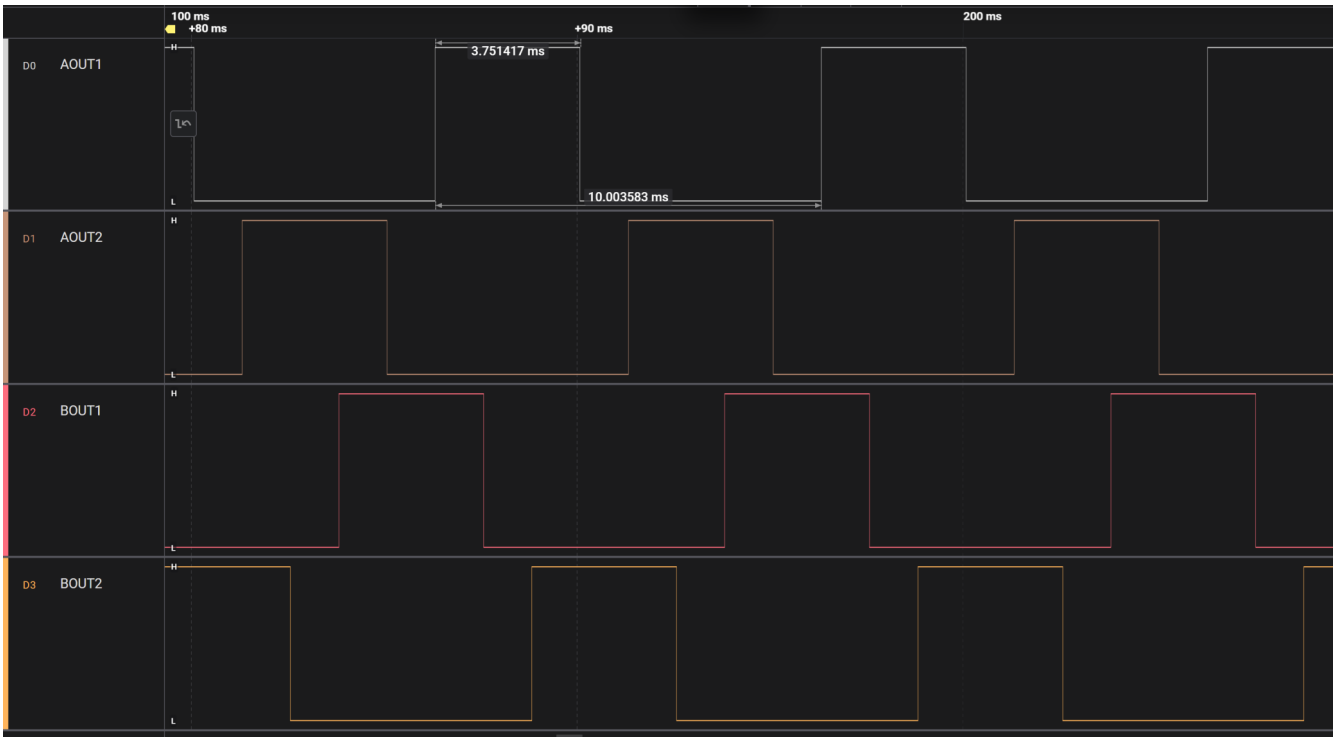

Figure 5-1. Stepper Motor Waveforms in Half-Stepping With Fast Decay Mode

ADC values can be monitored using the Code Composer Studio's Watch window during a debug session. In the image below, even entries display VSEN_A whereas odd entries reflect VSEN_B. During this brief period in time it is possible to observe that both motor phases are being actively driven as the values of both are rising with the progression of time.

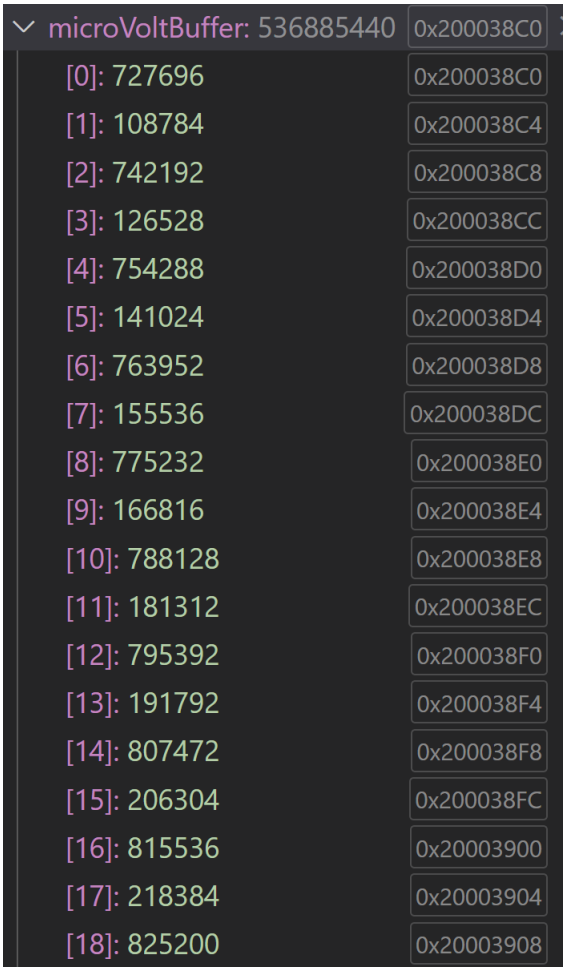

Figure 5-2. ADC Buffer Measurements

ENERGYTRACE has been used to evaluate the power consumption of the CC2340R5 MCU has been measured with consideration to both standby activity, where only periodic Proprietary RF radio transmit intervals occur, and instances for which the motor is actively being driven for a set amount of time. A comparison of these observations is shows below. The average standby current consumption can be further optimized to achieve less than 10 µA by further configuring the Proprietary RF definitions explained previously. The active motor measurements only take into account the CC2340R5, not the current draw required by the DRV8411EVM to drive the motor.

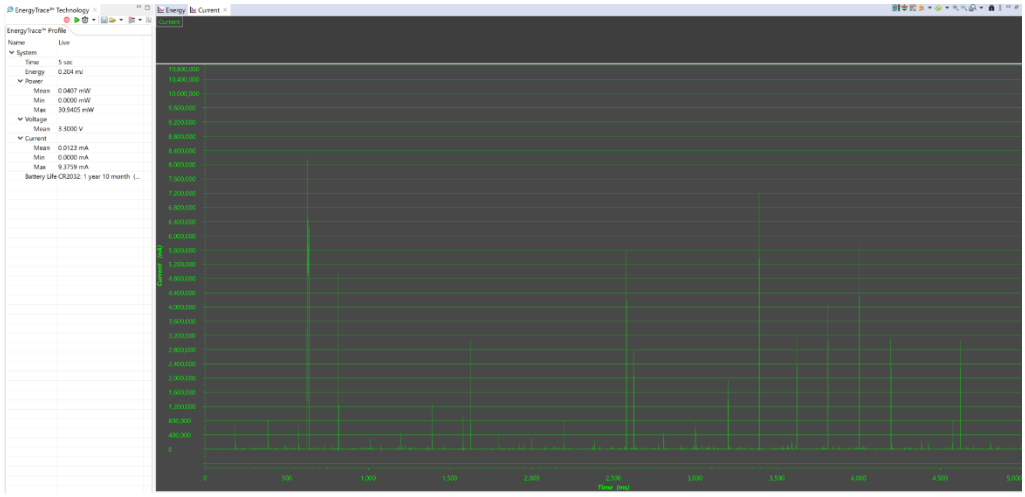

Figure 5-3. EnergyTrace CC2340R5 MCU Power Consumption Measurements of Standby Low Power Mode

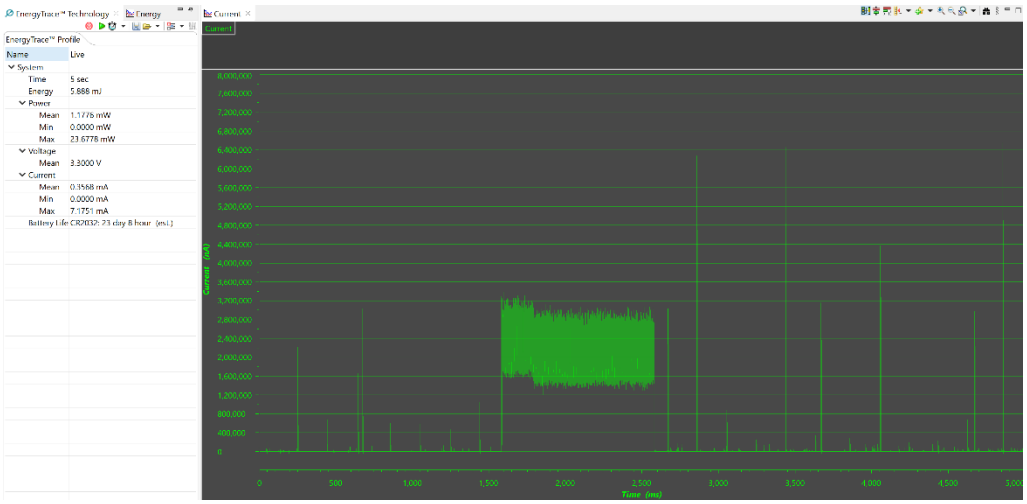

Figure 5-4. EnergyTrace CC2340R5 MCU Power Consumption Measurements during an Active Motor State

6 Summary

This application report has fully defined a stepper motor Proprietary RF solution using the SimpleLink CC2340R5 and DRV8411. 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 solution. 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 their unique application requirements. Readers are encouraged to post to the E2E forum concerning any additional questions or support needs as it pertains to these resources provided.

7 References

1. Texas Instruments, *CC2340R SimpleLink™ Family of 2.4GHz Wireless MCUs*, data sheet.
2. Texas Instruments, *LP-EM-CC2340R5* quick start guide.
3. Texas Instruments, *DRV8411 Dual H-Bridge Motor Driver with Current Regulation*, data sheet.
4. Texas Instruments, *DRV8410_DRV8411_DRV8411AEVM User's Guide*, user's guide
5. Texas Instruments, Proprietary RF documentation
6. Texas Instruments, PropRF SimpleLink Academy Labs
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