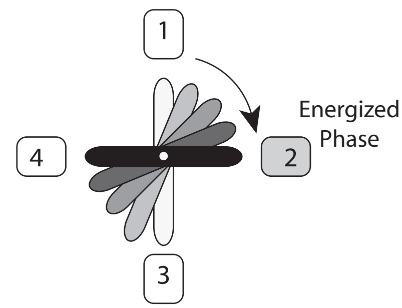
Stepping Motor Control

Charles O'Neill
19 November 2004

Description

This project's objective is to describe and control a DC stepping motor. Qualitative stepper motor theory is presented. An example driver using the PIC16F876 programmed in C is included.

Stepping motors are a class of multipole-multiphase DC motors with precision tracking capabilities. The stepping motor rotates in finite *steps* by energizing sequential motor phases; any further rotation requires energizing the next phase. The rotor typically contains the permanent magnet; the stator contains the coils.



The figure shows a 4 phase motor rotating from phase 1 to phase 2. Clockwise rotation require energizing —in sequence— the phases: 1, 2, 3, 4, and so on. Counterclockwise rotation requires the reverse order: 4, 3, 2, 1. Energizing each phase requires a voltage step to the appropriate coil(s) provided by a driver.

By necessity, multiple step sequences are required for one revolution. Common stepping motor resolutions are 200 and 400 steps —1.8° and 0.9° respectively. Most motors have about 4 phases, for a total of 5 wires when including the common or ground wire.

Advanced techniques are available for artificially increasing the step resolution: half-stepping, and micro-stepping. Half stepping recognizes that two adjacent phases can be energized simultaneously to move the rotor between steps. Micro-stepping adjusts each coil's field to provide a theoretically infinite resolution. The drawback is that a significantly more advanced driver is required.

Example

Using a stepping motor requires three parts: the actual motor, a driver, and a controller. The following figure shows a schematic for a stepping motor controlled by a PIC16F876.

Stepping motors and drivers are readily available from suppliers. However, 5.25 disk drives provide a source of DC stepping motors in the disk track mechanism

The driver converts low power inputs to high power outputs for each motor phase. A simple driver can be constructed from NPN transistors capable of more than about 500mA continuous. Diodes are needed to prevent back EMF or motor-generated current from burning out the transistors. An LED array is used to visualize the driver output.

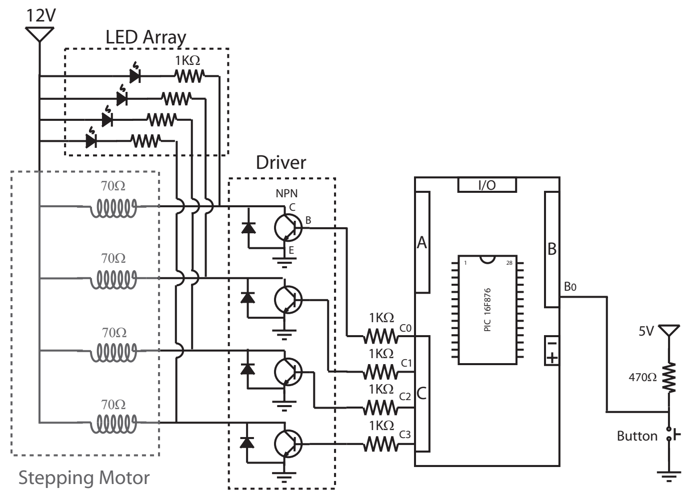
The controller is written in C for a PIC micro controller. Output ports are C0 through C3. Half stepping is implemented, so that the outputs are:

```
int positions[8]={0x01, 0x03, 0x02, 0x06, 0x04, 0x0C, 0x08, 0x09};
            Coils:  1     1+2   2     2+3   3     3+4   4     4+1
```

The controller sequentially steps through the `positions` array to control the proper port output as:

```
output_c(positions[motor_pole_position %8]);
```

A delay between step requests is needed to allow the motor to actually achieve that step. Increasing the delay slows the maximum motor rotation rate, but increases the available torque.



Results

The following C code shows an example stepping motor driver. This program rotates the motor counter clockwise forever. Within the main function is the following fragment:

```
void main(){
    while(1){
        motor_steps_desired=-1;
        motor_steps_desired= update_dial(motor_steps_desired);
        delay_ms(1);
    }
```

The update function corrects the motor by one step per call in the error direction.

```
signed int16 update_dial(signed int16 steps){
    /* Initialize */
    int positions[8]={0x01, 0x03, 0x02, 0x06, 0x04, 0x0C, 0x08, 0x09};
    motor_pole_position%=8;

    /* Rotate Clockwise */
    if(steps>0){
        output_c(positions[--motor_pole_position %8]);
        return(--steps);
        /* Rotate Counter Clockwise */
    } else if(steps<0){
        output_c(positions[++motor_pole_position %8]);
        return(++steps);
    /* No Rotation */
    } else {
        output_c(positions[motor_pole_position %8]);
        return(steps);
    }
}
```