Experiment 11: Controlling a Stepper Motor

Objectives

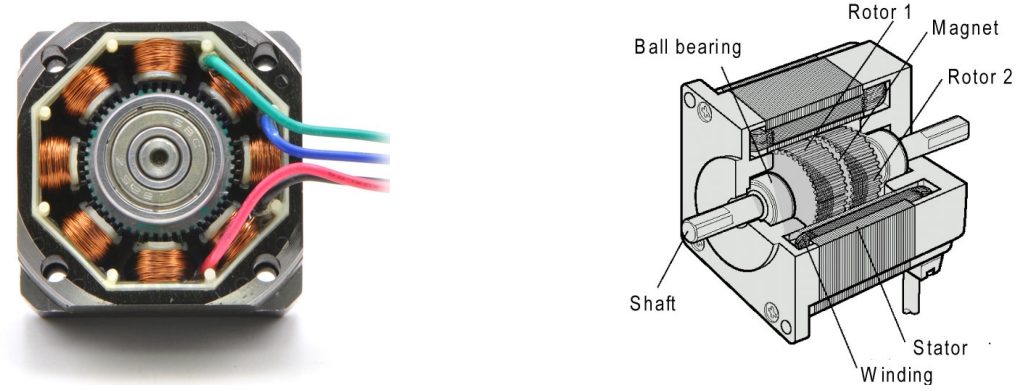
The purpose of this experiment is to analyze the direct driving of the stepper motor using the Cerebot MX7cK processor board.

Equipment List

- ChipKIT™ Pro MX7 processor board with USB cable
- Microchip MPLAB ® X IDE
- MPLAB ® XC32++ Compiler
- PmodSTEP™
- Stepper Motor (5V-12V, 25Ω, unipolar or bi-polar)

Overview

Stepper motors are DC motors that move in discrete steps. They have multiple coils that are organized in groups called "phases". By energizing each phase in sequence, the motor will rotate, one step at a time. The stepper motor consists of two sets of field windings positioned around a permanent magnet rotor, as illustrated in Figure 26.



(a) Photograph of field coils and rotor

(b) Internal Structure

Figure 26: 2-Phase Stepper Motor.

There are a wide variety of stepper types, some of which require very specialized drivers. In this lab, we are interested in the 2-phase stepper motor which is a mechanical stepper motor that consists of basically two parts, a stator and a rotor as shown in Figure 26b. The rotor in turn is made up of three components; rotor cup 1, rotor cup 2 and a permanent magnet. In a 2-phase motor the stator is made up of 8 magnetic poles with small teeth (refer to Figure 26a). The poles in the stator are each provided a winding.

A 2-phase motor has two phases, and the number of phases refers to the different combinations of poles that are energized in sequence to attract the rotor. The combinations of voltages applied to the four control terminals of the field windings control the magnitude and direction of the current through the windings, as illustrated in Figure 27. The current through the windings create an electromagnet. The motor shaft rotates to a position that minimizes the reluctance path between the field winding electromagnet's north and south poles of the rotor.

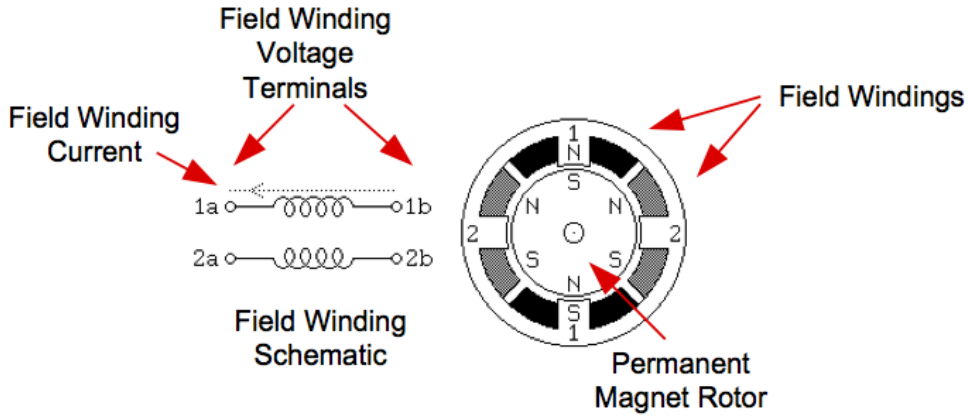


Figure 27: Bipolar Stepper motor diagram.

Considering the combinations of voltages on the winding terminals as possible control states, there are only eight states that produce current in the field windings, as shown in Table 11. In order to move the rotator shaft from one stable position to the physically adjacent stable position, the control voltages must switch to one of four out of the eight possible combinations of voltages. The action of moving from one stable position to an adjacent stable position is referred to as either a full-step or a half-step. Half-step increments are half the angular rotation of full-steps. Repeating a sequence of full- or half- step movements at a uniform rate will cause the rotator shaft to appear to rotate at a constant speed in discrete steps.

Step Control		**Winding Voltage**			
Step Name	**Hex Code**	**1a**	**1b**	**2a**	**2b**
S0_5	0x0A	H	L	H	L
S1	0x08	H	L	L	L
S1_5	0x09	H	L	L	H
S2	0x01	L	L	L	H
S2_5	0x05	L	H	L	H
S3	0x04	L	H	L	L
S3_5	0x06	L	H	H	L
S0	0x02	L	L	H	L

Table 11: Stepper motor control codes.

In this experiment, we are using a 2-phase 1.8° stepping angle stepper motor[8]. The stepping angle determines the positioning resolution of the motor which is the number of steps per revolution (expressed as degrees per step). A 1.8° motor is the same as a 200 step/revolution motor.

[8]Stepper motor model 39HS02 from www.leadshine.com.

The PmodSTEP driver module was designed to use I/O PORT B pins 7 through 10 that are assigned designations SM1 through SM4 to control the voltages on the four stepper motor field windings. (See Appendix E for PmodSTEP connection details.). To power the Pmod, do not forget to connect H5 connectors to power supply (controlled by Power supply jumper JP1).

Experiment Task

Part1:

Write a program to control the direction and step-mode of a stepper motor as shown in Table 12.

- Each time a button is pressed, the motor should complete a single revolution then it stops.
- Choose an appropriate delay (in ms) between motor steps (you can use timer driver for that).
- Your code should handle the button bouncing problem (software debounce).

Inputs		**Control Modes**	
BTN2	**BTN1**	**Direction**	**Step Mode**
OFF	OFF	CW	FS
OFF	ON	CW	HS
ON	OFF	CCW	HS
ON	ON	CCW	FS

Table 12: Task: Stepper motor controls.

Part2:

In this part you need to create an embedded design for a garage door control using the Cerebot MX7cK processor board.

Functional Specifications:

1. All buttons function as momentary contact push buttons.
   (a) BTN1 – Up
   (b) BTN2 – Down
   (c) BTN1 and BTN2 simultaneously pressed – SAFETY BEAM (LED1)
   (d) BTN3 – OPERATE BUTTON
2. LED Assignments
   (a) LEDA – Motor running up – door is opening
   (b) LEDB – Motor running down – door is closing
   (c) LEDC – Door is fully open
   (d) LEDD – Door is fully closed
3. If the door is closed (LEDD is on), pressing the button (BTN1) causes the door to begin opening causing LEDA to turn on and LEDD to turn off
4. If the door reaches the top while opening, the motor will stop (LEDA off) and door fully open indication, LEDC, is turned on.

5. If the door is open (LEDC on), pressing the button (BTN2) causes the door to begin closing causing LEDB to turn on and LEDC to turn off.

6. If the door reaches the bottom while closing, the motor will stop causing LEDD to turn on and LEDB to turn off.

7. If the electric eye detects that the light beam has been interrupted (LED1) while the door is closing (LEDB on) the door will reverse direction and begin opening by turning LEDA on.

8. If the light beam is interrupted (LED1) while the door is opening then the door continues opening.

9. Pressing the button (BTN3) while the door is opening causes the door to stop (LEDA and LEDB off). Pressing the button again (BTN3) causes the door to begin closing (LEDB on).

10. Pressing the button (BTN3) while the door is closing causes the door to stop (LEDA and LEDB off). Pressing the button again (BTN3) causes the door to begin opening (LEDA on).

11. 50ms switch contact debouncing is required for all inputs.