ECE 1770

ELECTRONIC MICROPROCESSOR SYSTEMS

Lab 6: Stepper Motor Control in Assembly

TASK#	Grading criteria	Instructor Initial
1	Your program runs successfully without any compilation error when you demo it. (10)_____ The behavior of the stepper motor follows the requirement of this task. (20)_____ Submit your complete and correct assembly codes on Courseweb. (20)_____	
2	Your program runs successfully without any compilation error when you demo it. (10)_____ The behavior of the stepper motor follows the requirement of this task. (20)_____ Submit your complete and correct assembly codes on Courseweb. (20)_____	
Bonus	The behavior of the stepper motor follows the requirement when press the up button of the joystick. (1)_____ The behavior of the stepper motor follows the requirement when press the down button of the joystick. (1)_____ The behavior of the stepper motor follows the requirement when press the left button of the joystick. (1)_____ The behavior of the stepper motor follows the requirement when press the left button of the joystick. (1)_____ Submit your complete and correct assembly codes on Courseweb. (1)_____	

In order to receive credit for this laboratory exercise, please submit this handout to your lab instructor when you are finished.

Team Members: Group Number:	TA:

ECE1770

Lab 6: Stepper Motor Control in Assembly

Jingtong Hu
March 10, 2019

1. Objective

1) Be familiar with the GPIO configuration
2) Program GPIO registers to perform more complex digital I/O output (stepper motor)
3) Understand the usage of full stepping and half stepping to control the speed and position of a stepper motor
4) Gain experience of generating pulse waveforms to control a stepper motor

2. Lab Assignments

1) Follow the textbook Chapter 16 Stepper Motor and implement an assembly program that makes the stepper motor rotate 360 degrees clockwise with full stepping
2) Make the stepper motor rotate 360 degrees anti-clockwise with half stepping
3) **Bonus**: Use buttons in the joystick to control the stepper motor (the detail will be discussed in the task description)

Note: In the project template on Courseweb, the first two tasks are implemented in *main.s*. Write your codes in this file and after you finish these two assignments, rename this file *lab6.s*. Submit the file to the entry *Lab 6 Group Submission*. For the bonus part, name your code file *lab6 bonus.s* and submit it to another entry *Lab 6 Bonus Group Submission*.

3. Operating Principle of Stepper Motor

As shown in Figure 1, the stepper motor is connected to the driver board, and the driver board is controlled by 4 GPIO pins (PB2, PB6, PB3 and PB7) of the microcontroller. Please wire the microcontroller, the driver board and the stepper motor according to Figure 2. Note that the headers on the back of the microcontroller board are directly connected to GPIO pins of the microcontroller. For example, the PB2 on the left headers, as highlighted in this Figure, is connected to GPIO Port B Pin 2 of the microcontroller.

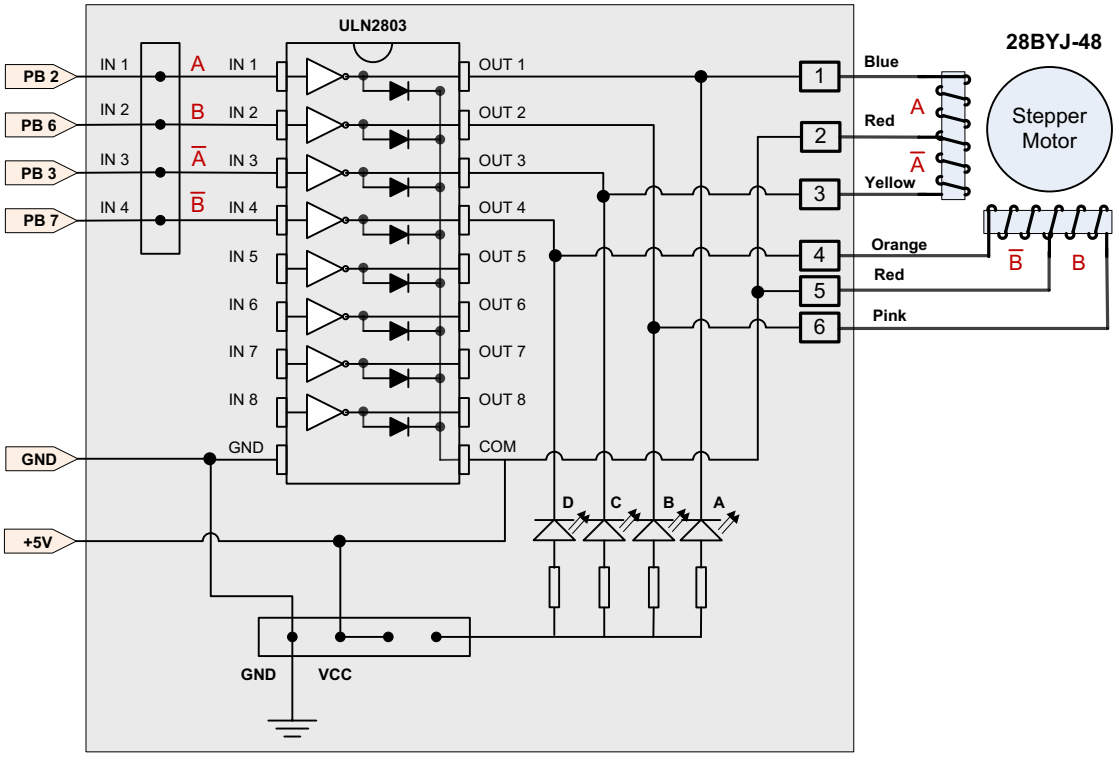


Figure 1. Connection of stepper motor

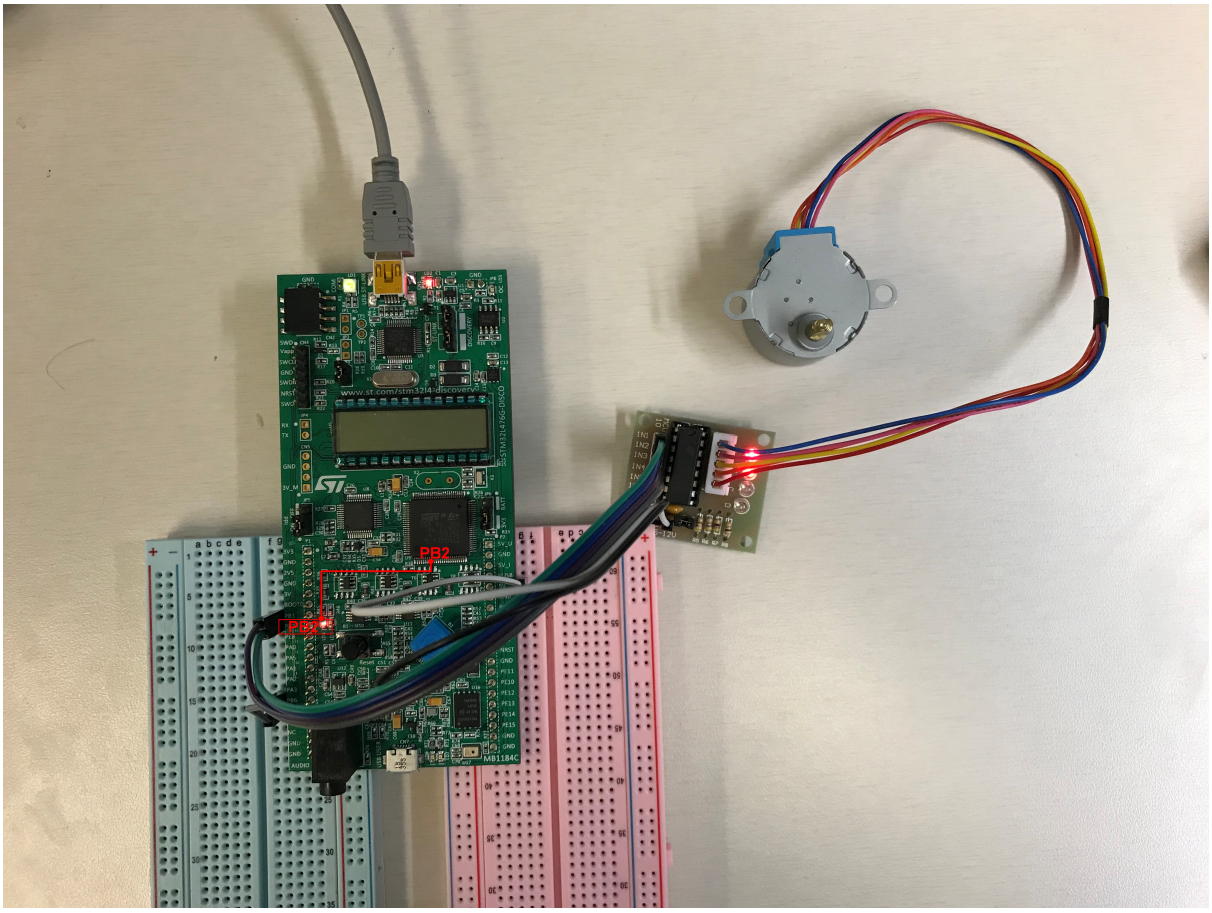


Figure 2. Wiring the microcontroller, the driver board and the stepper motor

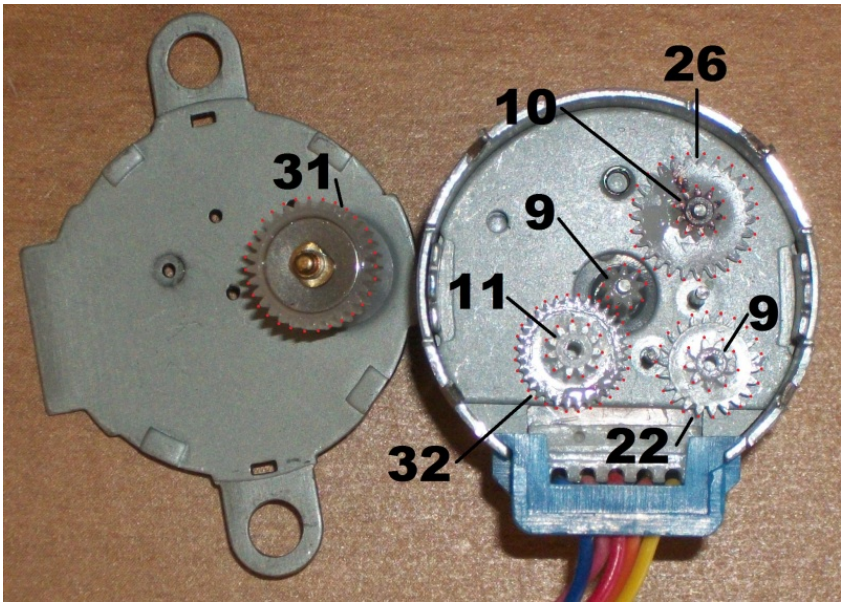


image from *forum.arduino.cc*

Figure 3 Gear ratio between the inner rotator and output shaft

The gears inside the stepper motor is shown in Figure 3. The gear ratio is:

$$\frac{31 \times 32 \times 26 \times 22}{11 \times 10 \times 9 \times 9} = 63.68395$$

The gear ratio means about **64** revolutions of the inner rotator result in 1 revolution of the output shaft. This gear ratio will be used when we calculate the number of steps per revolution for full stepping and half stepping.

There are two control methods for the stepper motor. One method is full stepping, and the other one is half stepping. Both of the two methods will be use in this lab.

Full-stepping

For simplicity, the operating principle of full stepping is shown by a stepper motor with 1 pair of poles on the rotator in Figure 4. The real stepper motor for this lab has 8 pairs of poles, and the number of steps will be calculated later.

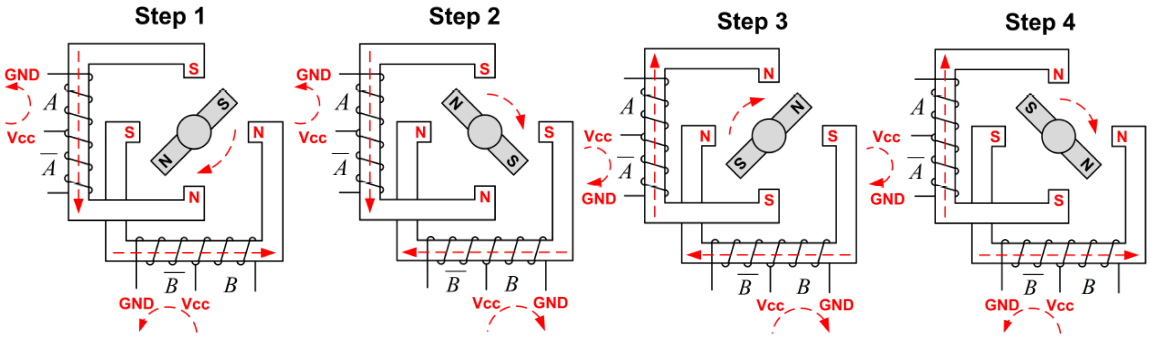


Figure 4. Operating principle of full stepping

As shown in Figure 4, for full stepping, there are 4 steps per revolution. In each step, the direction of current in the coils is controlled by the output in the GPIO pins of the

microcontroller (control sequence), and the poles on the stator is changed accordingly, which drives the rotator to rotate.

The control sequences of 4 steps are summarized in the Figure 5. For example, in step 1, the voltage of 4 GPIO pins connecting $A, \bar{A}, B, \bar{B}$ is high, low, low and high. You need to generate the correct output of GPIO pins following the control sequence shown in Figure 5 **within 4 steps**.

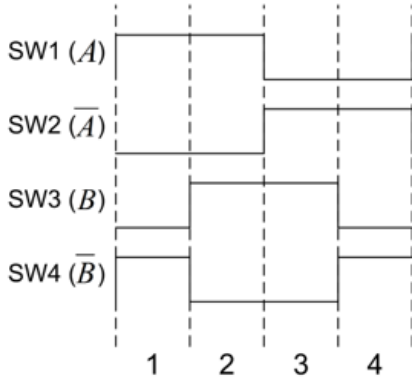


Figure 5. Control sequence of full stepping

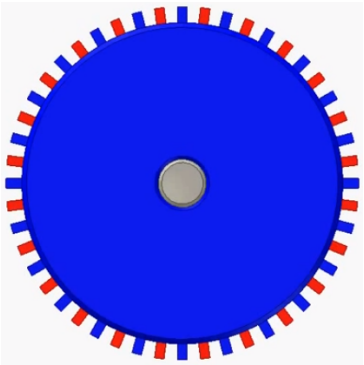


Figure 6. Many pairs of poles on the rotator

For the stepper motor we use in this lab, the number of steps per revolution is a little bit of different from theoretical analysis discussed above. And thus it needs to be adjusted. The stepper motor we use has 8 pairs of poles on the rotator, instead of 1 pair of poles in the previous analysis. As shown in Figure 6, the real rotator has many pairs of poles on the rotator. With more poles, the rotation angle per step is attenuated, and more steps are needed per revolution. To specify, with 8 pairs of poles on our stepper motor, we need 8 times more steps per revolution. Therefore we need 4*8=32 steps per revolution. Considering both the gear ratio and numbers of poles, the number of steps per revolution is adjusted from 4 to 2048, which is done by conducting the same 4 steps shown in Figure 5 **iteratively**. The detailed reasoning is listed as follows.

- Internal motor with 8 pairs of poles: 4*8=32 steps per revolution
- Great reduction ratio: 1/63.68395, approximately 1/64
- So it takes $32 \times 64 = 2048$ steps per revolution for the output shaft

Half-stepping

For simplicity, the operating principle of half stepping is shown by a stepper motor with 1 pair of poles on the rotator in Figure 7. The real stepper motor for this lab has 8 pairs of poles, and the number of steps will be calculated later.

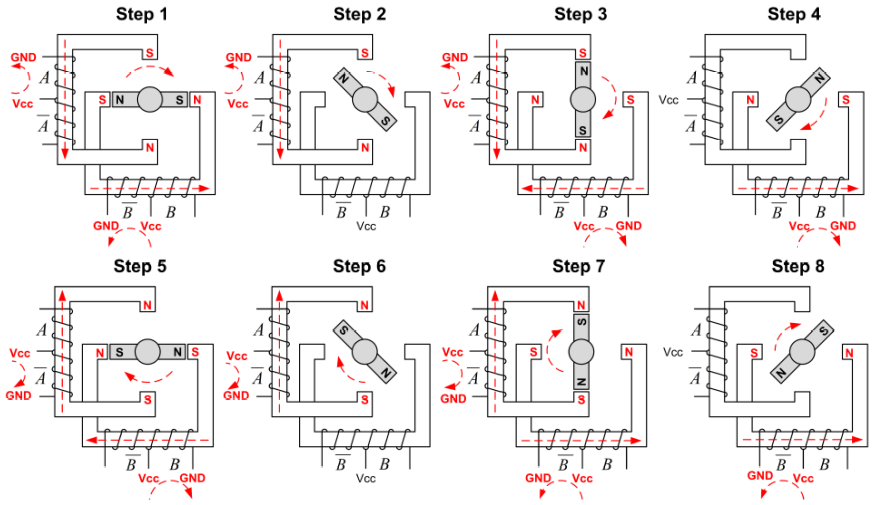


Figure 7. Operating principle of half stepping

As shown in the Figure 7, for half stepping, there are 8 steps per revolution. In each step, the direction of current in the coils is controlled by the output in the GPIO pins of the microcontroller (control sequence), and the poles on the stator is changed accordingly, which drives the rotator to rotate.

The control sequence of 8 steps are summarized in Figure 8. For example, in step 1, the voltage of 4 GPIO pins connecting $A, \bar{A}, B, \bar{B}$ is high, low, low and high, respectively. You need to generate the correct output of GPIO pins following the control sequence shown in Figure 8 **within 8 steps**.

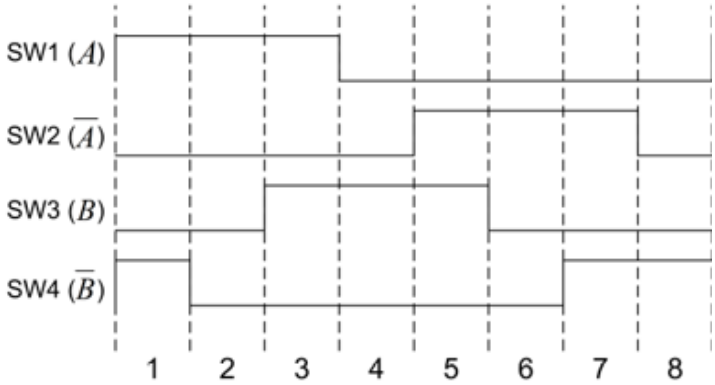


Figure 8. Control sequence of half stepping

Similar to full stepping, the number of steps per revolution for half stepping is adjusted from 8 to 4096, which is done by conducting the same 8 steps shown in Figure 8 **iteratively**. The detailed reasoning is listed as follows.

- Internal motor with 8 pairs of poles: 8*8=64 steps per revolution
- Great reduction ratio: 1/63.68395 ≈ 1/64
- So it takes $64 \times 64 = 4096$ steps per revolution for the output shaft

4. Pre-Lab

Please read this part and fill in the blanks before coding. It is conducive to finishing your lab tasks.

Videos about Stepper Motor:

If you want to know more about stepper motor, you can watch the video tutorial: *How the Stepper motors are made and how they operate*. The link is as follows:

a. Part 1 (5 minutes): http://www.youtube.com/watch?v=MHdz3c6KLrg
b. Part 2 (8 minutes): http://www.youtube.com/watch?v=t-3VnLadIbc

Interfacing the stepper motor requires four pins. We select the following four pins to control the stepper motor: **PB 2**, **PB 3**, **PB 6**, and **PB 7**. So before you write the codes related to spinning the stepper motor, you need to configure these GPIO pins first.

1. Enable the Clock of GPIO Port B (for Stepper Motor)

Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AHB2ENR														RNGEN		AESEN			ADCEN	OTGFSEN					GPIOPHEN	GPIOPGEN	GPIOPFEN	GPIOPEEN	GPIOPDEN	GPIOPCEN	GPIOPBEN	GPIOPAEN
Mask																																

Value of AHB2 Enable Register MASK for Port B = 0x_____ (in HEX)

2. Pin Initialization for Stepper Motor (PB 2, 3, 6, 7)

2.1 Configure PB 2, 3, 6, 7 as Output

GPIO Mode: Input (00), Output (01), Alternative Function (10), Analog (11, default)

Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODER	MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]		MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	

Mask_1																														
Mask_2																														

Before you set these pins, you need to clear the corresponding bits in the register. To achieve it, GPIOB Mode Register MASK_1 Value = 0x_____ (in HEX)

After you clear the corresponding bits, you need to set some bits to configure these pins as output.
To achieve it, GPIOB Mode Register MASK_2 Value = 0x_____ (in HEX)

2.2 Configure PB 2, 3, 6, 7 Output Type as Push-Pull

Push-Pull (0, reset), Open-Drain (1)

Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OTYPER	Reserved																OT15	OT14	OT13	OT12	OT11	OT10	OT9	OT8	OT7	OT6	OT5	OT4	OT3	OT2	OT1	OT0
Mask																																

GPIOB Output Type Register MASK Value = 0x_____(in HEX)

2.3 Configure PB 2, 3, 6, 7 Output Type as No Pull-up No Pull-down

NO PUPD (00, reset), Pullup (01), Pulldown (10), Reserved (11)

Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PUPDR	PUPDR15[1:0]		PUPDR14[1:0]		PUPDR13[1:0]		PUPDR12[1:0]		PUPDR11[1:0]		PUPDR10[1:0]		PUPDR9[1:0]		PUPDR8[1:0]		PUPDR7[1:0]		PUPDR6[1:0]		PUPDR5[1:0]		PUPDR4[1:0]		PUPDR3[1:0]		PUPDR2[1:0]		PUPDR1[1:0]		PUPDR0[1:0]	
Mask																																

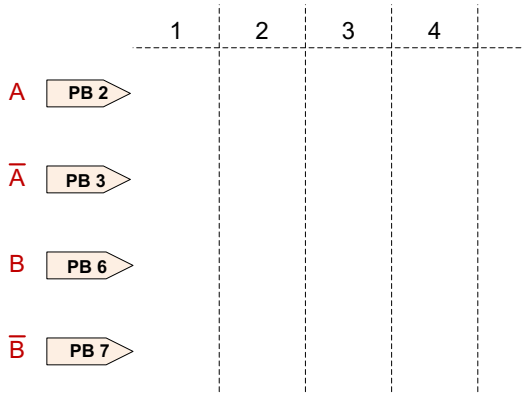
GPIOB Pull-up Pull-down Register MASK Value = 0x_____ (in HEX)

3. Spin the Stepper Motor

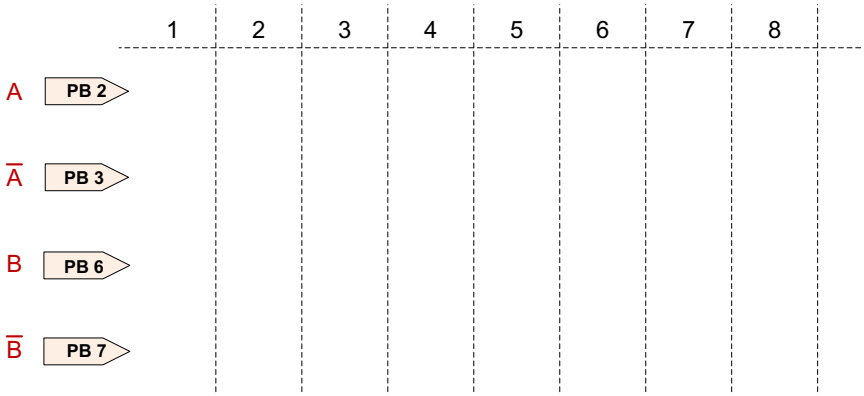
As we mentioned in Section 3 about full stepping and half stepping, you need to execute the control sequence for a certain number of times to finish a revolution with full stepping or half stepping. In order to execute the control sequence as required, you need to change the output of the different GPIO pins periodically.

Refer to Figure 5 and Figure 8, please complete the following two diagrams. You need to draw the pulse waveforms for different pins.

Full stepping sequence



Half stepping sequence



In this lab, the high voltage means the bits of corresponding pins should be set. And the low voltage means the corresponding bits should be cleared. According to the diagrams you finished above, please write down the output sequence in Output Data Register (ODR) for port B:

For full stepping, the value output sequence *stepsfull* (in HEX) is 0x_____ (Step1), 0x_____ (Step2), 0x_____ (Step3), 0x_____ (Step4).

For half stepping, the value output sequence *stepshalf* (in HEX) is 0x_____ (Step1), 0x_____ (Step2), 0x_____ (Step3), 0x_____ (Step4), 0x_____(Step5), 0x_____ (Step6), 0x_____ (Step7), 0x_____ (Step8).

Warning: Motor Overheating

The motor constantly draws electrical currents. The motor will be overheated if you leave the power on for an extended period. Make sure to disconnect the power (Vcc) to the Darlington array if you are not debugging/testing it.

Tasks

Note: For the following tasks, we have already provided part of the codes in the template projects for lab 6 on Courseweb. Please download it to your computer. Try to imitate the existing codes in *main.s* to finish the rest part of it. The comments are crucial hints for coding. Make full use of them. In some lines, you may need some value as a mask for specific registers, which can be found in the **Pre-Lab** section if you have finished them. You may also need some other constants predefined in *stm32l476xx_constants.s*. Refer to the slides of the related class may help you to do this.

Task 1: Full Stepping

In this task, you should make the stepper motor spin 360 degrees clockwise with full step. And when this task finishes, the program should jump to execute task 2.

Hint: *The total steps for full stepping is 2048. Use the output sequence mentioned in the pre-lab to set the value in Output Data Register (ODR) for port B every **four** steps. Remember to clear the corresponding bits before you set them. Besides, after each step, the program will delay for some time by calling the function delay near the end of the code section in main.s of project template and then execute the next step.*

Note: It is possible that the stepper motor spins anti-clockwise if you following the control sequence you get in the pre-lab. If this happens, just reverse your sequence and try again. And the for the function *delay* shown in Figure 9, you should pass the value of parameter to r0. The value in r0 represents the time to wait. We recommend you to use 5000 here, which means the program delays for 5000 clocks after each steps.

```
; call delay after each step
delay   PROC
        SUB r0, r0, #1          ; delay--
        NOP
        CMP r0, #0              ; exit when delay == 0
        BGT delay
        BX LR
```

Figure 9. Detail of function *delay*

Task 2: Half Stepping

After you finish the task 1, the program should execute the codes for task 2. In task 2, you should make the stepper motor spin 360 degrees anti-clockwise with half step. The codes for task 2 are similar to task 1.

Hint: *The total steps for full stepping is 4096. Use the output sequence mentioned in the pre-lab to set the value in Output Data Register (ODR) for port B every* ***eight*** *steps. Remember to clear the corresponding bits before you set them. Besides, after each step, the program will delay for some time by calling the function delay near the end of the code section in main.s of project template and then execute the next step.*

Note: It is possible that the stepper motor spins clockwise if you following the control sequence you get in the pre-lab. If this happens, just reverse your sequence and try again.

Bonus:

In the bonus part, you need to use buttons in the joystick to control the stepper motor. The requirement is as follows:

a. Up Button (PA3): spin 180 degrees clockwise with full stepping
b. Right Button (PA2): spin 360 degrees clockwise with full stepping
c. Left Button (PA1): spin 180 anticlockwise with full stepping
d. Down Button (PA5): spin 360 degrees anticlockwise with full stepping

Try to get the correct control sequence for these different cases by running your program in the first two tasks with different initial sequence. Besides, if you want the spin the stepper motor 180 degrees rather than 360 degrees, you just need to reduce the total number of steps by half. For example, since spinning 360 degrees with full stepping needs 2048 steps, it only needs 1024 steps to spin 180 degrees.

Hint: In lab 5, you already know how to read the input of the center button of the joystick. And it is similar to read the input from other buttons of the joystick. Besides, in lab5, after the center button is pressed and released, the Red LED is toggled. The control flow of the bonus part is similar to it, except that the stepper motor is spun instead of toggling the Red LED. You can use your codes for lab 5 to help you finish the bonus part more quickly.