Application Note

Solving Sensorless Brushed DC Motor Speed and Position Control Using Ripple Counting

Karan Doshi and Jacob Thompson

ABSTRACT

This application report describes the integrated sensorless implementation of speed and position sensing using the current ripple of a brushed dc (BDC) motor. When an BDC motor application requires position and speed information of the motor, these are usually measured using encoders or hall effect sensors. Such designs are expensive, require more space, and are complex to design and maintain over the lifetime of an application. Texas Instruments DRV8214 and DRV8234 fully integrated BDC motor drivers eliminate the need for these external hall sensors and encoders. In addition to detailed description of the setup and tuning of this implementation, examples with test results are included. The accuracy of the test results under various test conditions is also analyzed.

Table of Contents

1 Introduction: Need for Sensorless Designs 2
2 Ripple Counting − Concept 2
2.1 Ripple Counting Algorithm Details 3
3 Case Study: Robotic Wheel Drive 4
3.1 Robotic Wheel Motor Operating Conditions 4
3.2 Tuning Parameters for Ripple Counting 4
3.3 Robotic Wheel Motor with Ripple Counting 6
4 Challenges and Workarounds 13
4.1 Low Average Currents 13
4.2 Motor Inertia During Stop 15
4.3 Inrush 17
4.4 High Load Conditions 18
5 Summary 20
6 References 20
7 Revision History 21

Trademarks

All trademarks are the property of their respective owners.

1 Introduction: Need for Sensorless Designs

Brushed DC motors require the knowledge of real-time motor speed, voltage, position, temperature, etc. for functioning in a closed loop system. While existing drivers integrate current, voltage, and temperature sensing, speed and position sensing require the use of external sensors like optical encoders and hall effect based sensors. This leads to:

- Increase in design complexity
- Increase in BOM count
- Increase in BOM cost
- Greater risk of failure due to added components, leading to overall reduced reliability

DRV8214 and DRV8234 implement an integrated ripple counting algorithm that enables sensorless speed and position sensing. This leads to significant system level benefits. Parameters can be tuned through I2C to achieve high accuracy results.

2 Ripple Counting − Concept

Ripple Counting works by detecting current ripples that occur during commutation of the armature windings in a BDC motor. Commutation is the process of current reversal in the windings of a BDC motor to make sure that the motor torque remains in the same direction during motor operation. The contact brushes used with the commutator cause momentary short circuits in the motor windings which appear as ripples in the observed motor current waveform. Each commutation corresponds to one ripple. Thus, counting the number of ripples over one revolution of the motor helps estimate the position of the motor from the initial position.

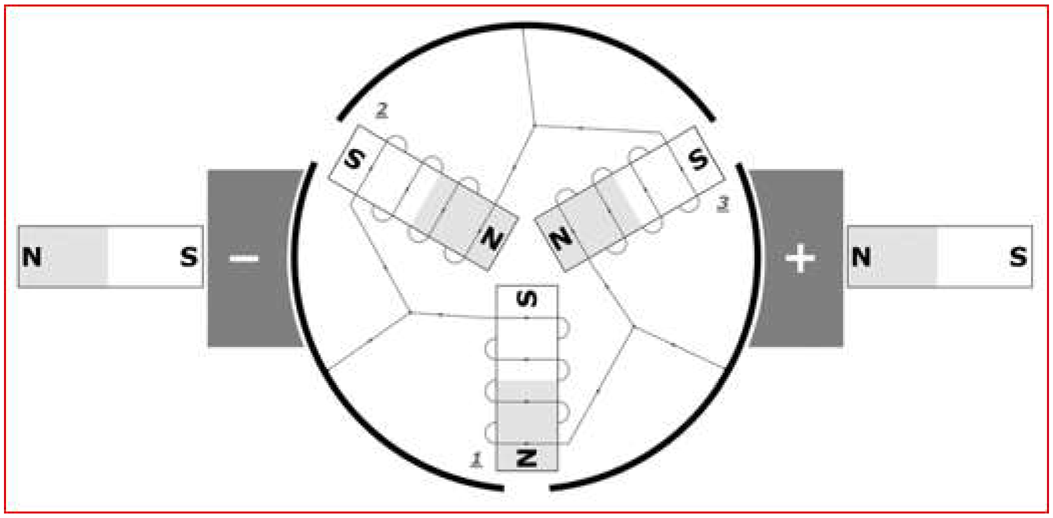


Figure 2-1. DC Motor Construction with 2 brushes and 3 commutators

Measuring the ripples per unit time provides the ripple speed (or ripple frequency), ω_R in Hz. To convert into rad/s, use the following equation:

$$\omega_R \text{ (in rad/s)} = \omega_R \text{ (in Hz)} \times 2\pi \quad (1)$$

Knowing the number of commutator segments and brushes, the motor speed, ω_M, can then be calculated by dividing the ripple frequency by the number of ripples per revolution. Number of ripples per revolution, N_R, can be found by taking the lowest common multiple (LCM) of the number of brushes, N_B, and the number of commutator segments, N_C.

$$N_R = \text{LCM}(N_B, N_C) \quad (2)$$

Hence, speed of the motor is given by:

$$\omega_M \left(\text{in rad/s}\right) = \frac{\omega_R \left(\text{in rad/s}\right)}{N_R} \tag{3}$$

$$\omega_M \left(\text{in rpm}\right) = \omega_M \left(\text{in rad/s}\right) \times \frac{60}{2\pi} \tag{4}$$

Please note that LCM can be easily calculated using an online calculator.

As an example, from Figure 2-1, we see that N_B =2, and N_C=3. Hence, N_R = LCM(2,3) = 6. This is verified by logic, since each brush short-circuits each pair of commutator segments once during each revolution. There are 2 brushes and 3 commutators causing 6 short circuits per revolution. As a result, there are 6 ripples per revolution. Figure 2-2 shows an example current waveform during steady state operation at 100% duty cycle.

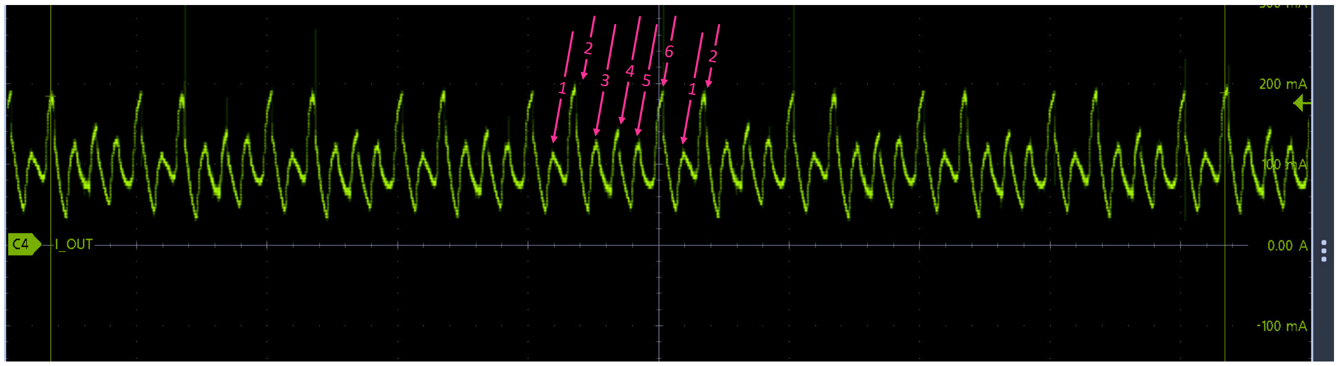


Figure 2-2. Brushed DC Motor Current Waveform with Ripples

2.1 Ripple Counting Algorithm Details

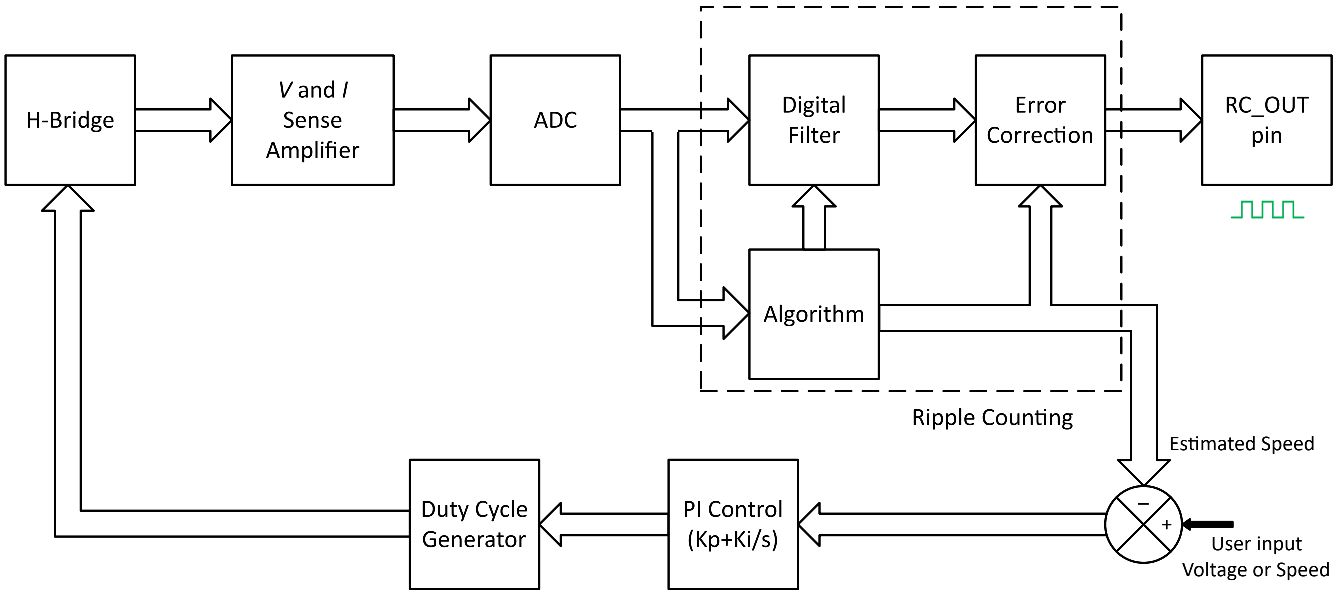


Figure 2-3. Ripple Counting Block Diagram

Figure 2-3 shows the block diagram for the ripple counting algorithm. The dynamic nature of the commutation process, along with brushes, generates unwanted noise and causes distortion of the current ripple waveform. This unwanted noise in the current signal is filtered out using a digital filter. To improve accuracy, an error corrector block performs signal conditioning by adding or subtracting a set number of ripple counts to the result to compensate for over-counting or under-counting due to excessive noise in the system. The conditioned ripple output waveform is output through the RC_OUT pin in DRV8214 and DRV8234. The PI (Proportional-Integral) control loop generates the duty cycle command based on the difference between desired and detected values for speed or voltage regulation.

3 Case Study: Robotic Wheel Drive

Autonomous Robots use brushed DC motors to drive the wheels that enable the robot move across floors. Owing to the autonomous nature of these devices, accurate and reliable position and speed control are critical to precise navigation and obstacle avoidance. This case study examines the tuning procedure and challenges of the ripple counting algorithm by taking a vacuum robot motor as an example. Optimized calibration through I^2C results in similar performance, consequently eliminating the need for encoders.

3.1 Robotic Wheel Motor Operating Conditions

The motor used in this example is a typical brushed DC motor driving the wheels of a vacuum cleaner robot. DRV8234 was used to drive the motor. Table 3-1 describes the operating conditions of the robotic wheel motor.

Table 3-1. Motor Operating Conditions

Parameter	Value
Supply Voltage	11V
Coil Resistance	10Ω(1)
Number of Commutators	3
Number of Brushes	2
Load	Unloaded

(1) Resistance was calculated as the average of 50 values at various voltage levels. Details on how to calculate motor resistance are provided in the data sheets for DRV8214 and DRV8234.

3.2 Tuning Parameters for Ripple Counting

The tuning process for the ripple counting algorithm is explained in the data sheets of DRV8214 and DRV8234. The results obtained in each step of the tuning process for the Robotic Wheel Motor example are described in this section. Please note that the tuning process is performed automatically in the evaluation module (EVM) boards using the GUI . Refer to the product pages for DRV8214 and DRV8234 for further details.

3.2.1 Resistance Parameters

This section refers to section 8.2.3.1.1 (Resistance Parameters) of the DRV8234 data sheet. The first step is to find the motor resistance. **Using the recommended method**, the robotic wheel motor's resistance was found out to be **10Ω**.

Note

To perform the voltage sweep:

1. Connect the motor directly to the power supply at a voltage just below where the motor starts to spin. For the robot wheel motor, this value was 1.2V.
2. Read the current using a current probe, inline multimeter, or power supply readout.
3. Calculate the motor resistance using the following equation: Motor Resistance = Voltage/Stall Current.
4. Repeat this test across a range of voltages. Note that the motor needs to be intentionally stalled at higher voltages.
5. Take the average of all values to calculate the value of motor resistance.

Taking the previous example of **10Ω**, we have the following possible results based on the choice of INV_R_SCALE:

Table 3-2. Selection Example for INV_R_SCALE and INV_R

Bit	INV_R_SCALE value	INV_R_SCALE/Motor Resistance (Actual Value)	Rounded Value INV_R	Comment
00b	2	2/10=0.2	0	Do not select, since output is 0.
01b	64	64/10=6.4	6	Avoid selecting, since low bit precision.
10b	**1024**	**1024/10=102.4**	**102**	**Can select this value.**
11b	8192	8192/10=819.2	819	Cannot select this value because 819 exceeds the maximum limit for INV_R (255).

Please note that the GUI does not calculate the motor resistance automatically. The GUI selects the appropriate values for INV_R and INV_R_SCALE based on the value input for motor resistance. For details on how to use the Evaluation Module (EVM) and GUI , please refer to the EVM User Guide.

3.2.2 KMC and KMC_SCALE

Selection of KMC_SCALE and KMC can be divided into two cases:

1. Value of the motor back emf constant, K_V is known to the user from the data sheet of the motor. Please refer to section 8.2.3.1.2.1 in DRV8234 data sheet for the tuning guide in this case.
2. Value of the motor back emf constant, K_V is unknown to the user. Please refer to section 8.2.3.1.2.2.1 in DRV8234 data sheet for the tuning guide in this case.

For the robot wheel motor, **the value of K_V was unknown**. Thus, step 2 was followed. KMC and KMC_SCALE need to be tuned manually from scratch. Both parameters are reset at the start before tuning is begun.

3.2.2.1 Tuning KMC_SCALE

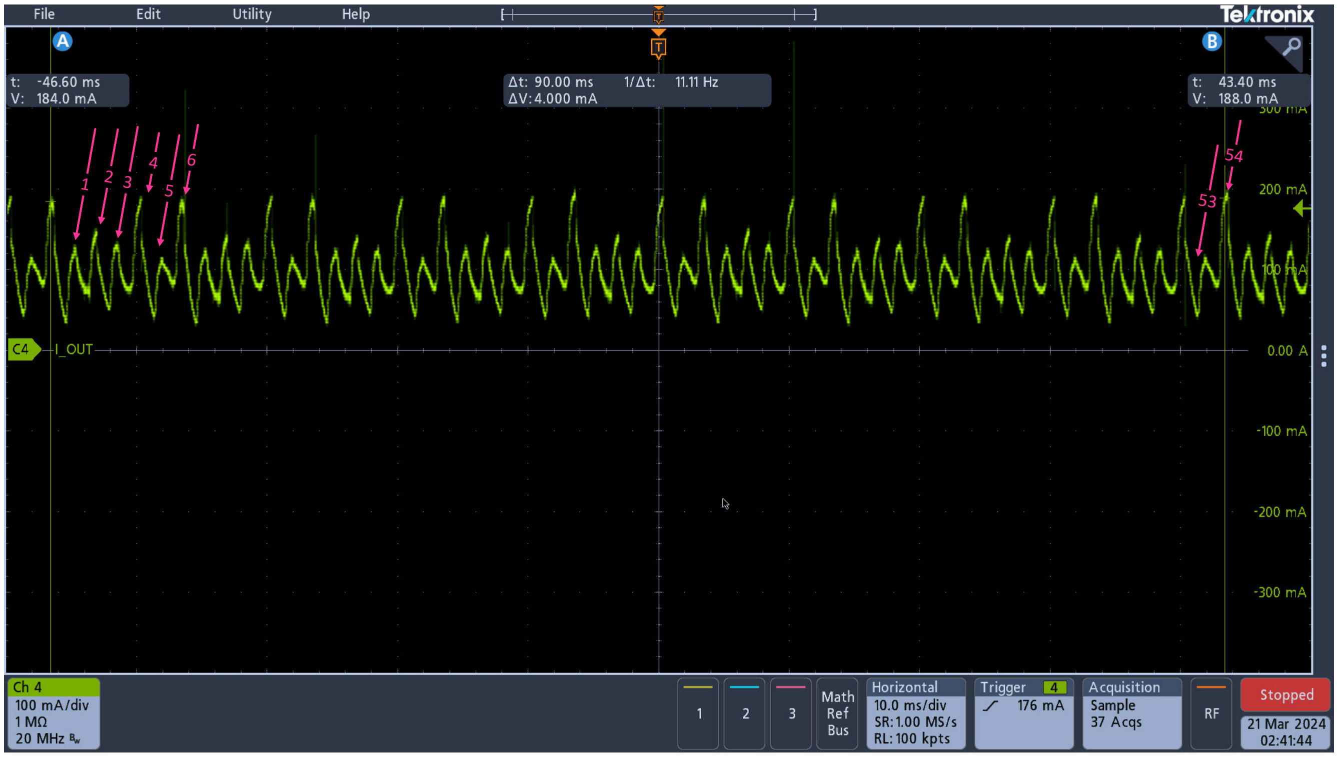


Figure 3-1. Ripple Frequency Calculation

1. From section 8.2.3.1.2.2.1.1, the first step in tuning KMC_SCALE is to obtain the value of actual ripple speed in rad/s. For this method, the motor was connected to DRV8234 at 11V at no-load conditions. The motor current waveform was observed on the oscilloscope using a current probe to obtain the ripple frequency in Hz as shown in Figure 3-1. The number of ripples (54) observed was divided by the time (90ms) to obtain ripple frequency as 600Hz. Using Equation 1, the ripple frequency in rad/s was calculated as 3769.91 rad/s. Let OBS_SPEED = **3770 rad/s**.
2. The next step is to select the lowest value of KMC_SCALE, 00b and set KMC to the highest possible value, 255.
3. The next step is to set W_SCALE set to a value under which maximum allowable ripple speed is more than OBS_SPEED. In this example, W_SCALE was set to 00b (allowing a maximum value of 4080 rad/s)*, for example, 16 rad/s.
4. Following the next steps as mentioned in the data sheet, we find that the **tuned value of KMC_SCALE for this example is 10b**. Figure 8-3 (KMC_SCALE Tuning Procedure) in the data sheet for DRV8234 displays a flowchart for tuning KMC_SCALE. The iterative steps for tuning KMC_SCALE in this example are summarized in Table 3-3.

Table 3-3. KMC_SCALE tuning

Iteration	KMC_SCALE	KMC	SPEED	W_SCALE	EST_SPEED (rad/s)	OBS_SPEED (rad/s)	Decision
1	00b	255	0x0F (15)	00b	240	3770	EST<OBS
2	**01b**	255	0x1E (30)	00b	480	3770	EST<OBS
3	10b	255	0xF0 (240)	00b	3840	3770	EST>OBS

3.2.2.2 Tuning KMC

1. From section 8.2.3.1.2.2.1.2 in the data sheet, the first step to tune KMC is to verify that EST_SPEED < OBS_SPEED and value of KMC is 255. If this is not the case, please restart the tuning process.
2. Following the next steps from the data sheet, we find that the **tuned value of KMC is 32**. Figure 8-4 (Binary Search Algorithm to Find KMC) in the DRV8234 data sheet displays a flowchart for tuning KMC using the Binary Search Algorithm. The iterative steps for tuning KMC in this example are summarized in Table 3-4,

Table 3-4. KMC Tuning

Iteration	KMC_SCALE	KMC	START	MID	END	EST_SPEED (rad/s)	OBS_SPEED (rad/s)	Decision
1	10b	1	1	128	255	4080	3770	EST>(OBS+16)
2	10b	128	1	64	128	960	3770	EST<(OBS-16)
3	10b	64	1	32	64	1920	3770	EST<(OBS-16)
4	10b	32	32	48	64	3776	3770	EST within OBS±16

3.3 Robotic Wheel Motor with Ripple Counting

The tuned parameters can be found in Table 3-5. Please assume default values for the writable register values not mentioned. Please note that it is advisable to turn on the error corrector (DIS_EC=0b) only while PWM-ing since current waveform is noisy.

Table 3-5. Tuned Parameter Values for the Vacuum Robot Wheel Motor

Parameter/Register	Bits/Decimals	Value
EN_RC	1b	-
W_SCALE	01b	32 rad/s
INV_R_SCALE	10b	1024
INV_R	0x66	102
KMC_SCALE	01b	24×2^9
KMC	0xFC	252
FLT_GAIN_SEL	11b	16
FLT_K	0110b	6

Note

The following sections demonstrate the performance of the ripple counting algorithm in various situations. Please note that:

1. RC_OUT output is denoted in yellow.
2. Motor current is observed using a current probe and shown in green.
3. Encoder output is shown in pink.
4. Accuracy, is calculated against the output of an encoder. Encoder emits 4 pulses per rotation. RC_OUT emits 6 counts per rotation. Thus, accuracy can be calculated using the following equation:

$$\text{Accuracy} = \left[\frac{\text{No. of RC\_OUT counts}}{6} \Big/ \frac{\text{No. of encoder counts}}{4} \right] \times 100\% \tag{5}$$

3.3.1 Inrush and Steady State Performance

In brushed DC motors, there is a large inrush current during start-up due to the absence of any back emf. Figure 3-2 shows the performance of ripple counting algorithm during inrush using the tuned parameters mentioned in Table 3-5. To improve accuracy, the T_MECH_FLT register was set to 000b, displayed in Figure 3-3. Please refer to Section 4.3 for the description of the T_MECH_FLT register. Section 4.3 explains more workarounds for transient conditions, including inrush. Operating conditions are unchanged from the tuning process. PWM is at 100% duty cycle. For examples on how to workaround low average currents due to PWM at low duty cycles, please refer to Section 4.1.

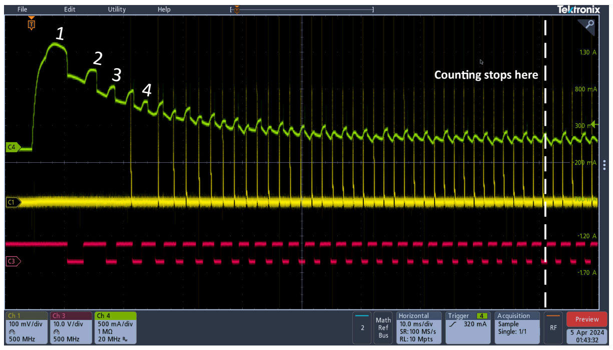


Figure 3-2. Inrush Current at 11V, 100% Duty Cycle

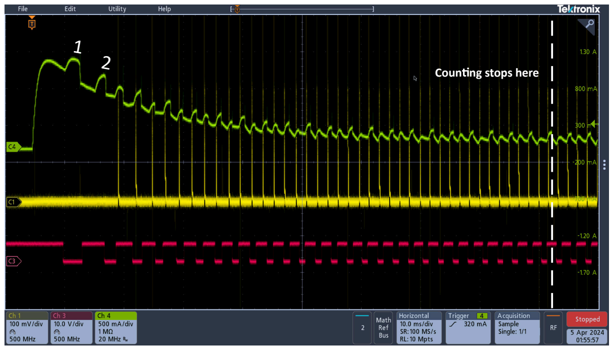


Figure 3-3. Improved Performance using T_MECH_FLT

Using Equation 5, accuracy calculations for Figure 3-2 and Figure 3-3 are described in Table 3-6. In the figures, counting is stopped when the motor completes 7 full revolutions since the motor current reaches steady state value.

Table 3-6. Accuracy During Inrush

Parameter	Untuned	Tuned
Encoder Counts	28	28
RC_OUT Counts	38	40
Accuracy	90.5%	95.2%

Figure 3-4 shows the performance of the ripple counting algorithm during steady state. Clearly, all ripples are tracked accurately through the RC_OUT pin. Table 3-7 compares ripple counting accuracy against an encoder.

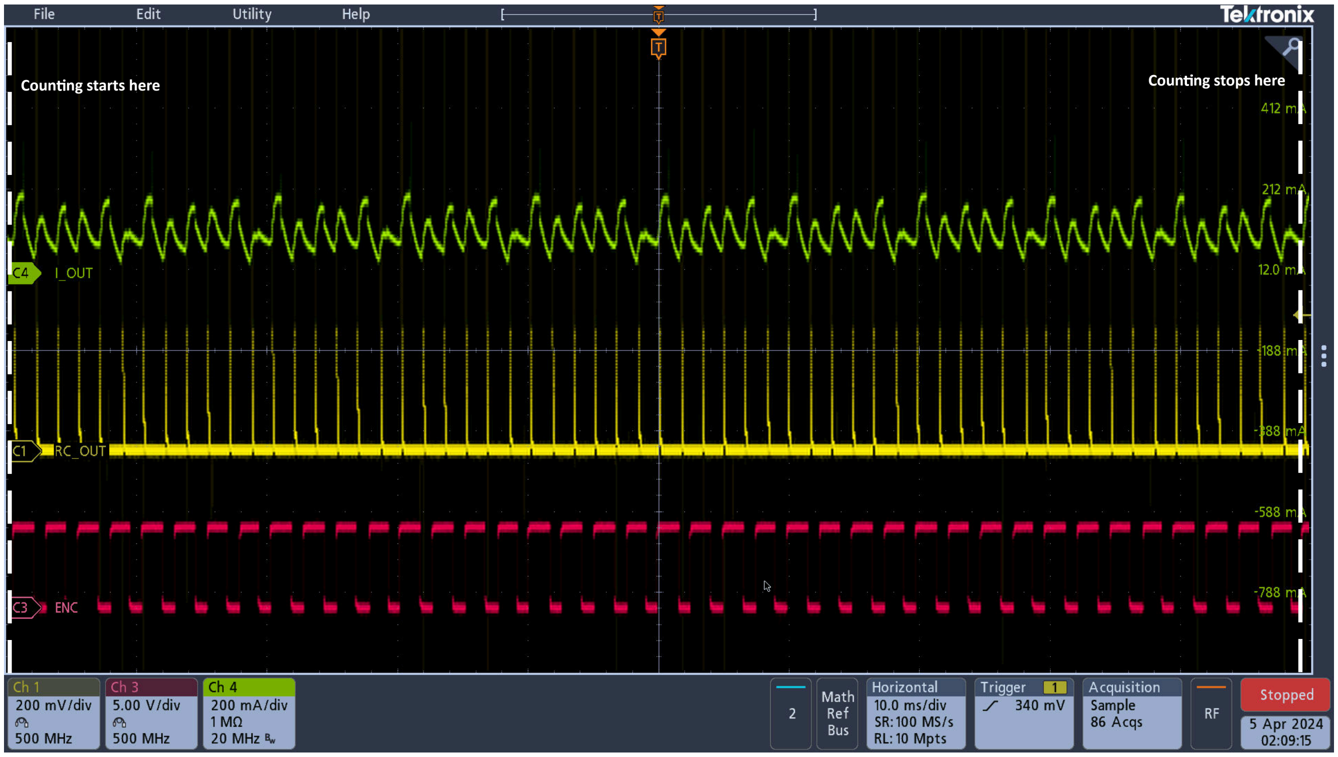


Figure 3-4. Performance During Steady State

Table 3-7. Accuracy During Steady State

Parameter	Steady State
Encoder Counts	40
RC_OUT Counts	60
Accuracy	100%

3.3.1.1 Motor Speed Calculation

To calculate motor speed and verify that ripple counting gives out accurate motor speed, we need to know the actual motor speed first. This is done with the help of the encoder. From Figure 3-5, the encoder gives out 32 pulses in 80.3ms. Since the encoder also gives out 4 pulses per motor rotation, the actual motor speed, is given by:

$$\text{Motor Speed (rpm)} = \frac{32}{80.3} \times \frac{1}{4} \times 1000 \times 60 = 5977.58\ \text{rpm} \quad (6)$$

From Figure 3-6, the RC_OUT pin gives out 48 pulses in 80.3ms. Since RC_OUT gives out 6 pulses per motor rotation, the calculated motor speed using RC_OUT is given by:

$$\text{Motor Speed (rpm)} = \frac{48}{80.3} \times \frac{1}{6} \times 1000 \times 60 = 5977.58\ \text{rpm} \quad (7)$$

Thus, ripple counting output matches encoder output during steady state conditions.

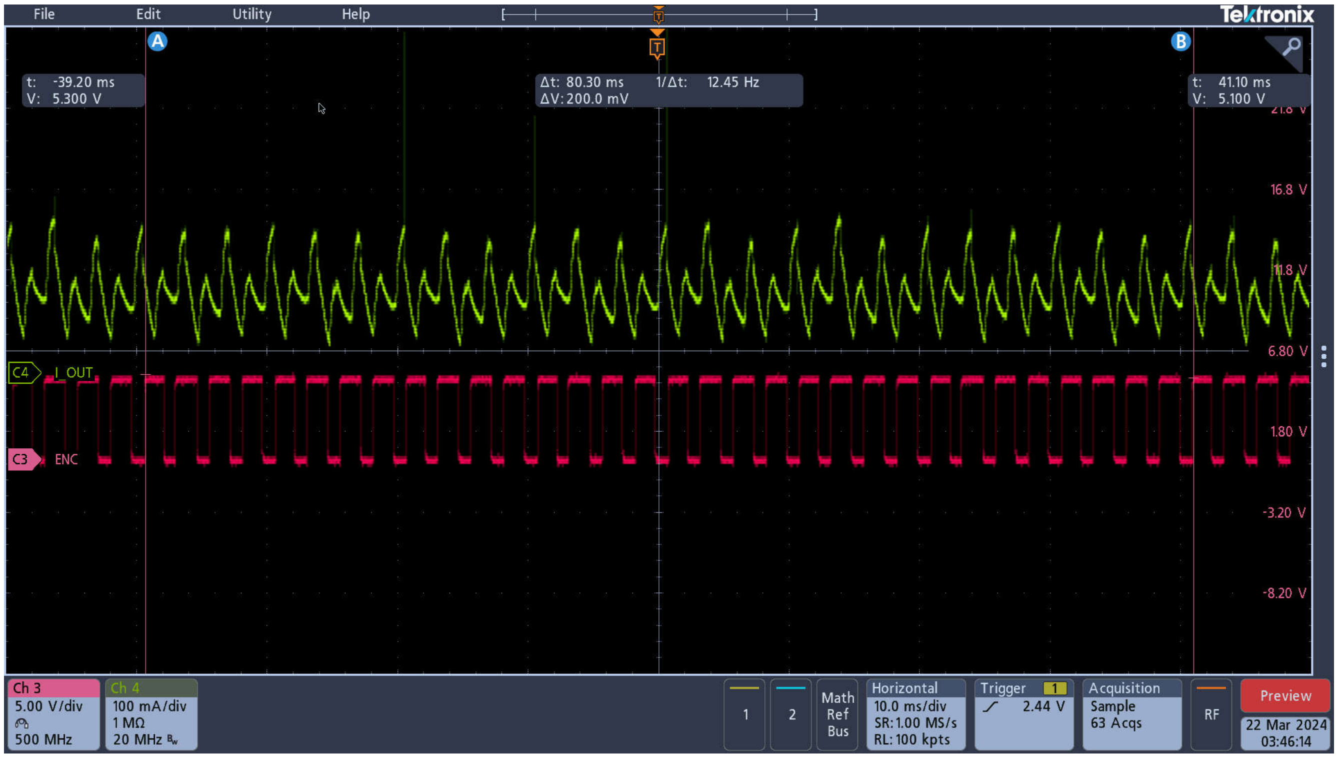


Figure 3-5. Actual Motor Speed Using Encoder

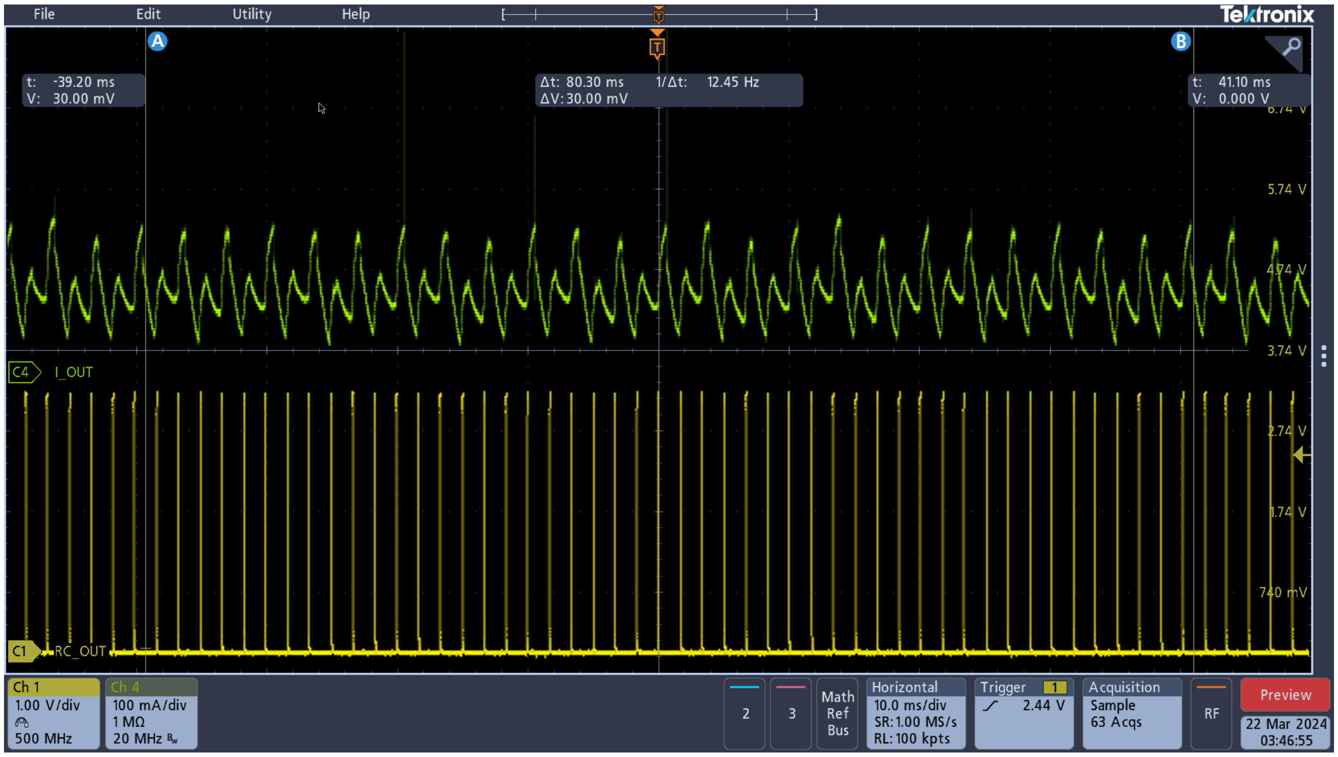
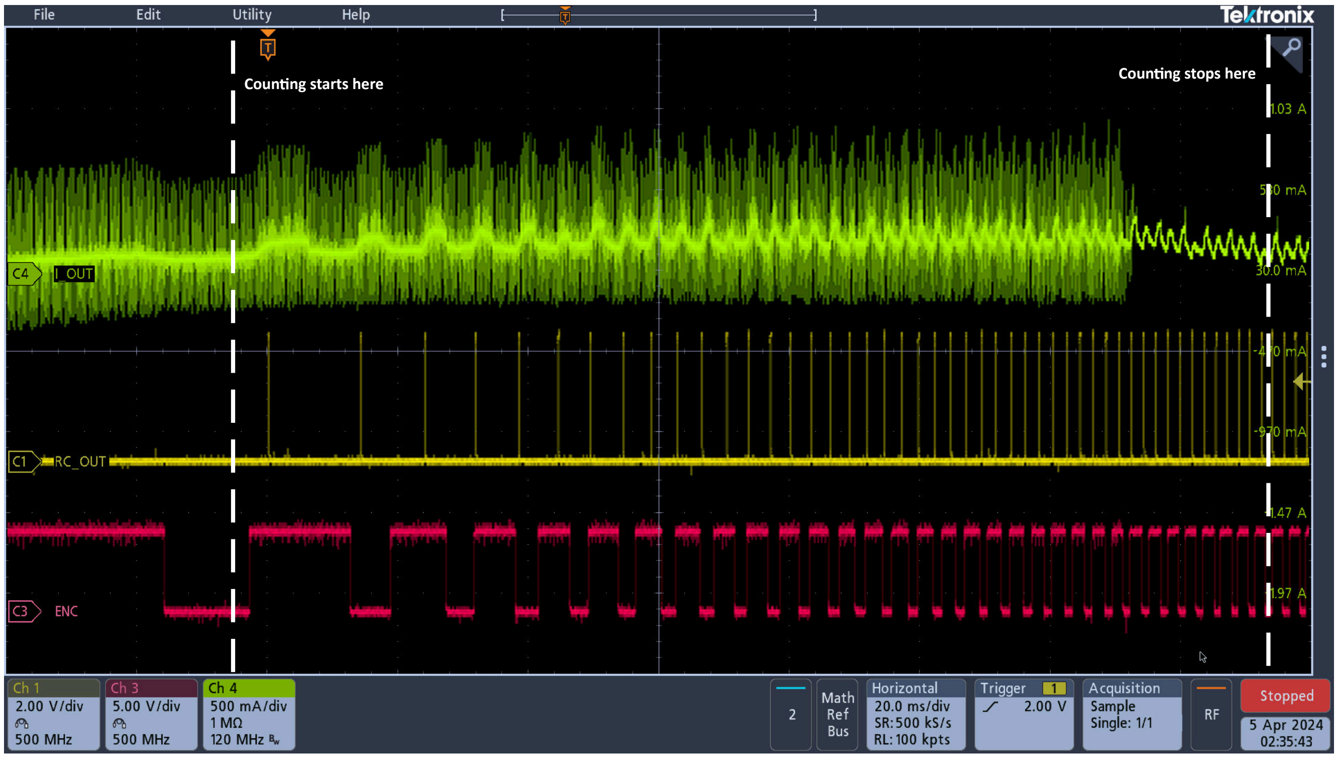


Figure 3-6. Motor Speed Calculated Using RC_OUT

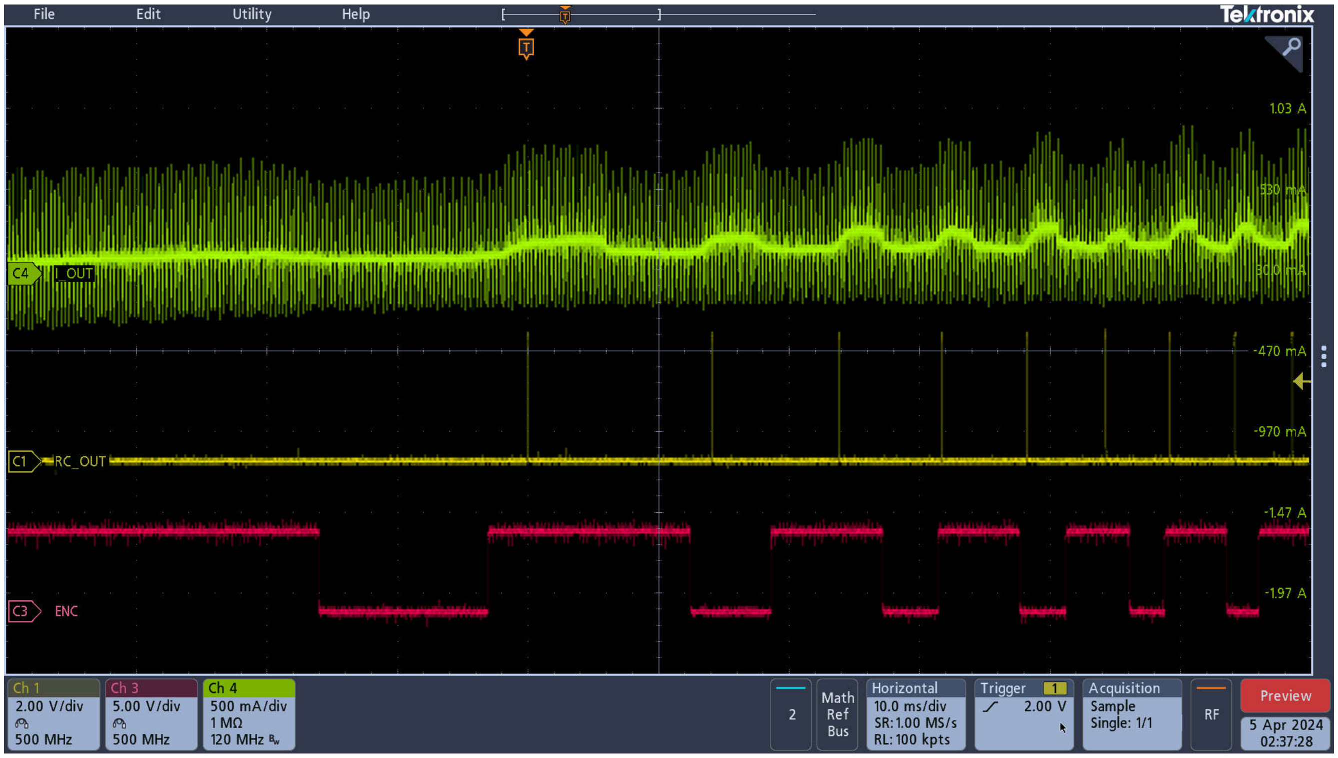
3.3.2 Soft Start

To protect against high inrush current transients as observed in Section 3.3.1, DRV8214 and DRV8234 have a soft-start feature that allows current to ramp-up to a preset value during the speed and voltage regulation modes. Figure 3-7 demonstrates the performance during soft start.



Ch 1 2.00 V/div 500 MHz; Ch 3 5.00 V/div 500 MHz; Ch 4 500 mA/div 1 MΩ 120 MHz; Horizontal 20.0 ms/div SR: 500 kS/s RL: 100 kpts; Trigger 1 2.00 V; Acquisition Sample Single: 1/1; 5 Apr 2024 02:35:43

Figure 3-7. Performance During Soft Start



Ch 1 2.00 V/div 500 MHz; Ch 3 5.00 V/div 500 MHz; Ch 4 500 mA/div 1 MΩ 120 MHz; Horizontal 10.0 ms/div SR: 1.00 MS/s RL: 100 kpts; Trigger 1 2.00 V; Acquisition Sample Single: 1/1; 5 Apr 2024 02:37:28

Figure 3-8. Soft Start - Zoomed

From Figure 3-7, we can see that all ripples are accurately tracked during soft start. This is validated by the accuracy calculation shown in Table 3-8. Counting is stopped when the motor completes 8 revolutions since the motor current value reaches steady state.

Table 3-8. Accuracy During Soft-Start

Parameter	Soft-Start
Encoder Counts	32
RC_OUT Counts	48
Accuracy	100%

3.3.3 Loaded Conditions

Steady-state current waveform for on-load conditions is shown in Figure 3-9. Full load (stall) current was observed to be 1.3A. Motor was loaded at 50% of the full load current value, for example, 650mA. PWM duty cycle is 100%.

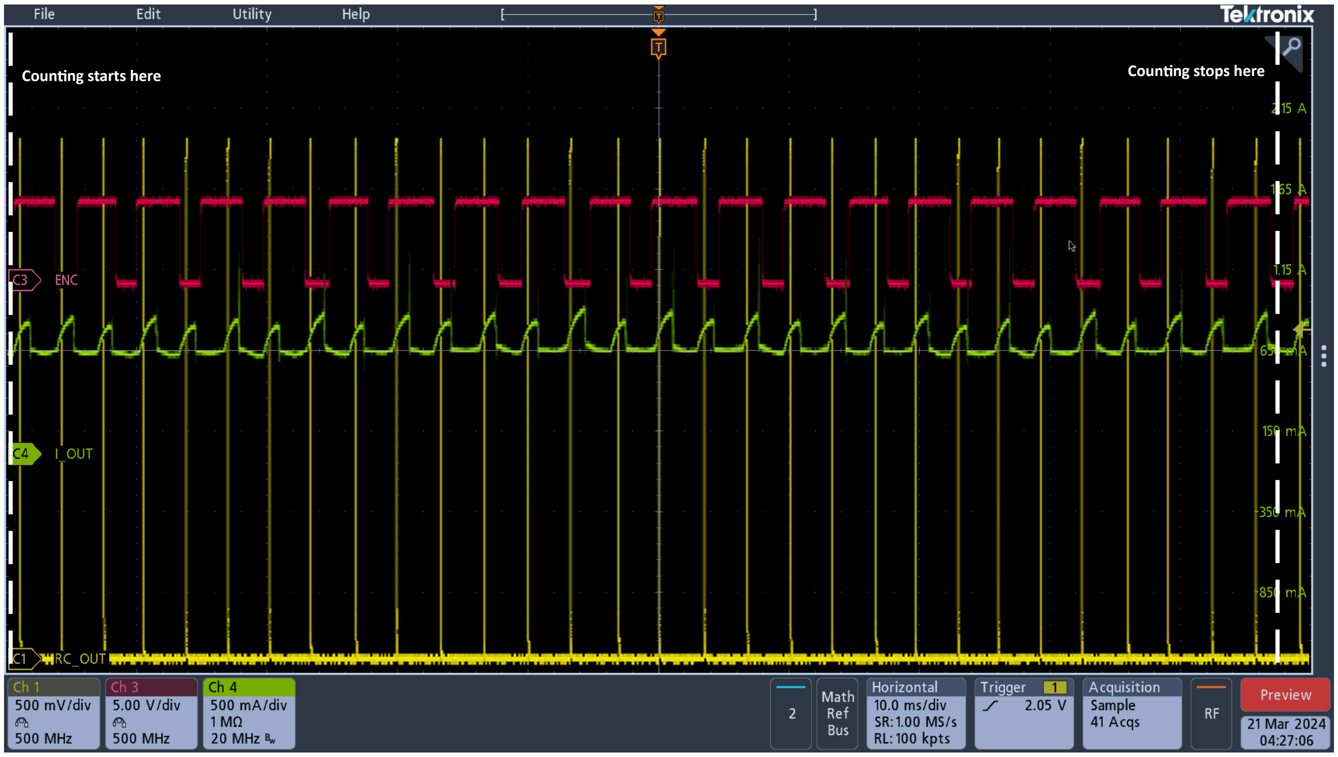


Figure 3-9. Steady State Performance at 50% Load

Counting is done for 5 revolutions at steady state. Calculation for accuracy is shown in Table 3-9.

Table 3-9. Accuracy at 50% Load

Parameter	50% Load
Encoder Counts	20
RC_OUT Counts	30
Accuracy	100%

Figure 3-10 displays the performance of ripple counting during transitory loading. Change in load occurs from 10% to 60% over approximately 80ms. Counting is performed for 6 revolutions of the motor.

Figure 3-11 displays the performance of ripple counting during unloading. Change in load occurs from 60% to 10% over approximately 80ms. Counting is performed for 6 revolutions of the motor.

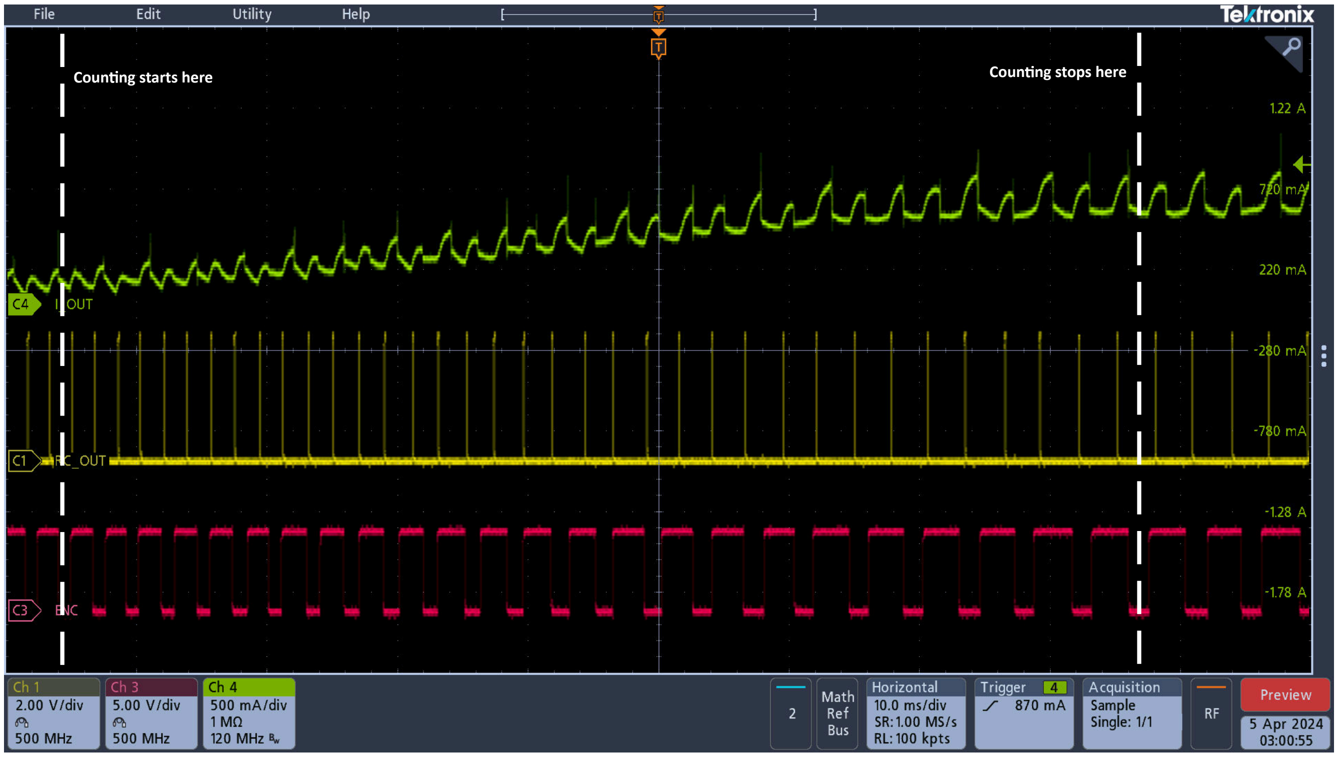


Figure 3-10. Transient Loading

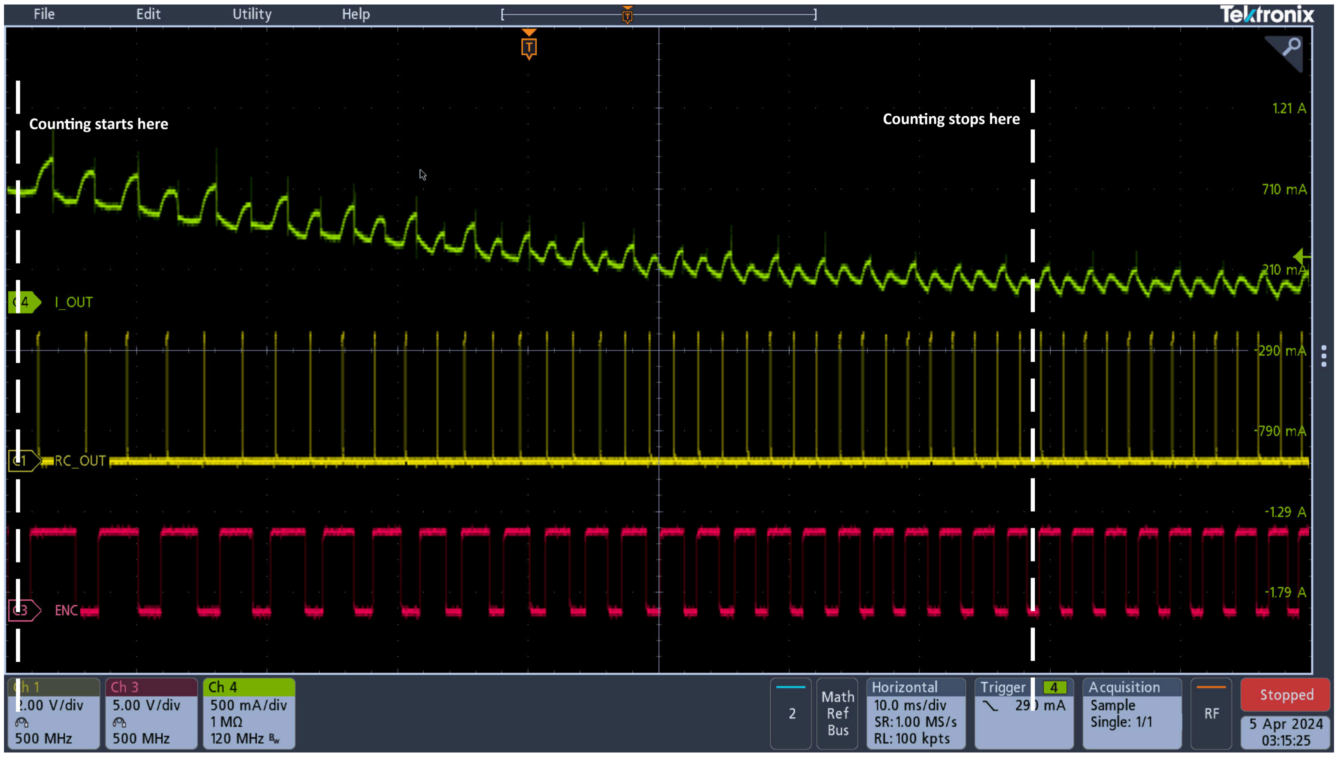


Figure 3-11. Transient Unloading

Table 3-10 shows the calculation for accuracy during both conditions.

Table 3-10. Accuracy During Transient Conditions

Parameter	Transient Loading	Transient Unloading
Encoder Counts	24	24
RC_OUT Counts	36	36
Accuracy	100%	100%

4 Challenges and Workarounds

This section describes a few challenges for the ripple counting algorithm in DRV8214 and DRV8234 despite following the tuning procedure. The workarounds to track ripples better and improve accuracy for each case are described alongside.

4.1 Low Average Currents

Operation at low average current presents challenges due to the reduced signal to noise ratio (SNR). Usually, this happens at:

1. Low duty cycle during PWM
2. Low value of motor current's DC component

As shown in Figure 4-1, the current ripples are highly distorted at low average motor current. The device is unable to distinguish between noise and current ripples. For the vacuum robot wheel motor example, this value was observed to be 90mA. The waveform was obtained by setting the PWM duty cycle to 30% at 11V.

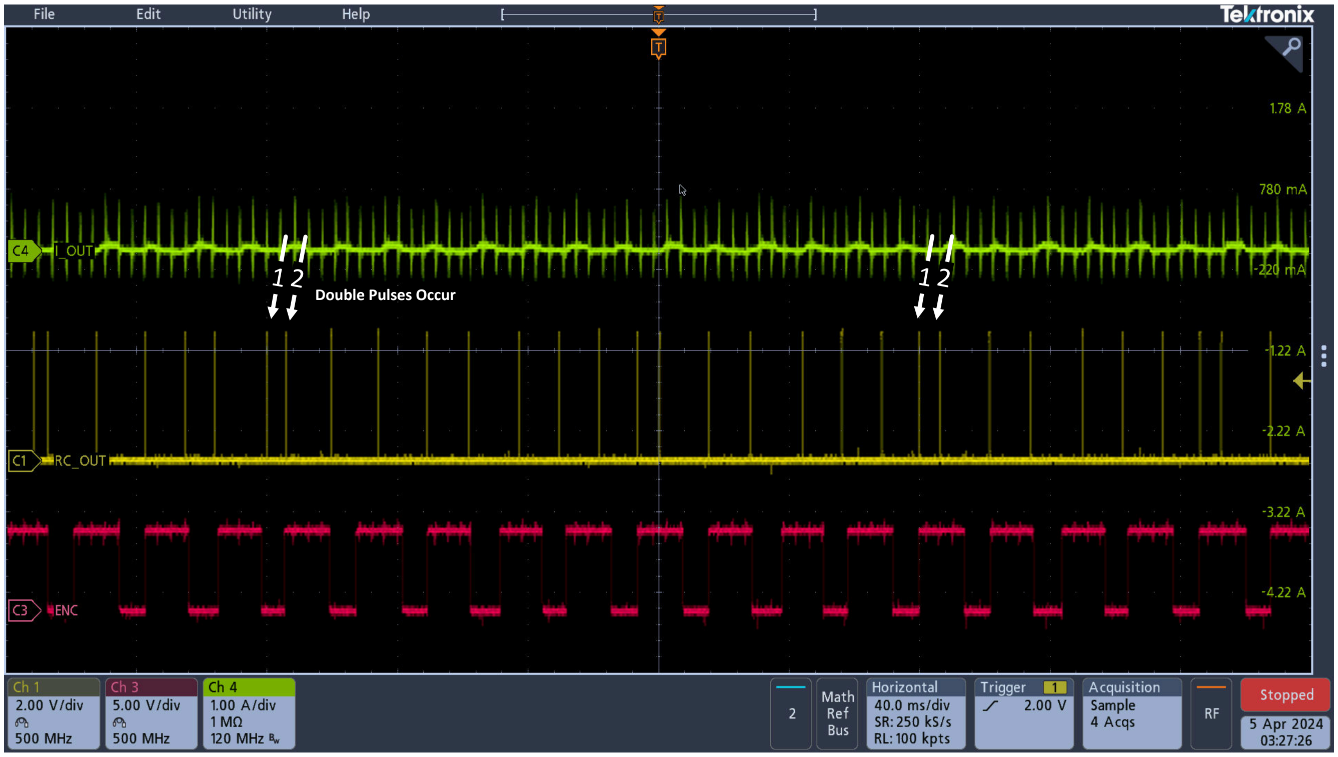


Figure 4-1. Low Current Performance

The possible workarounds to improve accuracy are listed as follows:

1. Set FLT_GAIN_SEL to 11b to utilize the full signal range. This assists the digital filter in differentiating between noise and current ripples.
2. If possible, lower the supply voltage and increase the PWM duty cycle such that average current remains the same. This improves the current ripple waveform.
3. The SPEED register value fluctuates during the tuning process, thereby facilitating erroneous KMC. Tweak the value of KMC to check if performance improvement is yielded.

4. Tweak the values of EC_FALSE_PER and EC_MISS_PER registers to see if tuning improves. For a detailed description, refer to the data sheet of DRV8234. As an example, Figure 4-2 shows the performance when EC_FALSE_PER = 10b and EC_MISS_PER = 10b from the default values of 01b. Table 4-1 calculates the accuracy in this case.

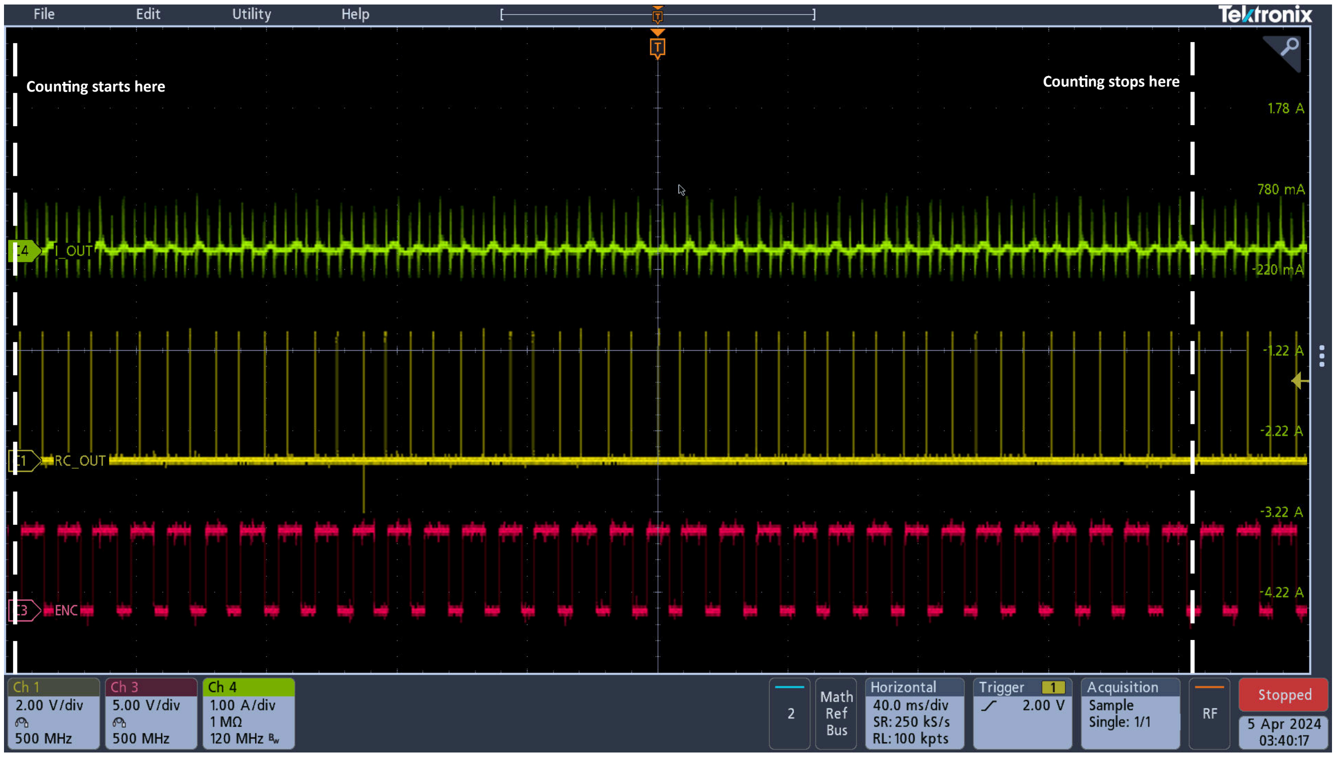


Figure 4-2. Tweaking Error Corrector Parameters

5. Re-tune the values for INV_R, INV_R_SCALE, KMC, and KMC_SCALE. Make sure that the registers are tuned as per instructions in Section 3.2.1 and Section 3.2.2.
6. As mentioned in section 8.2.3.1.2.2.2 of the data sheet for DRV8234, multiple pairs of tuned values for KMC and KMC_SCALE exist. Choose another possible pair.
7. Turn off the error corrector by setting DIS_EC = 1b. This also leads to improved performance as shown in Figure 4-3.

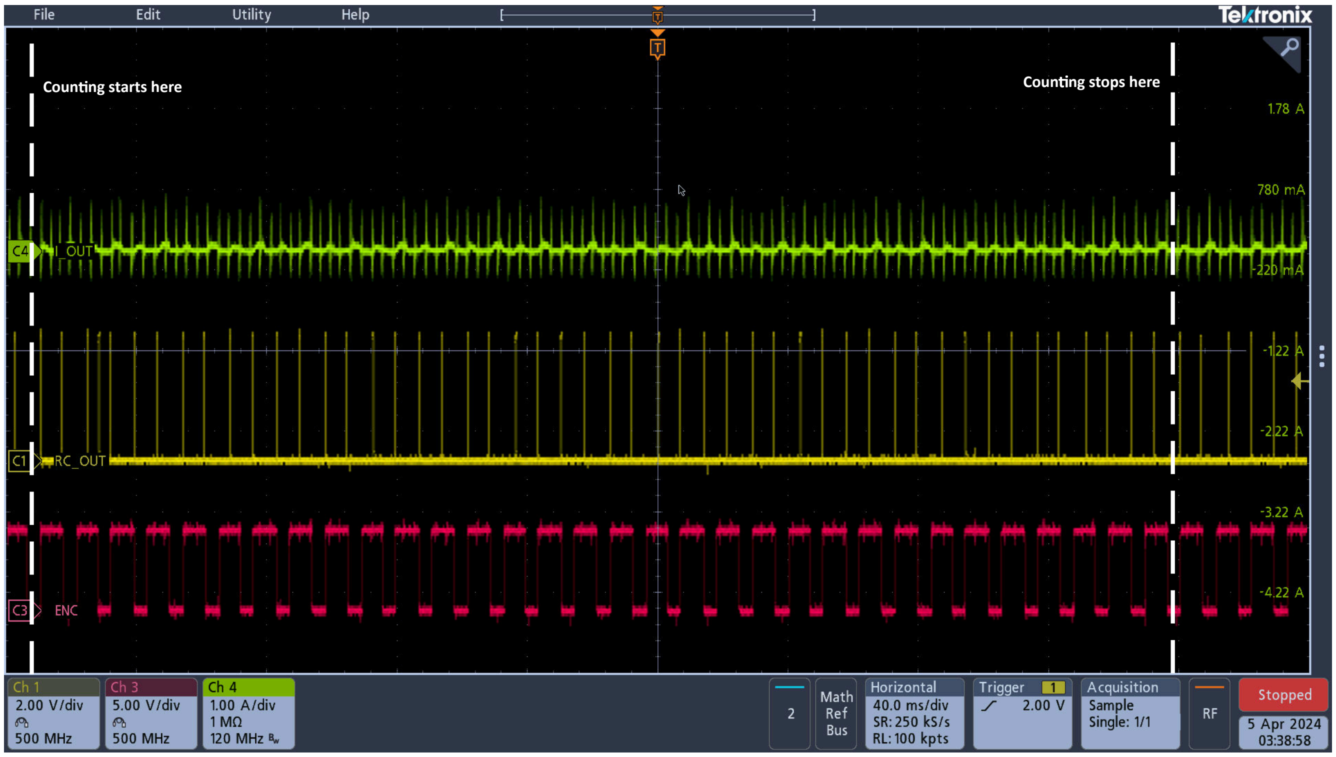


Figure 4-3. Error Corrector Turned off

Table 4-1. Improved Accuracy During Low Currents

Parameter	30% Duty	Tweaked Error Corrector Registers	Error Corrector turned off
Encoder Counts	16	32	32
RC_OUT Counts	28	48	49
Accuracy	116%(1)	100%	102%(2)

(1) Accuracy > 100% indicates that the algorithm counted some extra ripples. The percentage error can be calculated by subtracting the accuracy from 100. Hence, error at 30% duty cycle operation ios 16% over 4 revolutions.
(2) Error when the error corrector is turned off comes down to 2%.

4.2 Motor Inertia During Stop

In this case, the RC_OUT pin does not give out pulses when the current to the motor is turned off either during Hi-z or brake modes. Due to inertia, the motor continues to rotate for some time before coming to a halt, during which ripples fail to be detected. This is due to change in direction of current as observed in Figure 4-4. RC_OUT stops giving out pulses whereas the motor continues to rotate due to inertia.

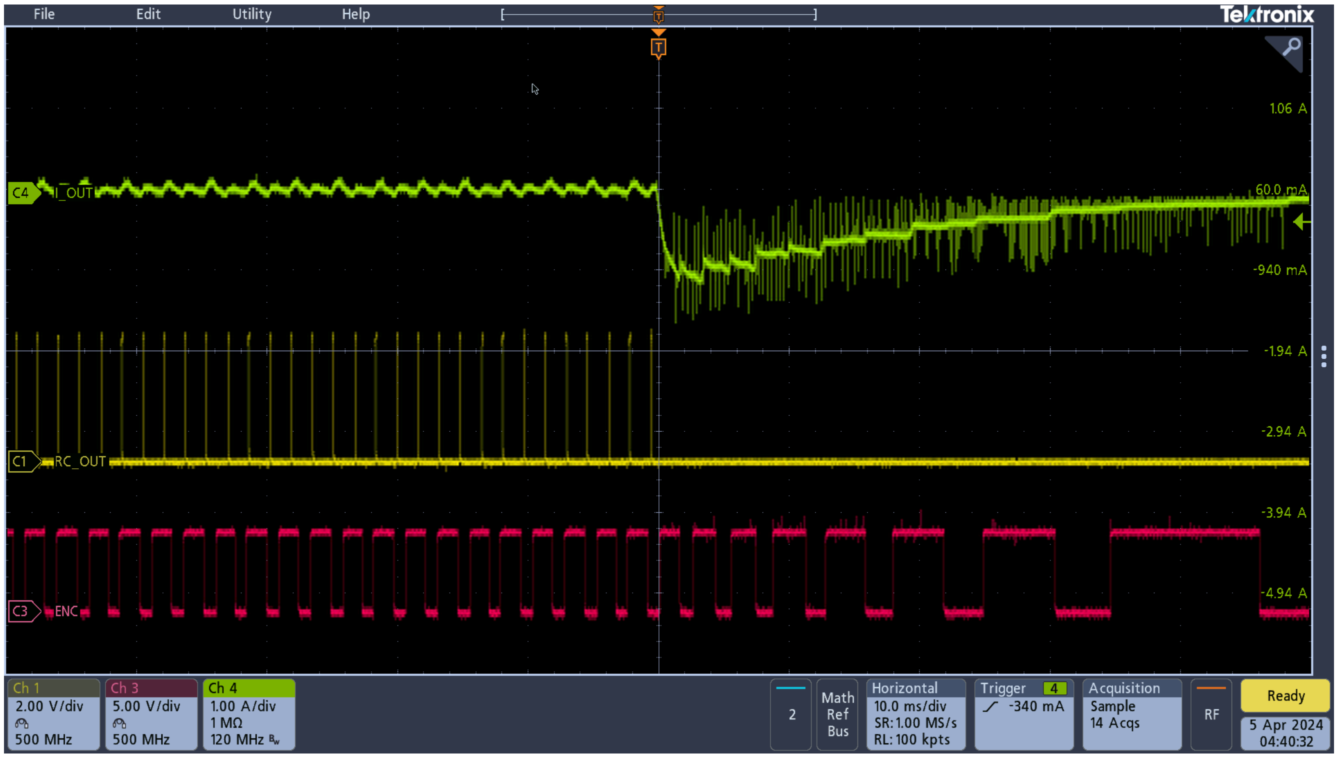


Figure 4-4. RC_OUT Stops Pulse Output During Normal Stop

A possible workaround for this case is to use the soft-stop feature (EN_SS = 1b). This allows the motor to come to a halt gradually which enables detection of ripples for a longer time leading to a lesser percentage of missed ripples. Figure 4-5 shows the performance of ripple counting during soft-stop. TINRUSH[15:8] (MSB) was set to 0x08.

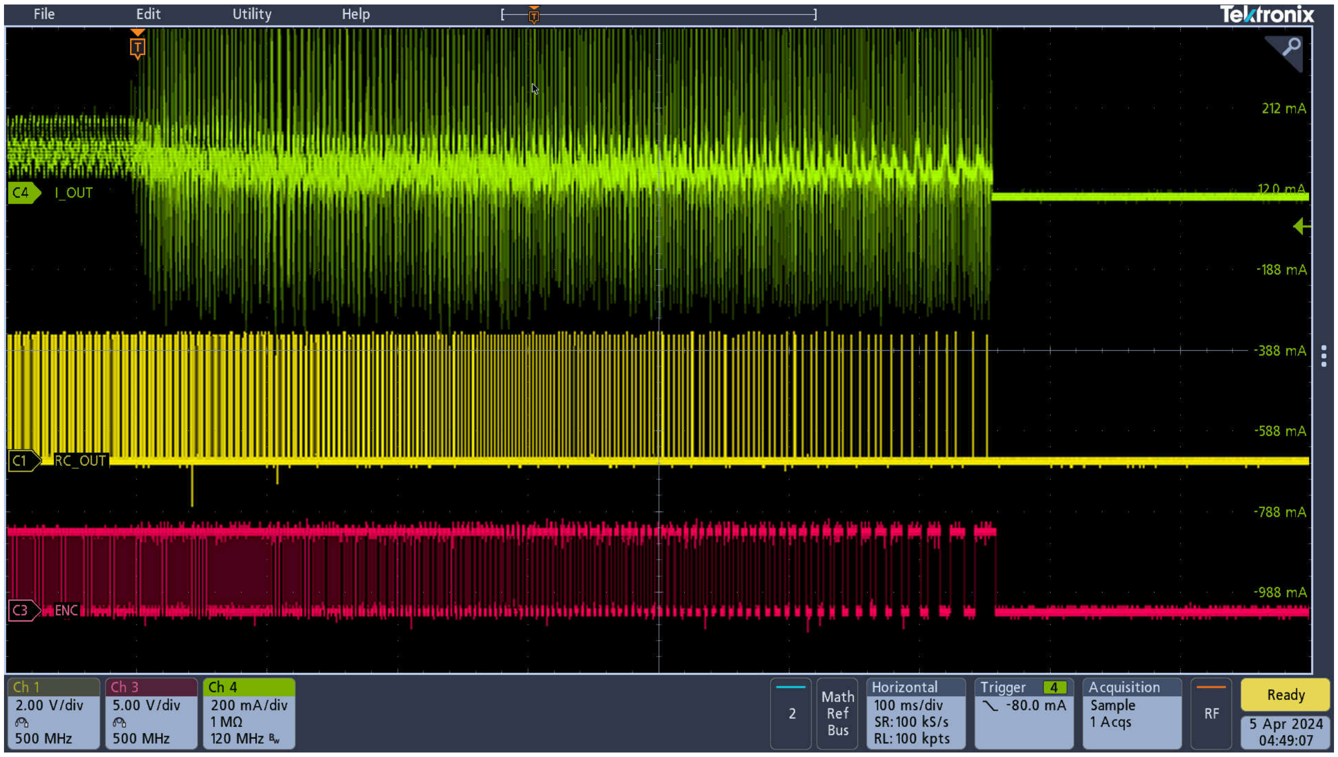


Figure 4-5. RC_OUT Tracking Pulses During Soft Stop

As explained in the data sheets of DRV8214 and DRV8234, the soft-start-stop feature is activated only during motor voltage or speed regulation. To use the soft-start-stop feature without speed regulation, use the following steps:

1. Set REG_CTRL register to 10b (speed regulation mode).
2. Set W_SCALE to 11b (128 rad/s).
3. Set WSET_VSET to 0xFF (255).
4. Set TINRUSH to an appropriate value.

This sets the speed reference value to the highest possible value (255*128 rad/s = 32640 rad/s). Since the upper limit for the duty cycle is 100% at a particular voltage level, this effectively means that the device is unable to regulate speed, thereby allowing the soft-start-stop feature to be used without speed or voltage regulation. If PWMing externally, programming the duty cycle to reduce gradually to 0 resembles soft-stop behavior.

4.3 Inrush

This case covers possible workarounds for ripples missed during motor start-up due to inrush current as observed in Figure 3-2.

1. Adjust the T_MECH_FLT register to match the system response time more accurately, as shown in Figure 3-3. The trade-off here is to sacrifice some accuracy during steady-state performance.
2. Using current regulation during inrush time also helps the algorithm track ripples more accurately.
3. If using internal speed or voltage regulation feature, enable soft-start and soft-stop (EN_SS = 1b).
4. If using DRV8214, program the controller to change CS_GAIN_SEL dynamically when loading/unloading is expected in DRV8214. For more details on functionality of CS_GAIN_SEL, refer to the data sheet.
5. Under same operating conditions and settings, the device consistently misses the same number of ripples each time. From Figure 4-6, Figure 4-7, Figure 4-8, and Figure 4-9, shows that the device misses 4 ripples during each start up. This can easily be accounted for simply by subtracting the missed ripples from the target ripple count. Programming the control unit to offset for four ripples enables position tracking with better accuracy.

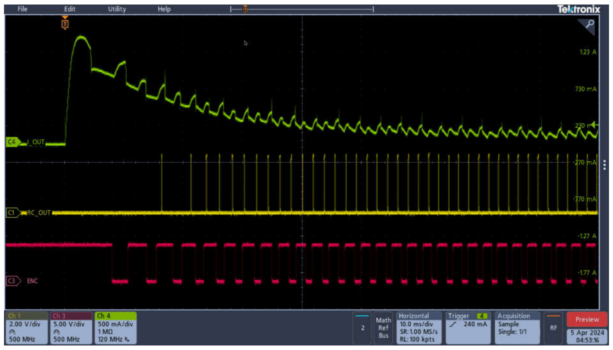


Figure 4-6. Inrush 1

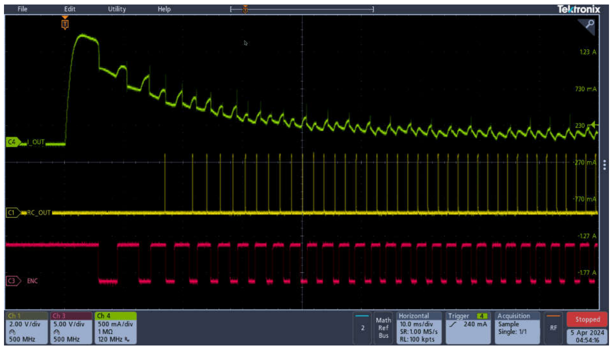


Figure 4-7. Inrush 2

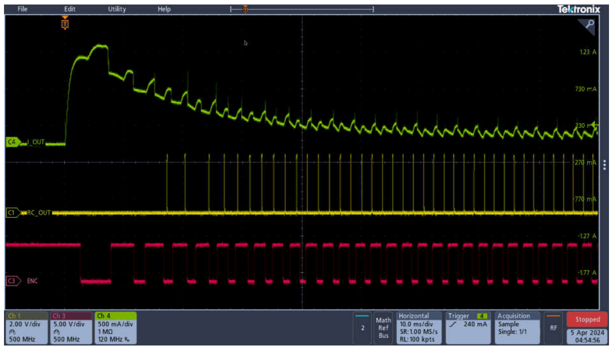


Figure 4-8. Inrush 3

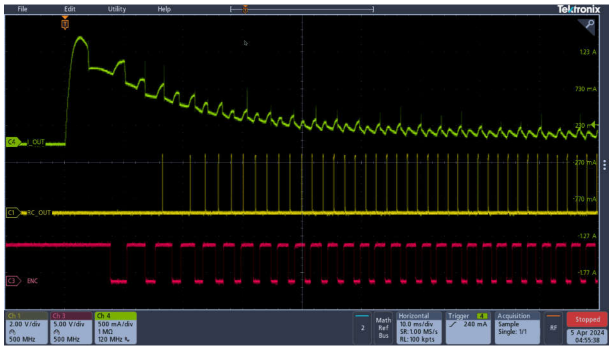


Figure 4-9. Inrush 4

4.4 High Load Conditions

During high load torque conditions, the ripple counting algorithm fails to detect a few current ripples. At these operating conditions, the motor speed is usually too slow for the algorithm to estimate accurately. Furthermore, at such low speeds, any error in the motor resistance value compounds multifold to the actual speed estimation by the algorithm. The workaround here is to turn off the error corrector (if already on) by setting DIS_EC to 1b. The recommendation is to program the MCU to detect motor current with the IPROPI pin or the IMTR register. Switch off the error corrector once the motor current value crosses 50% of full load (stall) current. Turn the error corrector back on when the motor current value is below 50%.

Accuracy calculation is shown in Table 4-2.

Table 4-2. Accuracy at High Loads

Parameter	Error Corrector ON			Error Corrector OFF		
	50% Load (650mA)	60% Load (780mA)	70% Load (910mA)	50% Load (650mA)	60% Load (780mA)	70% Load (910mA)
Encoder Counts	40	32	20	40	32	20
RC_OUT Counts	60	36	15	60	48	30
Accuracy	100%	75%	50%	100%	100%	100%

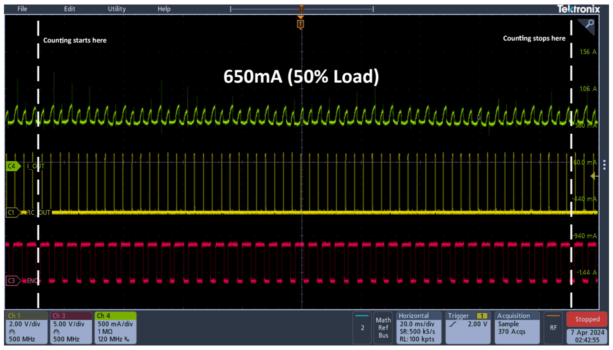


Figure 4-10. Error Corrector ON (50% Load)

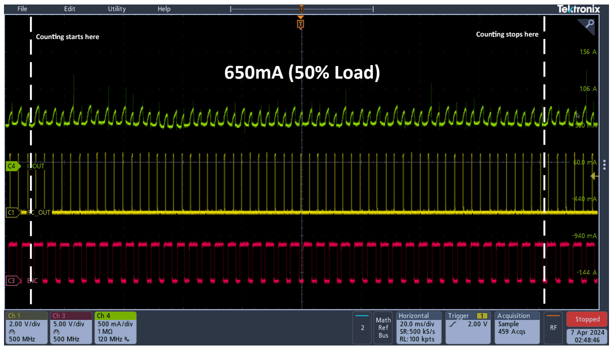


Figure 4-11. Error Corrector OFF (50% Load)

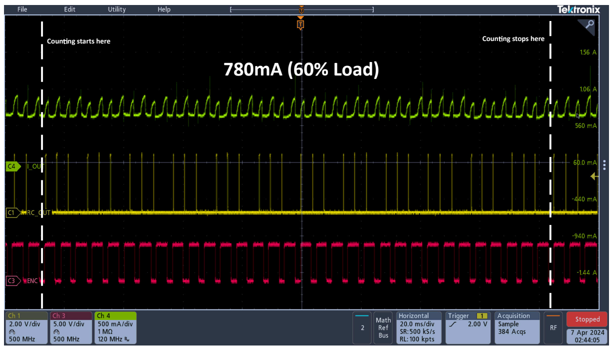


Figure 4-12. Error Corrector ON (60% Load)

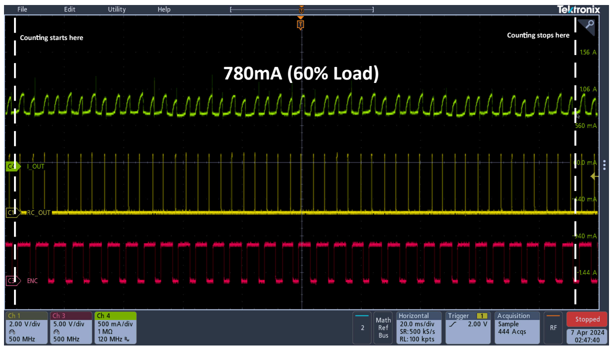


Figure 4-13. Error Corrector OFF (60% Load)

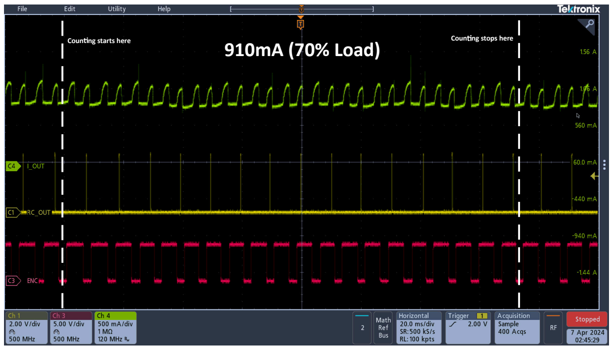


Figure 4-14. Error Corrector ON (70% Load)

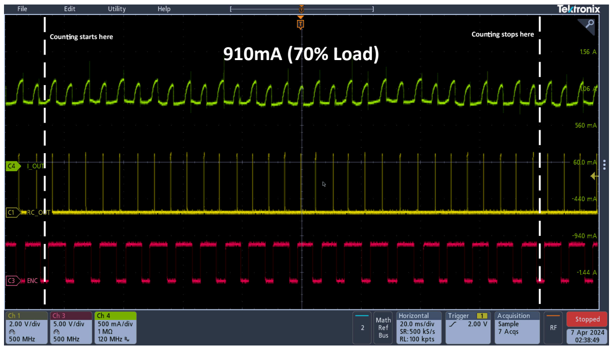


Figure 4-15. Error Corrector OFF (70% Load)

5 Summary

Sensorless position and speed control saves the user cost and design complexity. This algorithm reliably detects position and speed for various applications across wide voltage ranges. The tuning example in this document highlights the benefits of accurately tuned parameters. While challenges exist, particularly at low speeds, so do possible workarounds. The robustness and ease of use of the algorithm make DRV8214 and DRV8234 ideal options to replace costly position and speed sensors.

6 References

- Texas Instruments DRV821x DRV823x EVM GUI
- Texas Instruments *DRV8214 2-A Brushed DC Motor Driver with Ripple Counting, Stall Detection, and Speed Regulation*, data sheet.
- Texas Instruments *DRV8234 2-A Brushed DC Motor Driver with Ripple Counting, Stall Detection, and Speed Regulation*, data sheet.
- Texas Instruments, *Automotive Brushed-Motor Ripple Counter Reference Design for Sensorless Position Measurement*, reference guide.

7 Revision History

Changes from Revision * (April 2024) to Revision A (May 2024) **Page**

- Updated the numbering format for tables, figures, and cross-references throughout the document 1
- Deleted *The motor part number is N216053E* 4