Application Report
SLVAE40–November 2018

Design and Implementation of Stepper Motor Control Over I²C Communication

Vashist Bist, Analog Motor Drives
Manu Balakrishnan, Industrial Systems

ABSTRACT

Many stepper-motor applications, such as appliances, now demand increased protection, increased diagnostics and control flexibility at reduced system cost. The cost of the micro-controller is significant in the bill of material (BOM) of any system solution. With increased number of motors in the end-equipment, the number of micro-controller peripheral must increase, which increases overall system cost. The cost of this micro-controller can be reduced by decreasing the number of peripherals. This can be accomplished by communicating with the number of drivers together over a single communication line such as I²C communication. TI's next generation stepper motor driver (DRV8847S) are designed for controlling the stepper motor over I²C line with a flexible control interface options, detailed diagnostics of the faults and enhanced protection features. This article presents a detailed operation of the driver to attain a stepper motor control in full and half step mode. This article also covers the methodology of controlling single and multiple drivers over I²C line.

Contents

List of Figures

List of Tables

Trademarks

All trademarks are the property of their respective owners.

1 Introduction

The DRV8847 device is a dual H-bridge motor driver for driving two DC motors, a bipolar stepper motor, or other loads such as relays. The output stage of the driver consists of N-channel power MOSFETs configured as two full H-bridges to drive motor windings or four independent half bridges. The device support 4 different modes - the 4-pin interface, 2-pin interface, parallel bridge interface and independent bridge interface for supporting variety of loads.

The DRV8847 family consists for two variants of devices:

- Hardware Variant (DRV8847)
- I^2C Variant (DRV8847S)

The hardware variant device (DRV8847) has MODE and TRQ pins for setting the device operating mode and the motor torque/current regulation limit scalar setting. Whereas, the I^2C variant device (DRV8847S) replaces the MODE and TRQ pins with SDA and SCL pins for I^2C communication. The I^2C block-set in DRV8847S can be used to control the device with additional detailed diagnostics over I^2C line. DRV8847S is capable of controlling the stepper motor (or other loads such as brushed DC motor or solenoids) by controlling the bits of the I^2C registers which completely eliminates the requirement of four GPIOs for the input signals (IN1, IN2, IN3 and IN4) to motor driver. In addition, the DRV8847S gives more flexibility in configuration and diagnostics of motor and driver via multiple status registers.

Depending on the application requirement, the stepper motor can be controlled in full-step mode or half-step mode. The selection between the full step and half step mode depends on the step size requirement, speed and position-control accuracy. The control modes available in DRV8847 enable the selection full-step or half-step control with 2-pin or 4-pin or independent mode. The full step or half step stepper control can also be achieved through the sequence of I^2C register write commands and hence reduces the GPIO requirement from the MCU.

The multi-slave I^2C communication feature of DRV8847S enable the control multiple DRV8847S devices via a single set of I^2C lines. In effect, only two GPIOs of the MCU are enough to control multiple DRV8847S drivers and hence multiple motors and other loads. This helps to reduce the GPIO and peripheral requirement from MCU and reduces the MCU cost.

2 Stepper Motor Operation

DRV8847 can support three stepping modes for example, Full-Step Mode, Half-Step Mode (with Hi-Z Off State) and Half-Step Mode (with Brake Off State) as explained below.

2.1 Full Step Mode

In full-step mode, current in both of the full-bridges flows either in forward direction or reverse direction, with a phase shift of 90º between the two windings. The full-step operation of stepper motor can be achieved by any of the 2-pin, 4-pin or independent modes.

Table 1 shows the driver sequence to achieve the full-step mode.

Table 1. Full Step Operation Sequence in Different Modes

Sequence	Bridge Operation		2-Pin Mode		4-Pin Mode				Independent Mode			
	FB1	FB2	IN1	IN2	IN1	IN2	IN3	IN4	IN1	IN2	IN3	IN4
S1	FRW	REV	1	0	1	0	0	1	1	0	0	1
S2	FRW	FRW	1	1	1	0	1	0	1	0	1	0
S3	REV	FRW	0	1	0	1	1	0	0	1	1	0
S4	REV	REV	0	0	0	1	0	1	0	1	0	1

Figure 1, Figure 2, Figure 3 and Figure 4 shows the S1, S2, S3 and S4 sequence of stepper motor being driven in Full-Step mode.

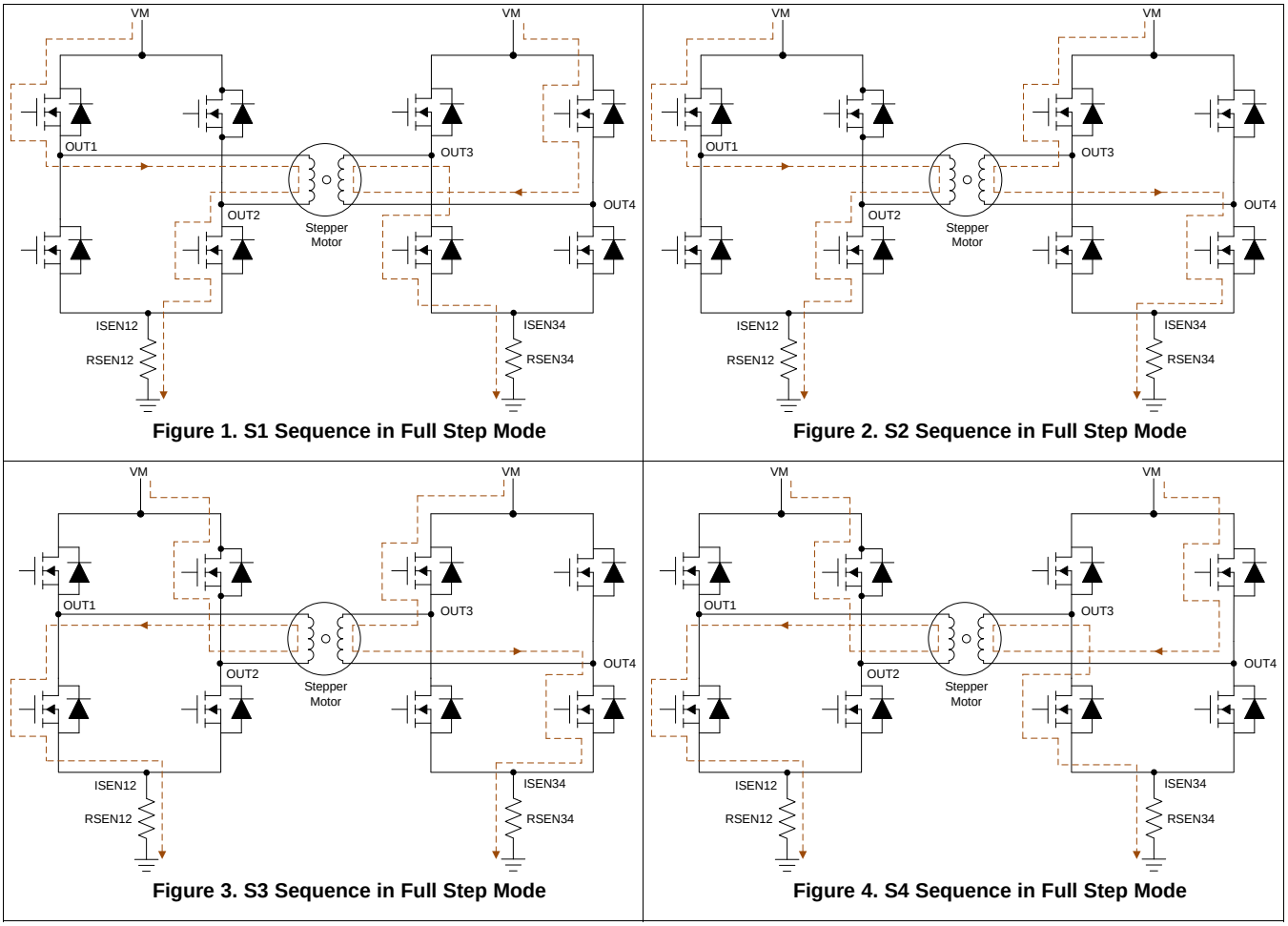

Figure 1. S1 Sequence in Full Step Mode

Figure 2. S2 Sequence in Full Step Mode

Figure 3. S3 Sequence in Full Step Mode

Figure 4. S4 Sequence in Full Step Mode

Figure 5 and Figure 6 shows the waveforms and circular chart of stepper motor operation in Full-Step mode.

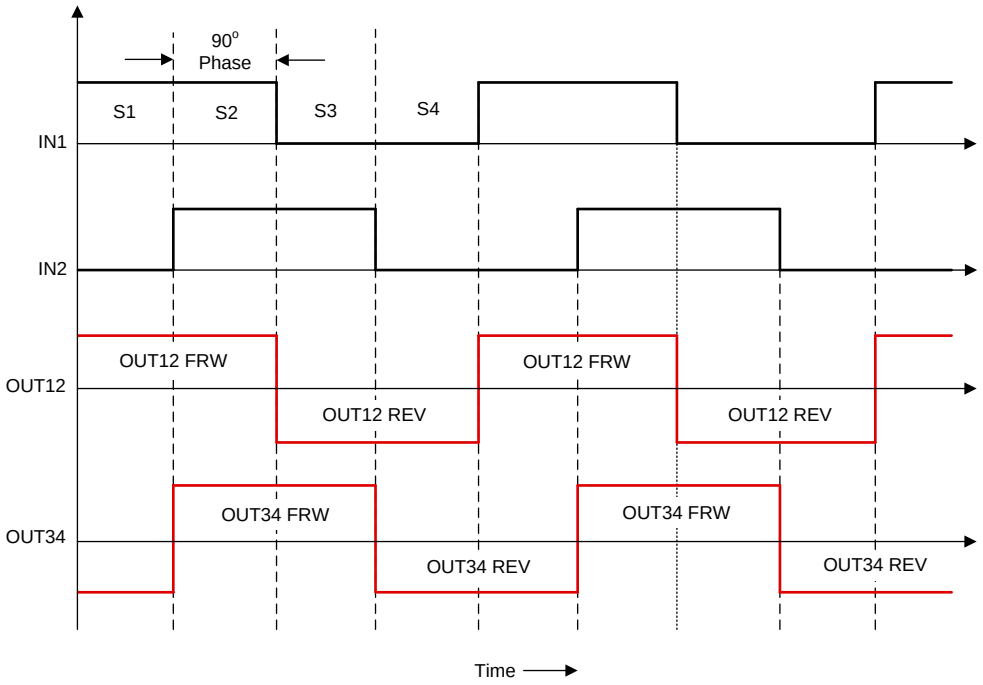

Figure 5. Waveforms for Stepper Motor Operation in Full Step Mode

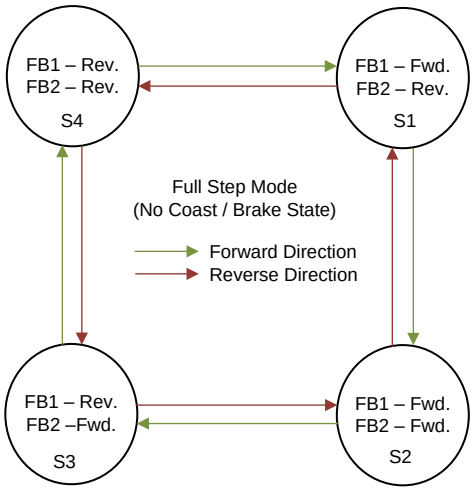

Figure 6. Sequence Chart for Stepper Motor Operation in Full Step Mode

2.2 Half Step Mode (with Hi-Z Off-state)

In half-stepping mode, the full-bridge operates in one of the three modes (forward, reverse, or coast/brake mode) with a phase shift of 45° between the two windings.

In coast/brake mode, the motor phase current goes to zero. The current in a winding can be set to zero by turning off the full bridge (Hi-Z off state), resulting in fast current decay method or "coasting". The other way to bring the winding current to zero is by turning on both the low sides FETs of the full bridge, resulting in slow current decay or "braking".

Table 2 shows the driver sequence to achieve the half-step mode (with Hi-Z off state). As shown in this table, the state S1 and S5 for full-bridge-1 (FB1) and state S3 and S7 for full-bridge-2 (FB2) are the driver coast state which is achieved when the driver is in Hi-Z mode.

Table 2. Half Step (with Hi-Z Off-State) Operation Sequence in Different Modes

Sequence	Bridge Operation		2-Pin Mode		4-Pin Mode				Independent Mode			
	FB1	FB2	IN1	IN2	IN1	IN2	IN3	IN4	IN1	IN2	IN3	IN4
S1	COAST	REV	Not Applicable		0	0	0	1	Not Applicable (No Hi-Z State Possible)			
S2	FRW	REV			1	0	0	1				
S3	FRW	COAST			1	0	0	0				
S4	FRW	FRW			1	0	1	0				
S5	COAST	FRW			0	0	1	0				
S6	REV	FRW			0	1	1	0				
S7	REV	COAST			0	1	0	0				
S8	REV	REV			0	1	0	1				

Figure 7, Figure 8, Figure 9 and Figure 10 shows the S1, S2, S3 and S4 sequence of stepper motor being driven in Half-Step mode (with Hi-Z off state operation).

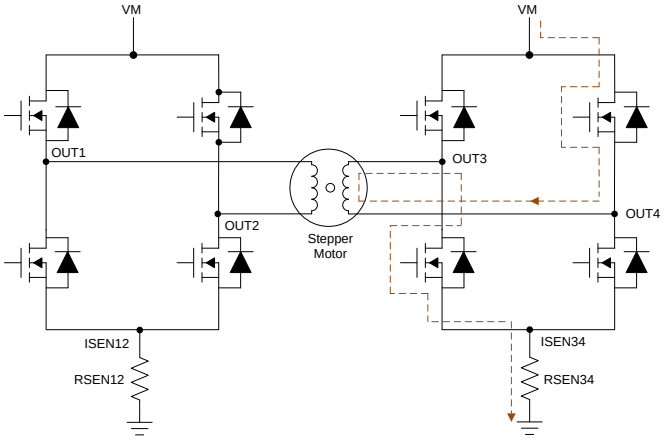

Figure 7. S1 Sequence in Half Step Mode (Hi-Z Off State)

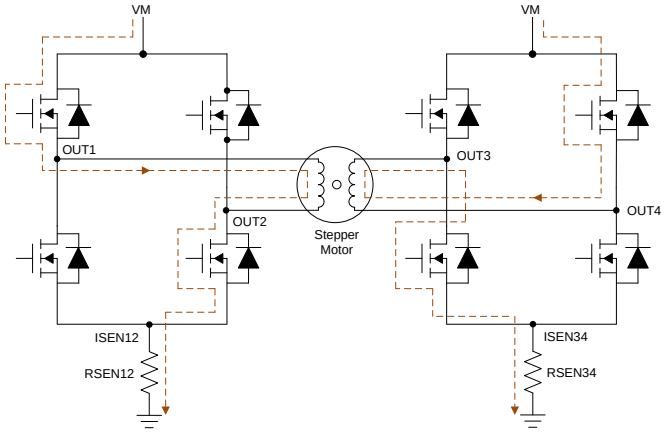

Figure 8. S2 Sequence in Half Step Mode (Hi-Z Off State)

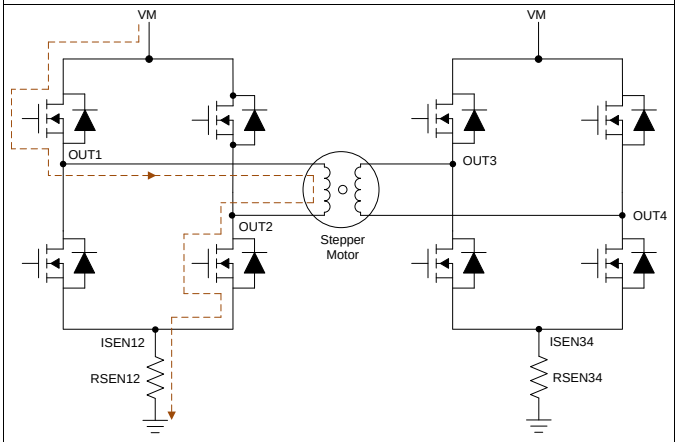

Figure 9. S3 Sequence in Half Step Mode (Hi-Z Off State)

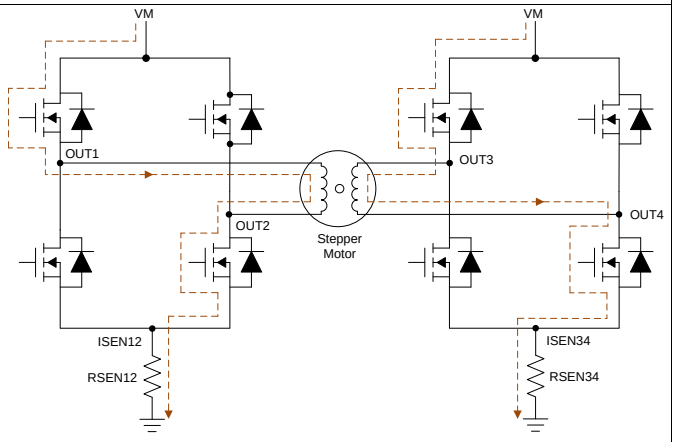

Figure 10. S4 Sequence in Half Step Mode (Hi-Z Off State)

Figure 11, Figure 12, Figure 13 and Figure 14 shows the S5, S6, S7 and S8 sequence of stepper motor being driven in Half-Step mode (with Hi-Z off state operation).

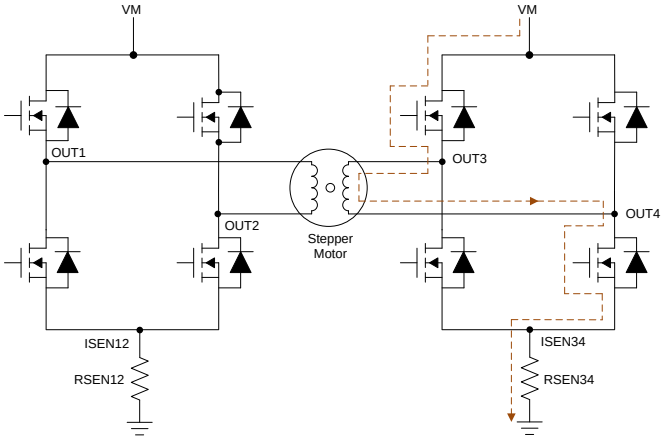

Figure 11. S5 Sequence in Half Step Mode (Hi-Z Off State)

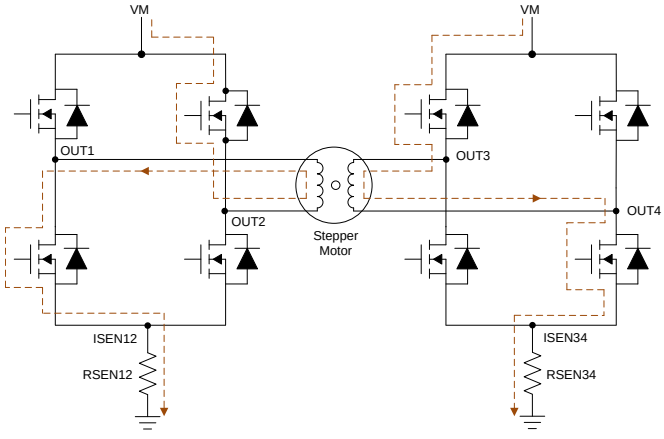

Figure 12. S6 Sequence in Half Step Mode (Hi-Z Off State)

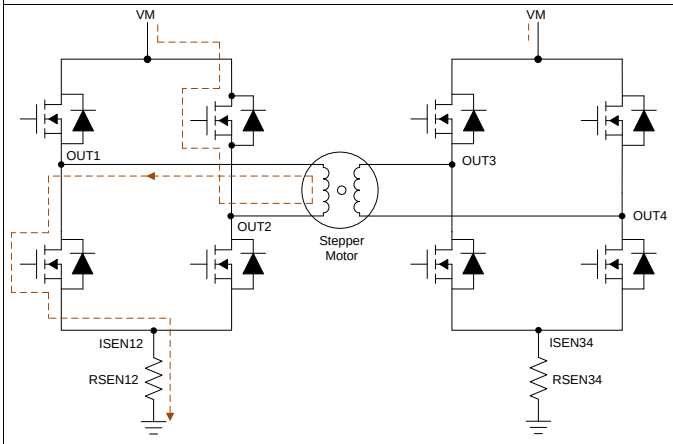

Figure 13. S7 Sequence in Half Step Mode (Hi-Z Off State)

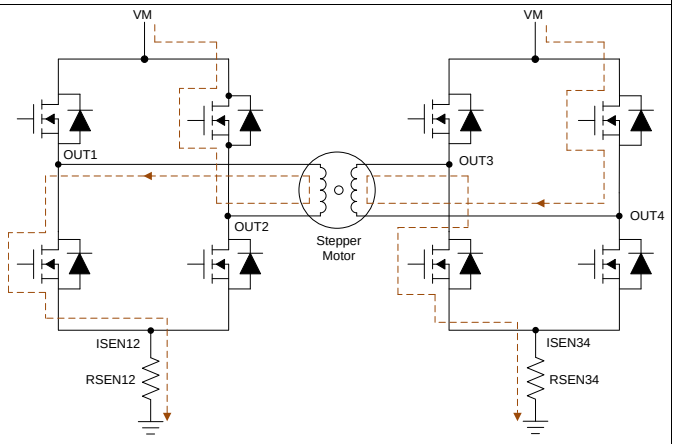

Figure 14. S8 Sequence in Half Step Mode (Hi-Z Off State)

Figure 15 and Figure 16 shows the waveforms and circular chart of stepper motor operation in Half-Step mode (with Hi-Z off state operation). Figure 15 shows a phase difference of 45 degrees between the input control signals. The voltage/current on the phase windings (OUT12 and OUT34) is still 90° out of phase.

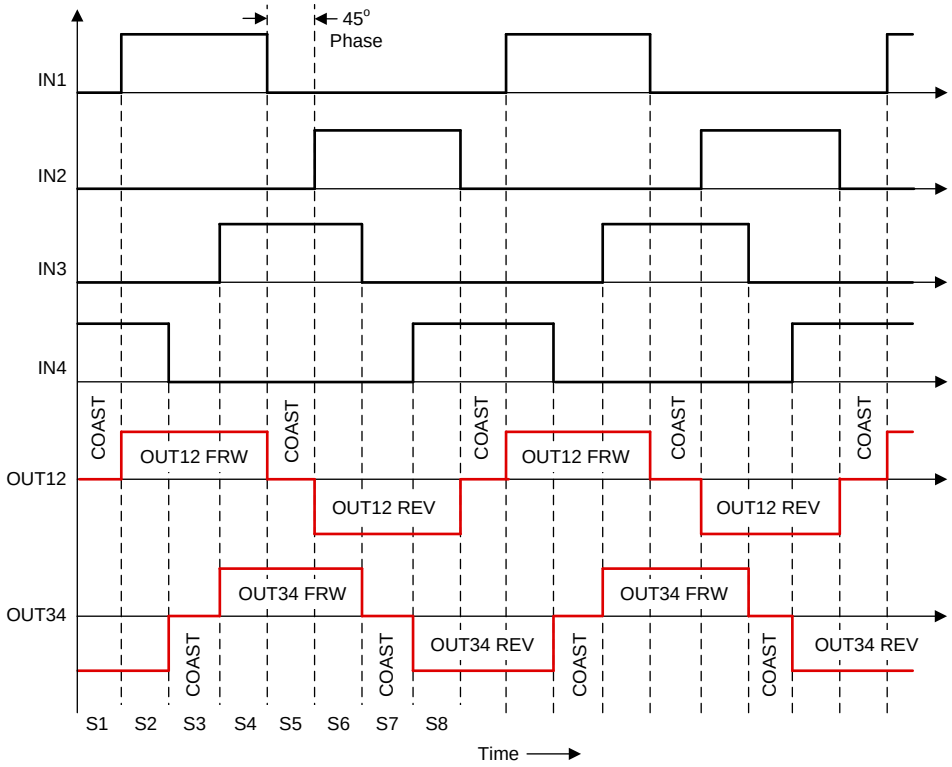

Figure 15. Waveforms for Stepper Motor Operation in Half Step Mode (with Hi-Z Off State)

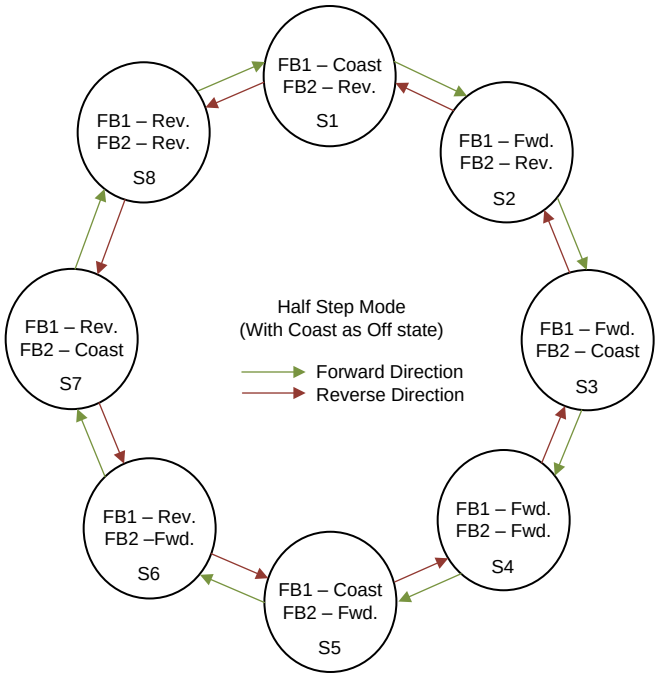

Figure 16. Sequence Chart for Stepper Motor Operation in Half Step Mode (with Hi-Z Off State)

2.3 Half Step Mode (with Brake Off-State)

Table 3 shows the driver sequence to achieve the half-step mode (with Brake off state). As shown in this table, the state S1 and S5 for full-bridge-1 (FB1) and state S3 and S7 for full-bridge-2 (FB2) are the driver brake state which is achieved when both low side FET's for full-bridge are turned on.

Table 3. Half Step (with Brake Off-State) Operation Sequence in Different Modes

Sequence	Bridge Operation		2-Pin Mode		4-Pin Mode				Independent Mode			
	FB1	FB2	IN1	IN2	IN1	IN2	IN3	IN4	IN1	IN2	IN3	IN4
S1	BRAKE	REV	Not Applicable		1	1	0	1	1	1	0	1
S2	FRW	REV			1	0	0	1	1	0	0	1
S3	FRW	BRAKE			1	0	1	1	1	0	1	1
S4	FRW	FRW			1	0	1	0	1	0	1	0
S5	BRAKE	FRW			1	1	1	0	1	1	1	0
S6	REV	FRW			0	1	1	0	0	1	1	0
S7	REV	BRAKE			0	1	1	1	0	1	1	1
S8	REV	REV			0	1	0	1	0	1	0	1

Figure 17, Figure 18, Figure 19 and Figure 20 shows the S1, S2, S3 and S4 sequence of stepper motor being driven in Half-Step mode (with Brake off state operation).

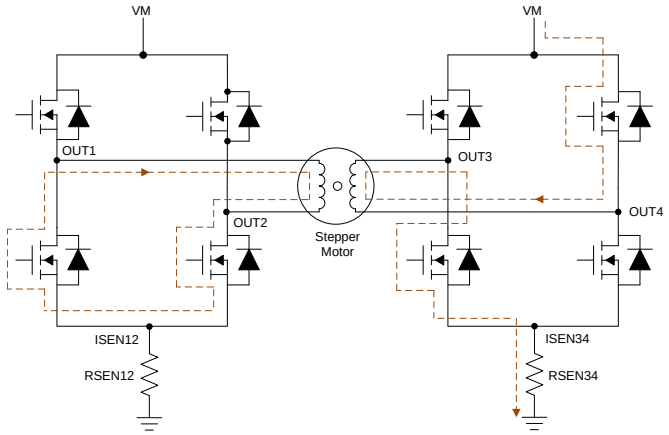

Figure 17. S1 Sequence in Half Step Mode (Brake Off State)

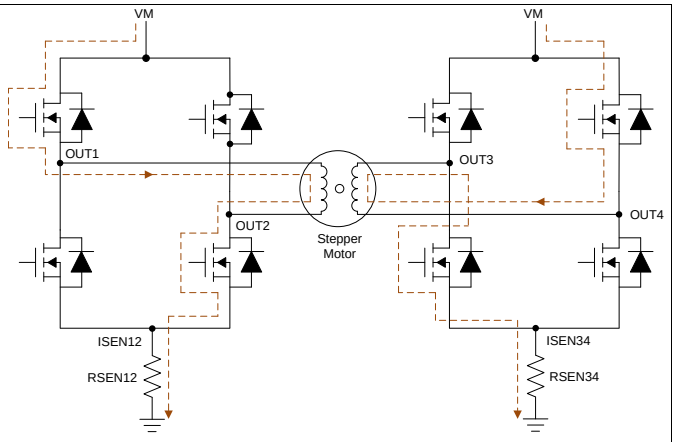

Figure 18. S2 Sequence in Half Step Mode (Brake Off State)

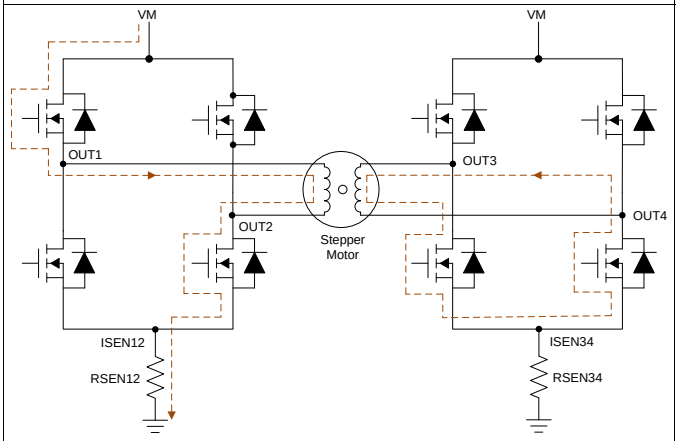

Figure 19. S3 Sequence in Half Step Mode (Brake Off State)

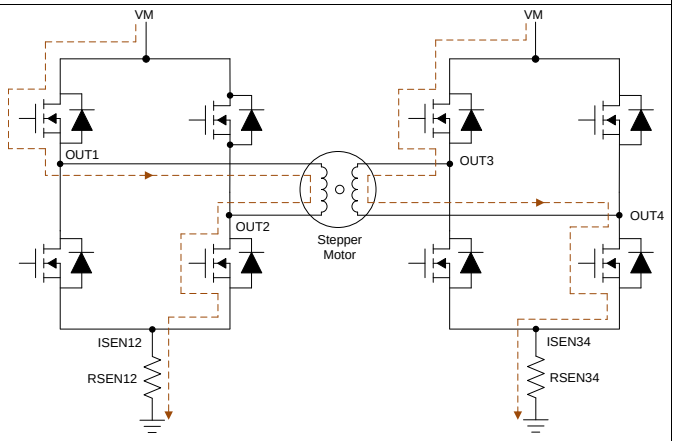

Figure 20. S4 Sequence in Half Step Mode (Brake Off State)

Figure 21, Figure 22, Figure 23 and Figure 24 shows the S5, S6, S7 and S8 sequence of stepper motor being driven in Half-Step mode (with Brake off state operation).

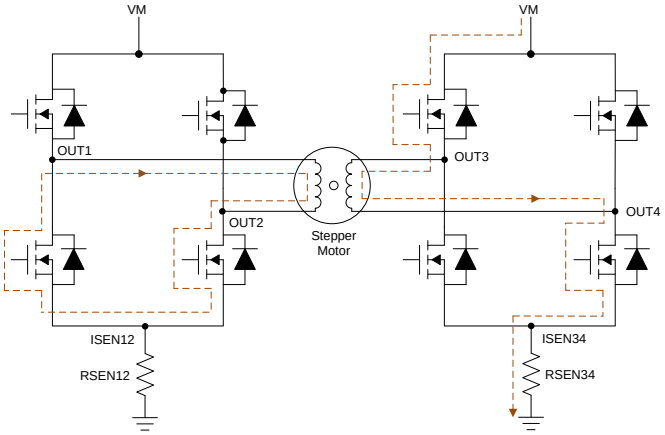

Figure 21. S5 Sequence in Half Step Mode (Brake Off State)

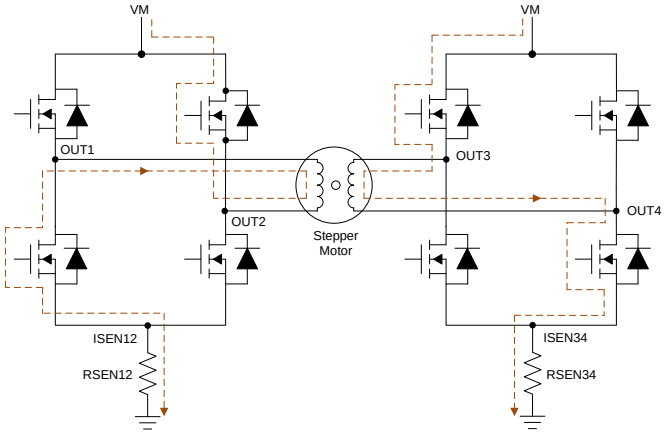

Figure 22. S6 Sequence in Half Step Mode (Brake Off State)

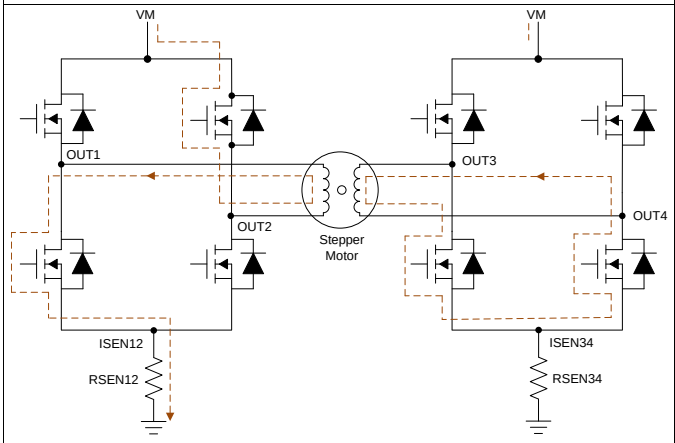

Figure 23. S7 Sequence in Half Step Mode (Brake Off State)

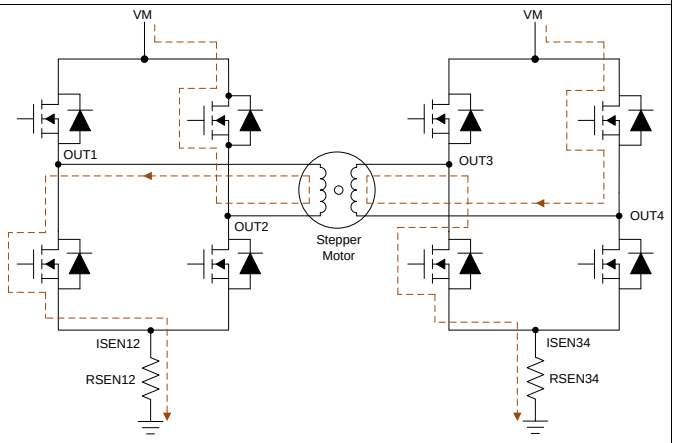

Figure 24. S8 Sequence in Half Step Mode (Brake Off State)

Figure 25 and Figure 26 shows the waveforms and circular chart of stepper motor operation in Half-Step mode (with Brake off state operation).

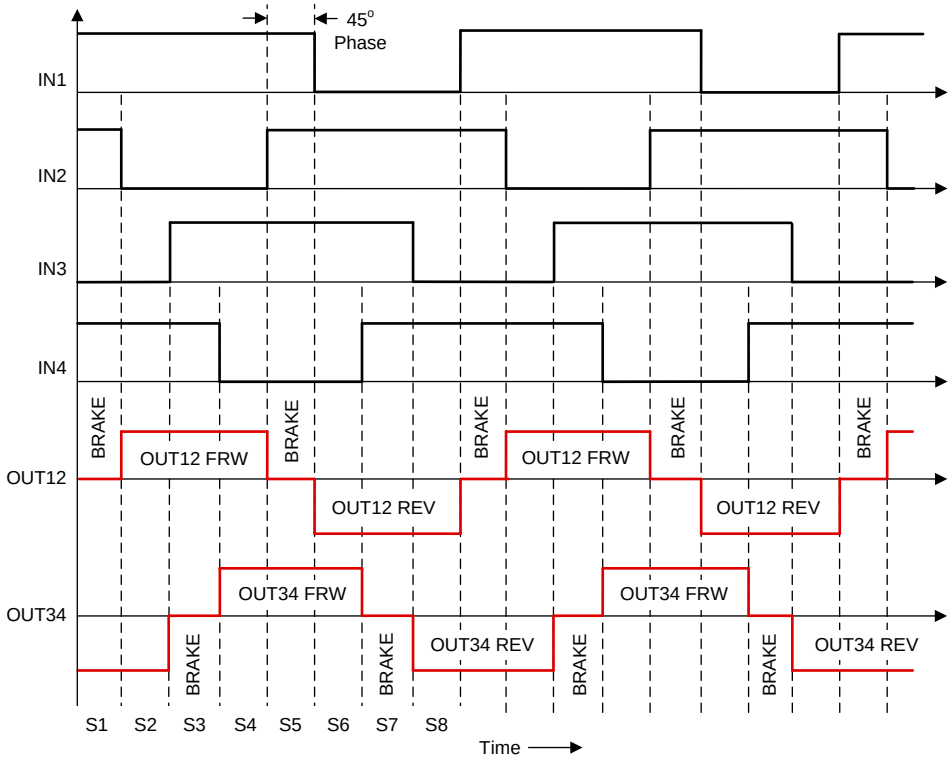

Figure 25. Waveforms for Stepper Motor Operation in Half Step Mode (with Brake Off State)

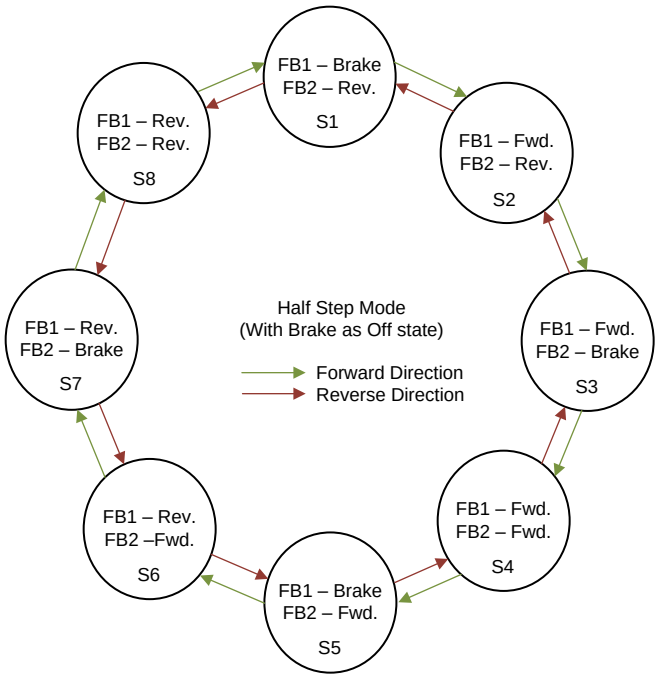

Figure 26. Waveforms for Stepper Motor Operation in Half Step Mode (with Brake Off State)

3 Stepper Motor Control Over I²C Communication

As discussed previously, the DRV8847S can communicate to MCU through I²C lines to drive motors or other loads and hence reducing the usage of number of MCU pins and peripherals. A single MCU can act as a master to control single DRV8847S (single slave operation) or multiple DRV8847S (multi-slave operation) devices. The real benefit of MCU pin reduction can be realized in the multi-slave operation.

3.1 Single Stepper Motor Control

Stepper motor control of DRV8847S through I²C lines is achieved by repeatedly writing the IN1, IN2, IN3 and IN4 bits in the IC1_CON register. Figure 27 shows the flowchart for controlling the stepper motor in half-step mode using the I²C control mode. Following steps are used for achieving this operation.

- A sequence counter (Seq_Counter) is initialized which count the 8 number of steps in a complete period. Each step corresponds to 45° electrical period.
- A timer is initialized for controlling the time period of each step (for example, speed)
- The first sequence of IN1, IN2, IN3 and IN4 bits are loaded onto I2C_CON register
- Start the timer and continue to perform other tasks
- Once the timer is completed (Timer Interrupt Service Routine (ISR)), increment the Seq_Counter and update the INx bits for the next step
- Check if sequence counter has reached 7 (for completing 8 steps of electrical cycle)
- Sequence counter is reset to '0' if the 8 steps are completed else the timer is started again

In I²C control, the MODE bits of the IC1 control register can be set for 4-pin interface. For bipolar stepper motor half stepping control, as per the above explanation, the following sequence steps setting can be written to IC1 control register one by one to control the OUTx pins:

- Sequence step 1: IN1 = 0, IN2 = 0, IN3 = 0, IN4 = 1
- Sequence step 2: IN1 = 1, IN2 = 0, IN3 = 0, IN4 = 1
- Sequence step 3: IN1 = 1, IN2 = 0, IN3 = 0, IN4 = 0
- Sequence step 4: IN1 = 1, IN2 = 0, IN3 = 1, IN4 = 0
- Sequence step 5: IN1 = 0, IN2 = 0, IN3 = 1, IN4 = 0
- Sequence step 6: IN1 = 0, IN2 = 1, IN3 = 1, IN4 = 0
- Sequence step 7: IN1 = 0, IN2 = 1, IN3 = 0, IN4 = 0
- Sequence step 8: IN1 = 0, IN2 = 1, IN3 = 0, IN4 = 1

With 100% torque scaling, the sequence of the IC1 control register vales are: {0x44, 0x4C, 0x0C, 0x2C, 0x24, 0x34, 0x14, 0x54}. Each step corresponds to 45 degree electrical, and the time period of each step determines the speed of the motor. A periodic time-based interrupt subroutine can be used to write the DRV8847S control register as per the required INx bit values.

For bipolar stepper motor full stepping control, only four switching sequences are used. The Seq_Counter is reset to zero once it reaches 3. the following sequence steps setting can be written to IC1 control register one by one to control the OUTx pins to achieve full stepping operation:

- Sequence step 2: IN1 = 1, IN2 = 0, IN3 = 0, IN4 = 1
- Sequence step 4: IN1 = 1, IN2 = 0, IN3 = 1, IN4 = 0
- Sequence step 6: IN1 = 0, IN2 = 1, IN3 = 1, IN4 = 0
- Sequence step 8: IN1 = 0, IN2 = 1, IN3 = 0, IN4 = 1

With 100% torque scaling, the sequence of the IC1 control register vales are: {0x4C, 0x2C, 0x34, 0x54}. Each step corresponds to 90 degree electrical, and the time period of each step determines the speed of the motor.

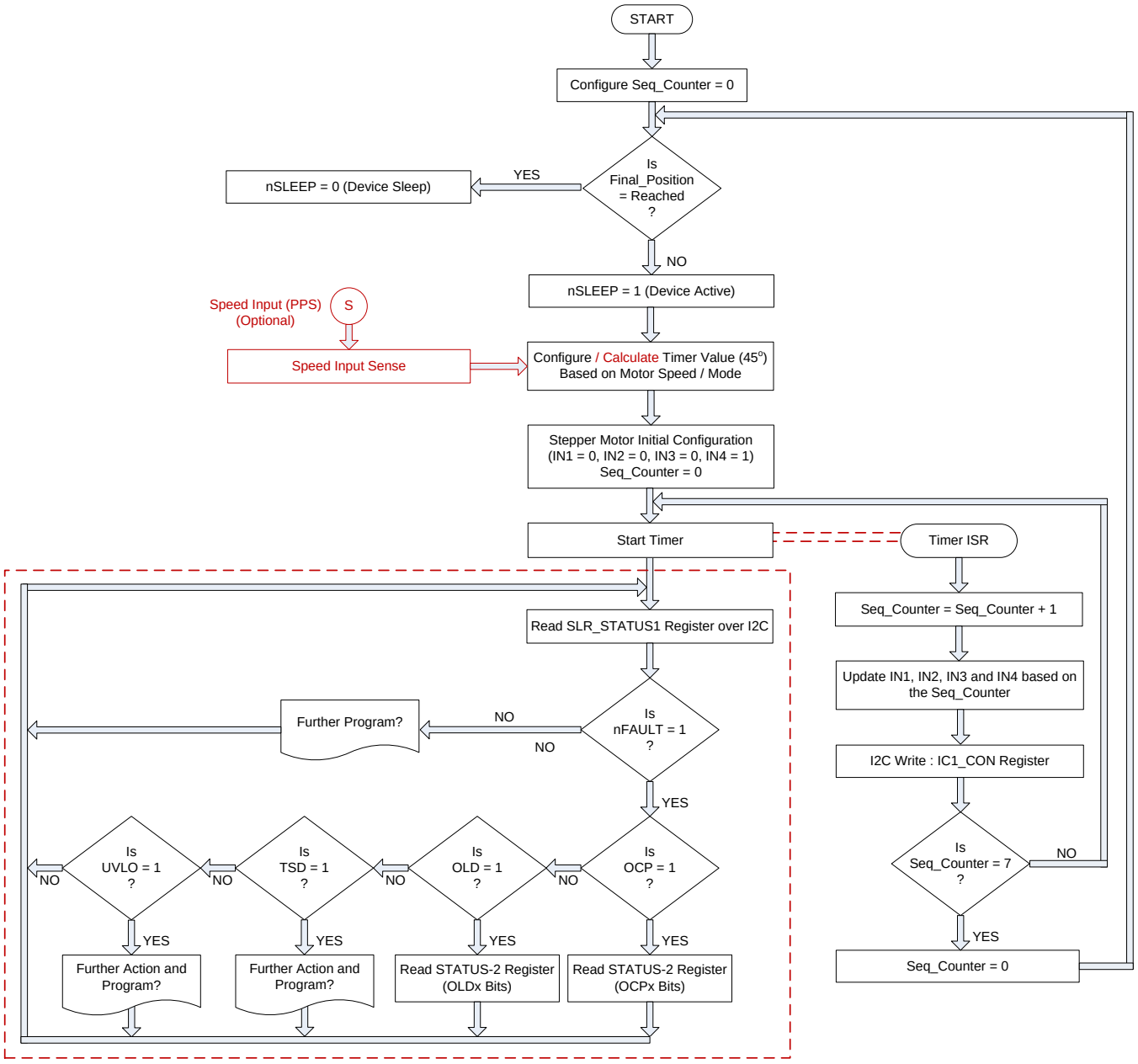

Figure 27. Flowchart of DRV8847S Controlling Single Stepper Motor Using I²C Bridge Control

3.2 Multiple Stepper Motors Control

Multiple stepper motors can be controlled over a single I²C line by using the multi-slave operation of DRV8847S. Figure 28 shows the connection of microcontroller to multiple DRV847S devices via a single I²C communication line. Following are the steps to configure the DRV8847S in multi-slave operation.

- The DRV8847S device default address is 0x60 (7- bit address). For achieving a multi-slave operation, this address has to be changed.
- The DRV8847S device releases the I²C bus as soon as the nFAULT pin is pulled-low externally (from micro-controller) as shown in Figure 29. The device has to disable the nFAULT pin before releasing the I²C buses by setting the DISFLT bit in IC2_CON register.
- To re-program the slave address of the device (1), the remaining devices nFAULT pin is pulled-low. Similarly the slave address of other connected DRV8847S devices are re-programmed.
- Once all devices address are reprogrammed, the nFAULT line is enabled again by clearing the DISFLT bit in IC2_CON register. This will enable the nFAULT output pin for fault flagging.
- Now all connected slave devices can be accessed using the newly reprogrammed address.
- The above steps should be repeated for any device in case of a power reset (nSLEEP).

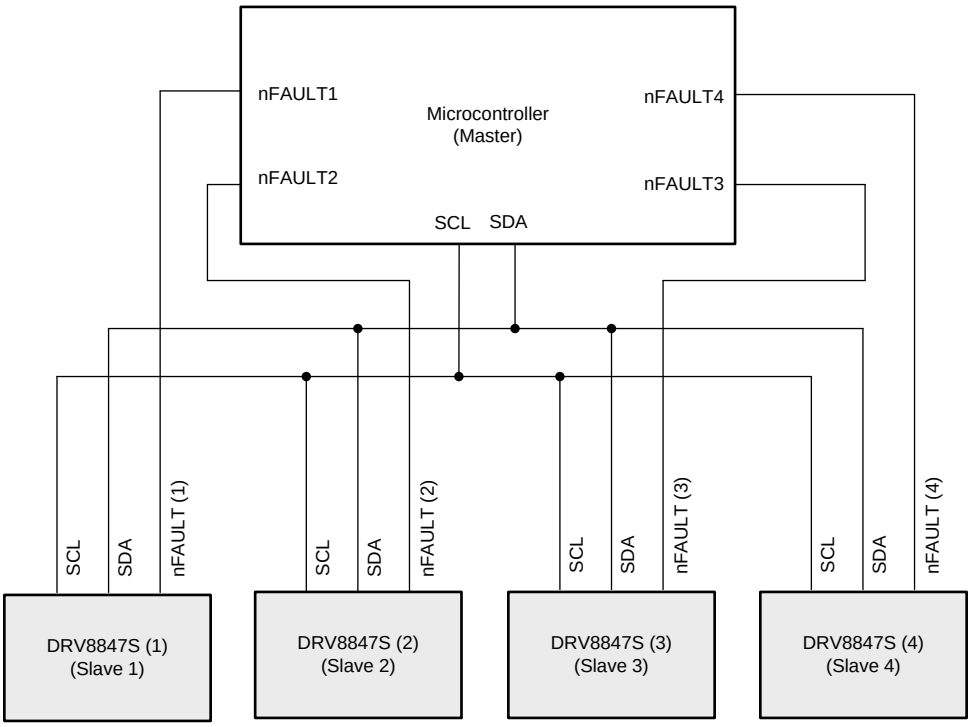

Figure 28. Multi-Slave Operation of DRV8847S

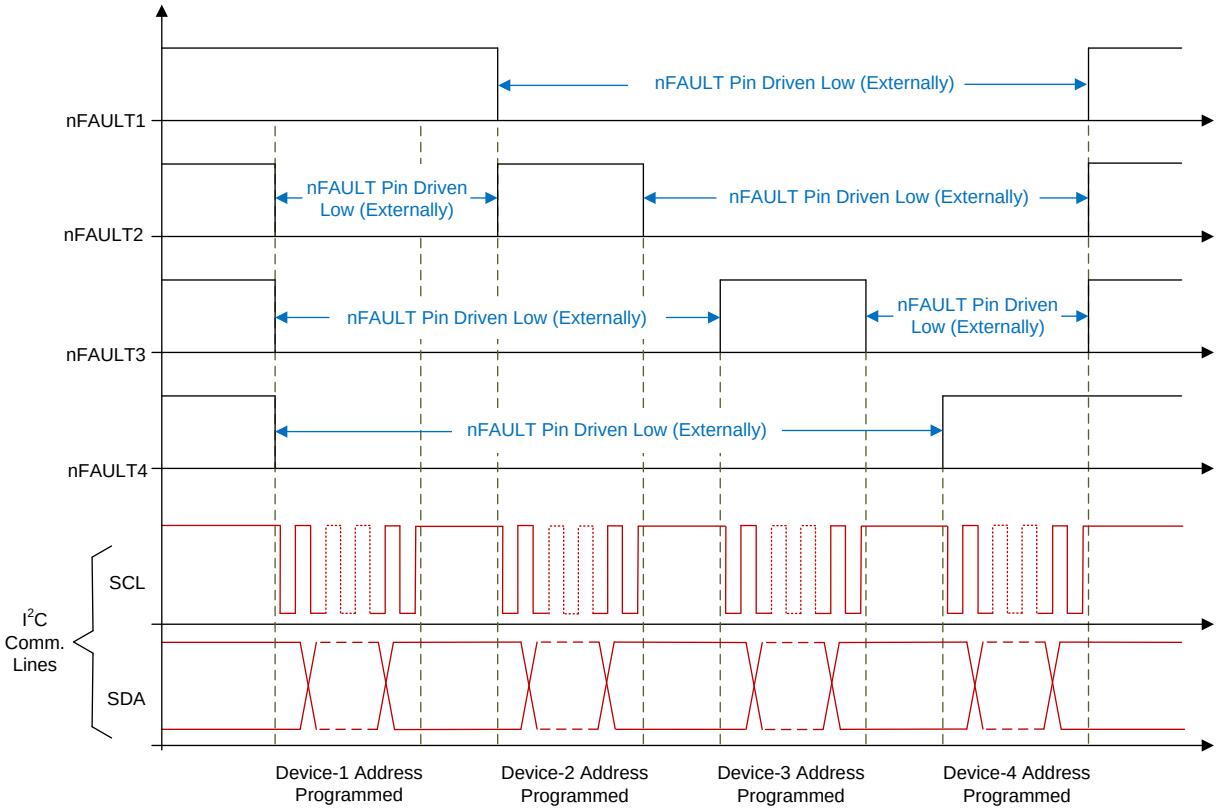

Figure 29. Multi-Slave Operation Waveforms

3.3 Stepper Motor Speed Control with I²C Control

When the DRV8847S is controlled through I²C line communication, the speed or frequency of the stepper motor depends on the rate at which the INx bits are updated in the IC1 control register. The maximum stepper rotation frequency achievable by I²C line control depends on following factors:

- I²C clock frequency
- Stepper control mode

3.3.1 Stepper Motor Speed Dependency on I²C Clock Frequency

In I²C based stepper motor control, each step change need an I²C write into the IC1 control register. So the maximum achievable stepper speed depends on the latency of one registers I²C write. The write cycle latency depends on I²C clock (SCL Frequency) and the I²C write sequence frame. The DRV8847 allows a maximum I²C clock frequency of 400-kHz in the fast mode.

Figure 30 shows one register write sequence frame. To write on the I²C bus, the master device sends a START condition on the bus with the address of the 7-bit slave device followed by a R/W bit, which is set to 0b to signifies a I²C write. After the slave sends the acknowledge bit (ACK), the master device then sends the 8-bit register address of the register to be written. This is followed by an acknowledge (ACK) signal again which notifies that the slave device is ready. Thereafter, the master device sends 8-bit data and terminates the transmission with a STOP condition. This sequence can be summarized as:

- START bit - MASTER (MCU) to SLAVE (DRV8847)
- Slave Device Address (7-bits) - MASTER (MCU) to SLAVE (DRV8847)
- Write Bit (R/W) - MASTER (MCU) to SLAVE (DRV8847)
- Acknowledge Bit (ACK) - SLAVE (DRV8847) to MASTER (MCU)
- Slave Register Address (8-bits) - MASTER (MCU) to SLAVE (DRV8847)
- Acknowledge Bit (ACK) - SLAVE (DRV8847) to MASTER (MCU)
- Slave Register Data (8-bits) - MASTER (MCU) to SLAVE (DRV8847)
- Acknowledge Bit (ACK) - SLAVE (DRV8847) to MASTER (MCU)
- STOP bit - MASTER (MCU) to SLAVE (DRV8847)

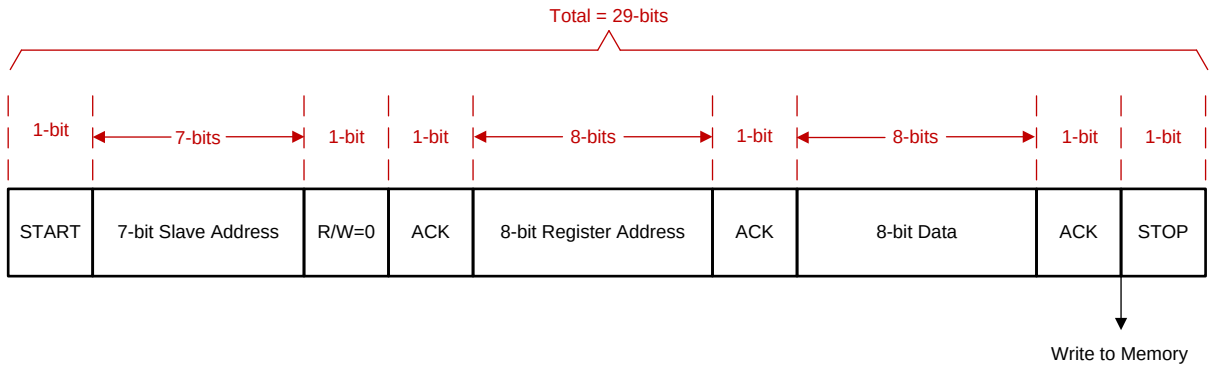

Figure 30. I²C Write Sequence

The write sequence frame length can be observed as 29-bits. With the allowed maximum I²C clock frequency of 400-kHz, the latency for one write cycle is 72.5-µs (= 29 / 400-kHz). This means the minimum step size duration possible with 400-kHz I²C clock is 72.5-µs.

Similarly if a 100-kHz clock is used, then the minimum motor control step duration possible is 290-µs (= 29 / 100-kHz).

3.3.2 Stepper Motor Speed Dependency on Stepper Control Mode

A bipolar stepper motor can be controlled in full stepping mode, half stepping mode and micro stepping mode. The DRV8847 allows the use of full stepping and half stepping modes as explained in Section 2. With half stepping mode, there are a total of eight steps in one complete electrical cycle and full stepping mode comprises of four steps in one complete electrical cycle.

3.3.2.1 Full-Stepping Mode

With full stepping mode and 400-kHz I²C clock, the theoretical minimum achievable motor electrical cycle period is calculated as 290-μs (= 4 x 72.5 μs). Therefore, the maximum achievable electrical rotation frequency with full stepping is 3.45-kHz (= 1/290 μs).

3.3.2.2 Half-Stepping Mode

With half-stepping mode and 400-kHz I²C clock, the theoretical minimum achievable motor electrical cycle period is calculated as 580-μs (= 8 x 72.5 μs). Therefore, the maximum achievable electrical rotation frequency with full stepping is 1.72-kHz (= 1/580 μs).

NOTE: This analysis doesn't include any latency in programming due to other instructions or functions. If an I²C read cycle for the fault signal is also incorporated after the write cycle, the maximum achievable speed reduces. The register read latency can be found in a similar way as explained in Section 3.3.1 by analyzing the read sequence frame. The designer has to calculate the latency and maximum achievable speed as per the clock used and other functions used in firmware.

4 Results and Discussion

The DRV8847S is tested to find the latency in an I²C write cycle with maximum I²C clock speed of 400-kHz. Figure 31 shows the SCL (400-kHz) and SDA signals with a single register write latency of 72.5-us.

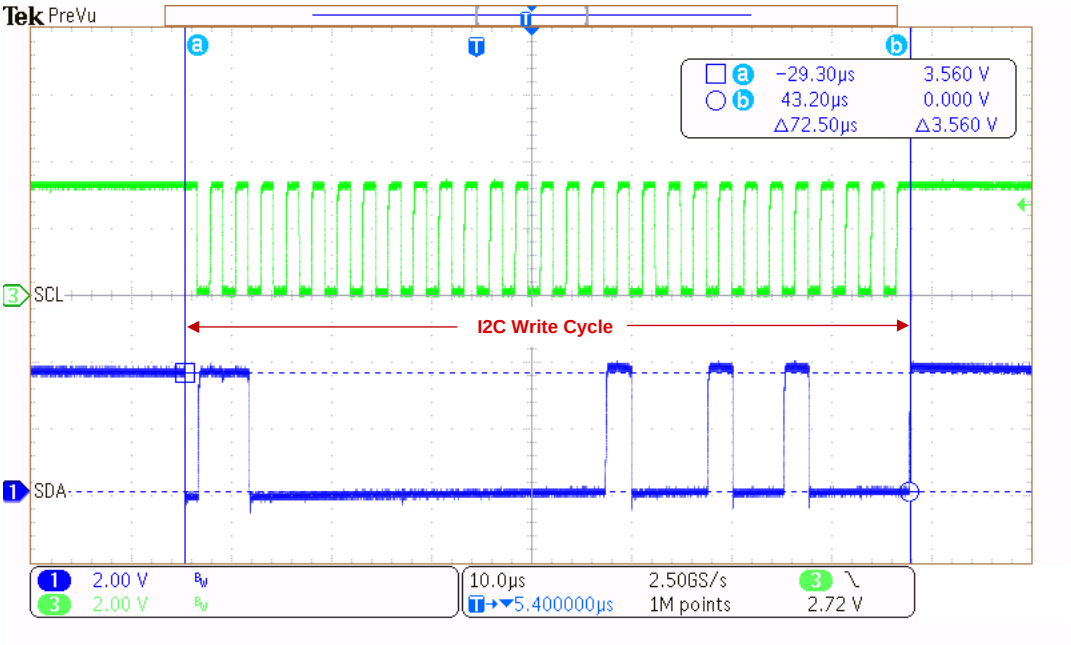

Figure 31. DRV8847S Single Register I²C Write Latency

Figure 32 shows the maximum frequency operation of the stepper motor in full stepping mode with I²C control using 400-kHz clock. The time period for one electrical cycle is 300-us, that is close to the theoretical minimum value of 290-us. Figure 32 shows the full-bridges output voltage (OUT12 and OUT34) with the I²C communication signals (SCL and SDA line).

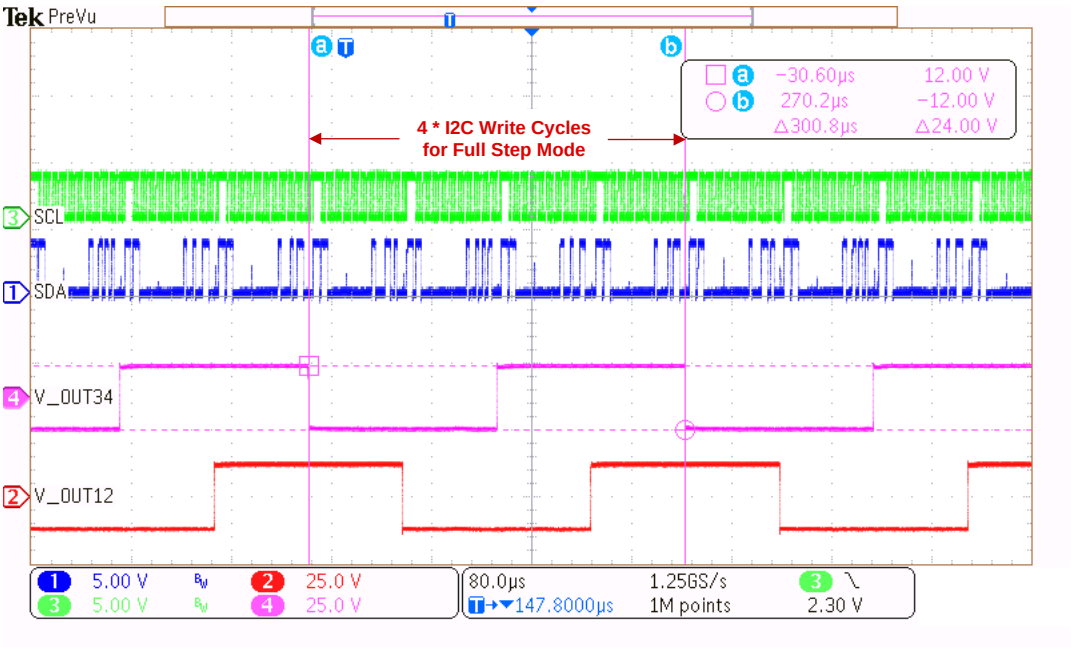

Figure 32. Maximum Frequency Full Stepping Operation in I²C Control Mode with 400-kHz I²C Clock

Figure 33 shows the maximum frequency operation of the stepper motor in half stepping mode with I²C control using 400-kHz clock. The time period for one electrical cycle is 602-us, that is close to the theoretical minimum value of 580-us. Figure 33 shows the full bridges output voltage (OUT12 and OUT34) with the I²C communication signals (SCL and SDA line).

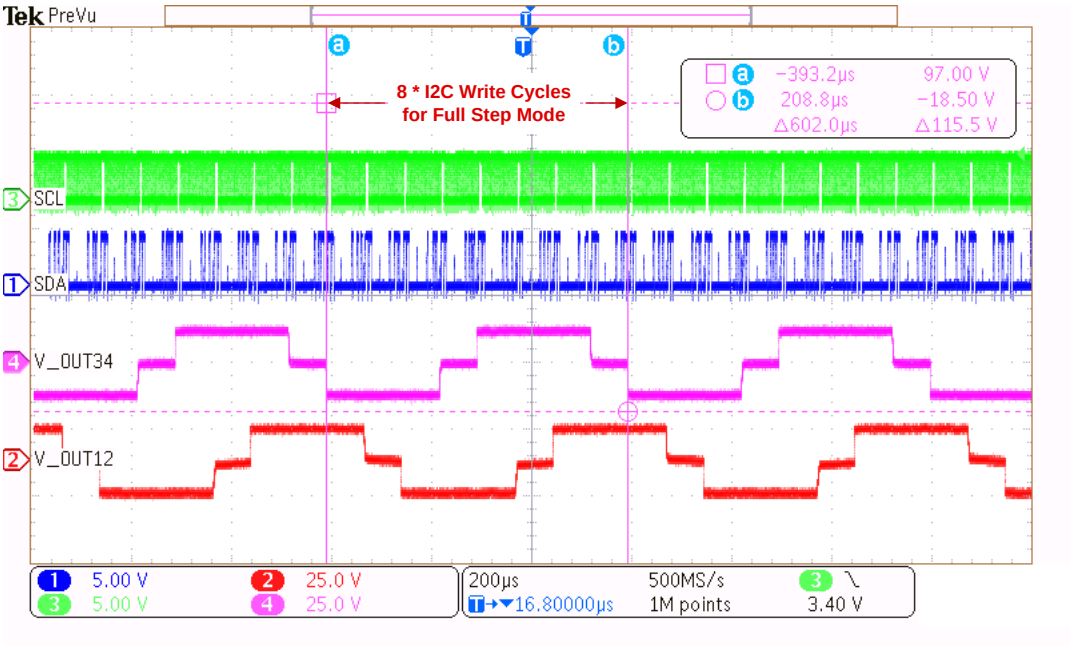

Figure 33. Maximum Frequency Half Stepping Operation in I²C Control Mode with 400-kHz I²C Clock

The I²C control mode of DRV8847S along with the multi-slave operation helps to considerably reduce the GPIO requirement from the micro-controller, still achieving stepper motor speed as high as 3.44 kHz (theoretically). The I²C control interface also brings detailed fault diagnosis, control optimization and system cost saving.

5 References

- Texas Instruments, DRV8847 Dual H-Bridge Motor Driver
- Texas Instruments, Small Motors in Large Appliances Application Report
- Texas Instruments, 12V, Highly Protected, Single Driver-Based Stepper, Brushed DC and Actuator Drive Reference Design

IMPORTANT NOTICE AND DISCLAIMER

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

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

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

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