Author: Ing. Alberto I. Leibovich & Ing. Pablo E. Leibovich
Date: January 12, 2017

Introduction

Stepper motors are among the most widely used actuators for high precision motion. They are used in routing systems, 3D printers, robots, etc.

For similar applications, servo motors were used in the past; but today, compared to stepper motors, they have higher costs and complexity because of the additional components needed to make them (position sensors and error amplifiers etc.). This is why today, the ideal choice is often a stepper motor because it's simpler to control it and it has good accuracy, good torque, moderate speed, and a low cost.

A stepper motor based application needs two more components:

- Controller: This is an IC capable of generating step pulses continuously for the driver to make the motor rotate.

- Driver: It is the interface between the controller's step signals and the necessary current for the motor windings. There are various types of drivers; a frequently used one is the H-bridge.

There are several ways to implement a stepper motor controller. One of them is by using a microcontroller. In such a scenario, the user has to program his own firmware to control the speed and direction. Generally, a dedicated microcontroller is used as the main controller because generating step pulses is a continuous task.

Another way to control the stepper motor is by using a dedicated standalone integrated circuit (IC). There are several different brands of commercial ICs that implement the step pulses based on an external clock and control signals. In such a case, the user has to configure the direction of motion and has to provide a clock signal.

For the most common applications, microcontroller based solutions are more expensive and complex because the user must also include all the external components such as crystal, resistors, transistors, diodes and decoupling capacitors to run the firmware. The commercial IC, instead, is designed to do the same task without the need to program anything.

In this application note, a SLG46531V will be configured to run as a commercial IC controller. The step pulses will be generated based on the external signals which set the direction of the motion. Also, the user will have the ability to enable or disable the motion externally and there will be a need to provide an external clock to define the speed motion.

Stepper Motor Description

A stepper motor is called so because the rotor moves with discrete steps, instead of rotating continuously like a conventional motor. There are two basic winding arrangements for the motor coils in a two phase stepper motor (the traditional ones): Unipolar and bipolar.

A unipolar stepper motor has singular winding with a center tap per phase. On each section of the winding, a unidirectional flow current can be used to determine the direction of the magnetic field. In figure 1, a schematic can be seen. To rotate, the user must apply sequential current with the correct direction to each half of the winding.

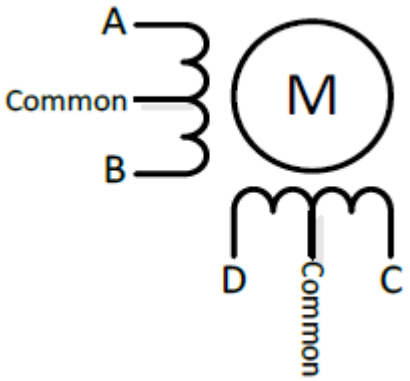


Figure 1. Unipolar Motor Schematic

A bipolar stepper motor has a single winding per phase. The current in the winding needs to be reversed in order to reverse a magnetic pole for rotating the rotor. In figure 2, another schematic can be seen. To rotate, the user must apply current to each winding sequentially, inverting the flow current direction in each step.

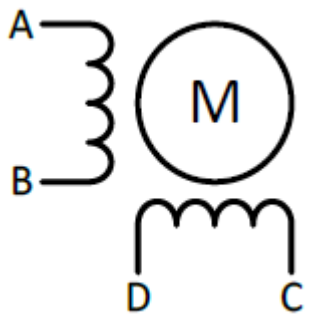


Figure 2. Bipolar Motor Schematic

In this application note, we will use a bipolar stepper motor, since the current flows across the entire coil, they have more torque. If a unipolar stepper motor is to be used, slight modifications will have to be done to the driver circuit; the controller, however, can be used without modifications.

There are three typical ways of controlling the winding current steps in a bipolar motor.

The **Full Step method** is the usual method for driving the motor. The two phases are always energized (with the corresponding current direction) ensuring that the rotor always aligns itself between the two pole positions. This way, the motor will provide its maximum torque.

The **Wave Drive method** energizes only one phase at a time. It has the same number of steps as the full step drive, but the motor will have significantly lesser torque. It is, thus, rarely used.

The **Half Step method** energizes one phase, then two, then one and so on.

This ensures that the motor moves with half step increments. With this method, the angular resolution gets incremented but the torque is considerably lesser when compared with that of the Full Step method.

One of the parameters of a stepper motor is the step angle per revolution. It indicates how many steps need to be done to make a revolution of the motor; this in turn defines how many times the sequence of the driving method has to be repeated for a given distance.

To interface the step pulses with the motor, an H-bridge is used to provide the current. There are several commercial H-Bridge integrated circuits or we can implement it using GreenPAK ICs.

This application note will implement the Full Step method and the Half Step method so the user may be able to select one of them with an input from the controller.

Logic Description and Schematic Diagram

The design implemented in this application note can be described with the schematic shown in figure 3.

The inputs and outputs of the system along with the simplified schematics of the H-Bridges can be seen in figure 3. The inputs and outputs are described in Table 1.

The H-bridge shown in figure 3 is a simplified schematic of the circuit. Each bridge controls the current of one phase of the stepper motor, so in this case two H-Bridges will be needed.

The implemented controller generates step pulses at the outputs A, B, C and D if the enable signal is set to high. In this case, signals OFF 1 and OFF 2 are both set to a high level and the sequence of steps is generated depending on the level of signal A/C. If A/C is low, the sequence for an anticlockwise motion is generated, but if it is high, the sequence for a clockwise motion is generated.

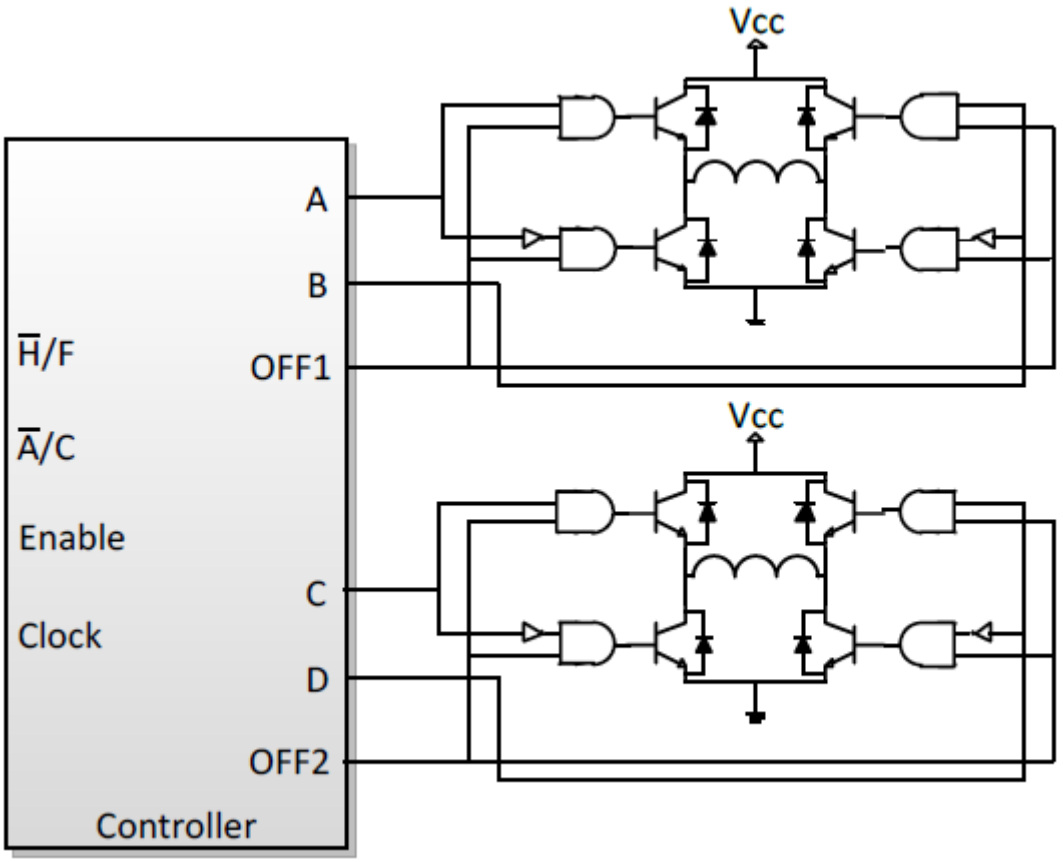


Figure 3. Application Schematic

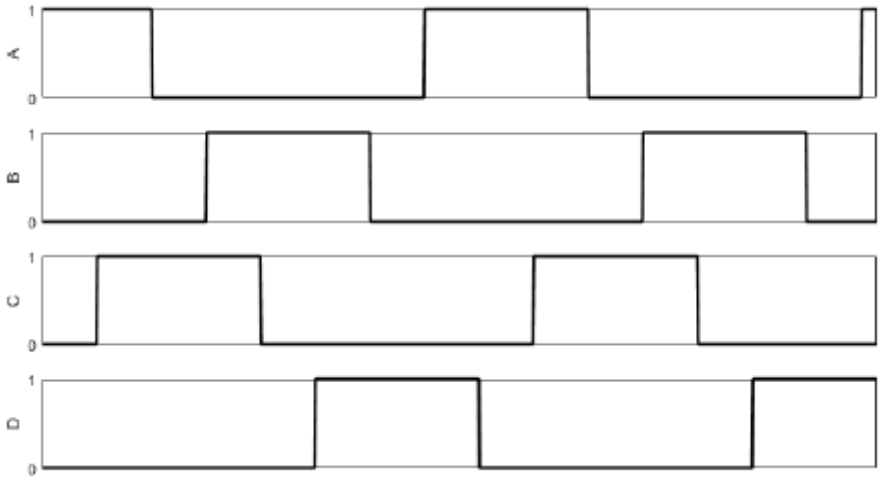


Figure 4. Clockwise step pulses sequence

SIGNAL	FUNCTION
H/F	Determines the control method. Low level indicates Half Step method and high level indicates Full Step method.
A/C	Determines the direction of the motion. Low level indicates anticlockwise rotating and high level indicates clockwise rotating.
Enable	A high level enables the controller, starting motion. A low level disables the controller, stopping motion.
A	Motor phase A step signal.
B	Motor phase B step signal.
C	Motor phase C step signal.
D	Motor phase D step signal.
OFF1	Controls one of the H-Bridges. When low, all of transistor outputs are turned off.
OFF2	Same as OFF 1 for the second H-Bridge.

Table 1. Inputs and outputs of the design

Also, the sequence depends on the selected method. If H/F is low, the half step sequence is generated and if H/F is high, the full step sequence is generated. For changing the direction or the method, it's recommendable to disable the motion first.

When the motion is disabled, the signals A, B, C, D, OFF 1 and OFF 2 are set to a low level.

In figure 4, the step pulses' sequence for the Half Step method is shown.

In figure 5, the step pulses' sequence for the Full Step method is shown.

To implement the sequence, a sequential signal generator with the ASM module of the GreenPAK will be used.

Implementation

As described earlier, a stepper motor motion is based on a sequence of steps. To implement it, the Asynchronous State Machine module of SLG46531V is used.

The ASM module is configured for the largest sequence (half step method), as it can be seen in figure 6. There are 8 states; one for every step.

The ASM is a linear sequencer which can go from State 0 to State 7 or from State 7 to State 0 for implementing clockwise or anticlockwise motion.

In figure 7, the ASM output configuration is shown.

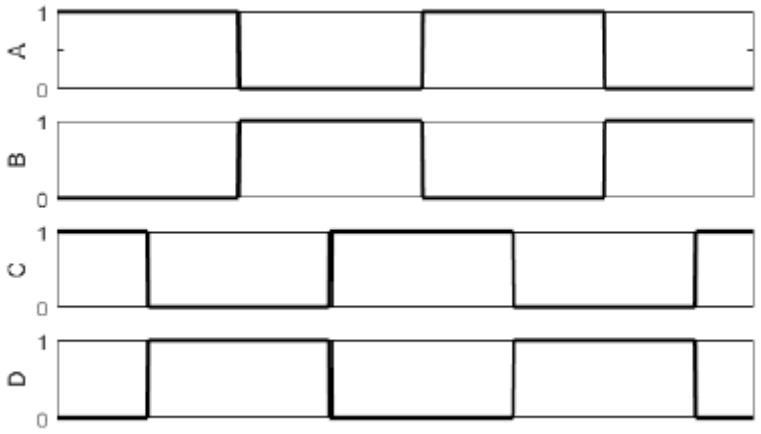


Figure 5. Clockwise step pulses sequence

A, B, C and D are configured with the corresponding values of the outputs of the system for the Half Step Method; so in the case of this method, the GreenPak outputs copy these values.

Ao, Bo, Co and Do are bit flags for the Full Step Method. Because in this method both coils are always energized, these bits indicate the output that has to be high in the steps, corresponding to the one energized coil of Half Step method.

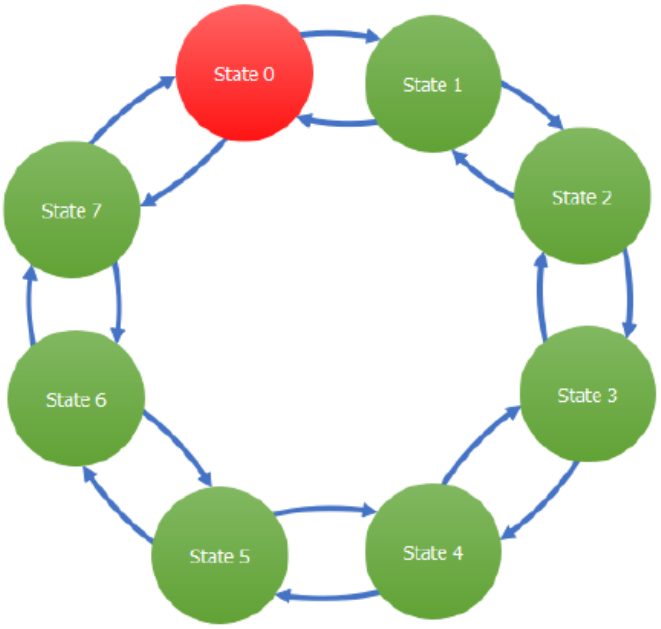


Figure 6. ASM states

State name	Connection Matrix Output RAM							
	Ao	Bo	Co	Do	A	B	C	D
State 0	0	0	1	0	1	0	0	0
State 1	0	0	0	0	1	0	1	0
State 2	0	1	0	0	0	0	1	0
State 3	0	0	0	0	0	1	1	0
State 4	0	0	0	1	0	1	0	0
State 5	0	0	0	0	0	1	0	1
State 6	1	0	0	0	0	0	0	1
State 7	0	0	0	0	1	0	0	1

Figure 7. ASM states

These flags are used later in the output logic. The clock control module can be seen in figure 8, with 4-bit LUT1 configuration in figure 9.

The transition between the ASM states is controlled by the 4-bit LUT1 output and the 2-bit LUTs.

To change from an even state to an odd state, the output of the 4-bit LUT1 must be low. To change from an odd state to an even state, the output of the 4-bit LUT1 must be high.

To implement this, in the case of Half Step method (H/F input in a low state), the clock input is connected to the DFF4, which is configured as a T-type FF. The 4-bit LUT1 copies the output of the FF (this is defined by the H/F connected to IN2).

In case of Full Step method (H/F input in a high state), the clock is directly copied by the 4-bit LUT1. With this logic, the ASM in the Half Step method changes its state on every rising edge of the clock (each rising edge is a DFF4 output transition from either low to high or from high to low).

In the Full Step Method, the ASM changes its state on both edges of the clock, so each motor's step (each clock cycle) corresponds to two states of the ASM. 2-bit LUT0, LUT1, LUT2 and LUT3 determines whether the transition is to be made to the next state or to the previous state, depending on the direction of the motion selected by the pin A/C.

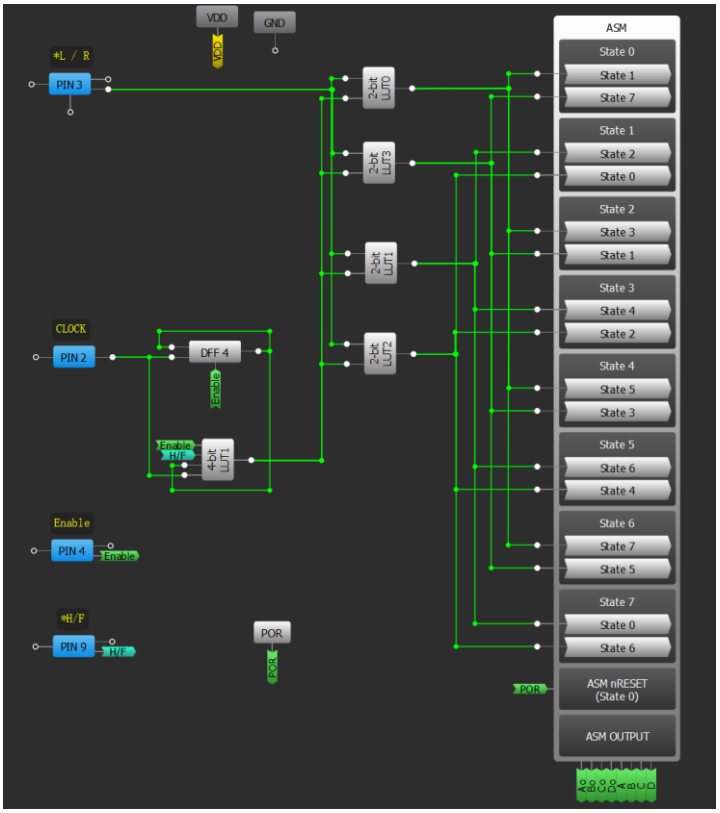
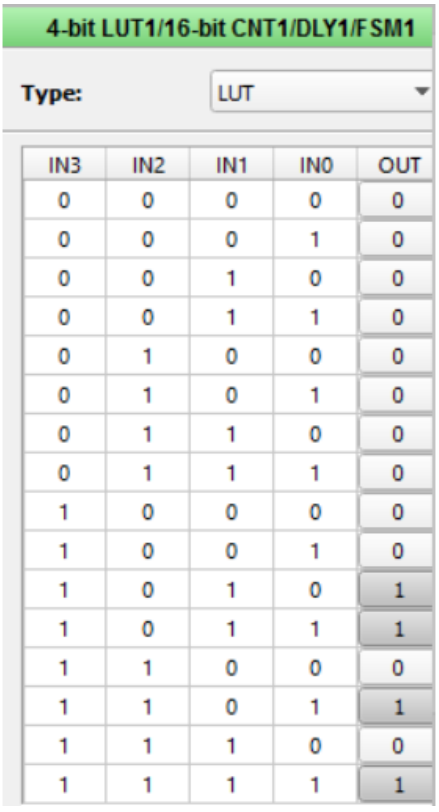


Figure 8. Clock control



IN3	IN2	IN1	IN0	OUT
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	1

Figure 9. 4-but LUT1 configuration

The output module can be seen in figure 10, with 4-bit LUT0 configuration in figure 11.

The logic of each output depends on the corresponding pin's ASM output (A), the corresponding ASM flag output (Ao), the enable signal and the motion direction signal. If the Half Step method is selected, the pin copies the pin output of the ASM. If the Full Step method is selected, the pin is high when the corresponding ASM output is high or when the corresponding ASM flag output is high. In this way, the output is high in two states of the same step.

In other words, considering that the ASM changes to another state on both edges of clock, and that every full step corresponds to two states of the ASM, a full step is done on every rising edge of the clock.

For outputs B, C and D, the same logic is used but is implemented with two 3-bit LUTs because GreenPAK doesn't have more 4-bit LUTs.

All the outputs and the clock module also depend on the Enable signal. The Enable signal determines whether the output signal of the modules is always low (when the user disables the controller) or whether the outputs are enabled.

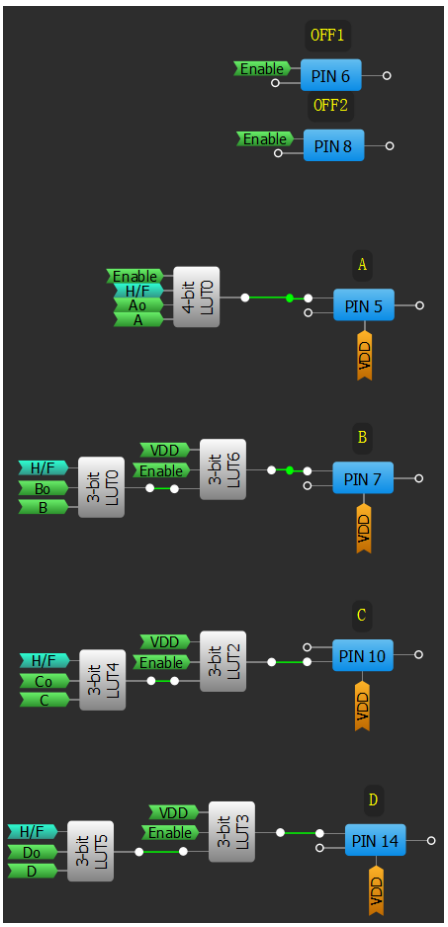


Figure 10. Output module

4-bit LUT0/WS Ctrl/16-bit CNT0/DLY0..

Type: LUT

IN3	IN2	IN1	IN0	OUT
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

Figure 11. 4-but LUT0 configuration

Also, this input is copied to OFF 1 and OFF 2 outputs. The entire implementation is shown in figure 12.

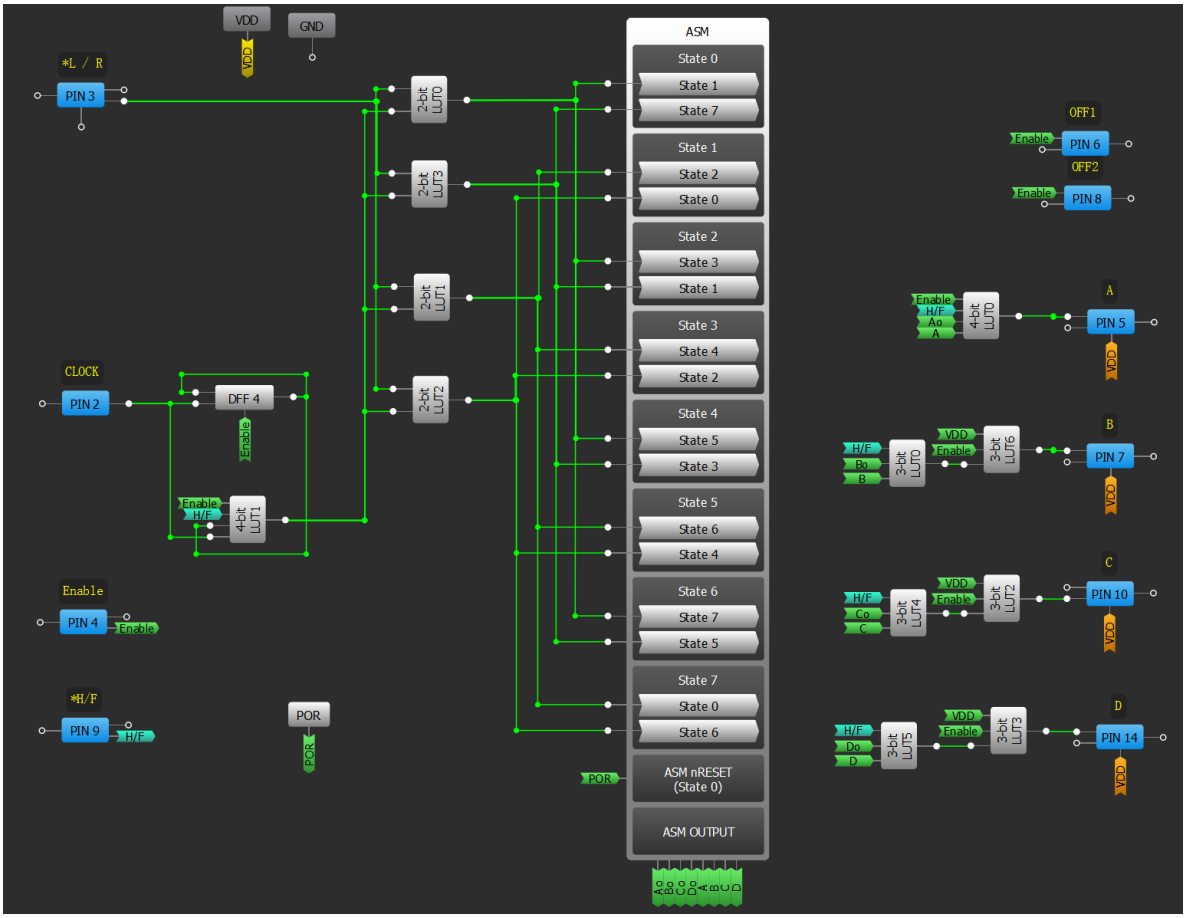


Figure 12. Stepper Motor Controller block diagram

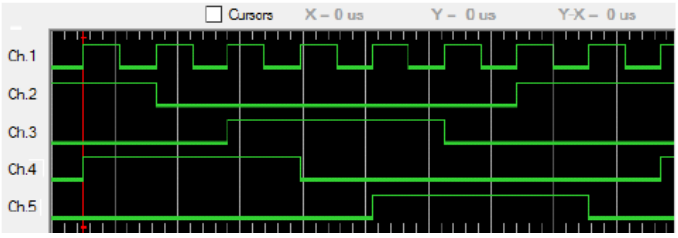


Figure 13. Half Step outputs test

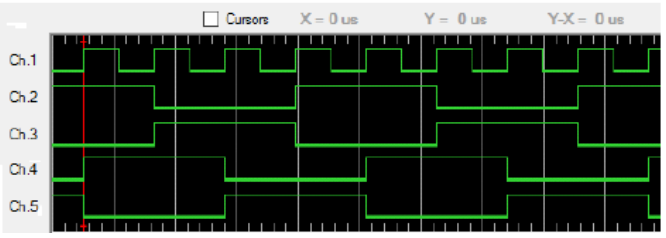


Figure 14. Full Step outputs test

Tests and Conclusion

To test the implementation, signals Clock, A, B, C and D were registered (in this order) with a logic analyzer.

The half step method outputs can be seen in figure 13 and the Full Step method outputs can be seen in figure 14.

In this application note, a stepper motor controller was implemented using SLG46531V. The user must set the direction, the controlling method and must also provide a clock to make the motor rotate.

Steps are done on every rising edge of the clock and the outputs are designed to control the driver circuit of the motor coils. With this implementation, the user only has to provide a clock signal without any other external circuit or controlled logic to generate step pulses.

IMPORTANT NOTICE AND DISCLAIMER

RENESAS ELECTRONICS CORPORATION AND ITS SUBSIDIARIES (“RENESAS”) PROVIDES TECHNICAL SPECIFICATIONS 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 OR 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 developers who are designing with Renesas products. You are solely responsible for (1) selecting the appropriate 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. Renesas grants you permission to use these resources only to develop an application that uses Renesas products. Other reproduction or use of these resources is strictly prohibited. No license is granted to any other Renesas intellectual property or to any third-party intellectual property. Renesas disclaims responsibility for, and you will fully indemnify Renesas and its representatives against, any claims, damages, costs, losses, or liabilities arising from your use of these resources. Renesas' products are provided only subject to Renesas' Terms and Conditions of Sale or other applicable terms agreed to in writing. No use of any Renesas resources expands or otherwise alters any applicable warranties or warranty disclaimers for these products.

(Disclaimer Rev.1.01)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Contact Information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit www.renesas.com/contact-us/.

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.