

Web Site: www.parallax.com Forums: forums.parallax.com Sales: sales@parallax.com Technical: support@parallax.com Office: (916) 624-8333 Fax: (916) 624-8003 Sales: (888) 512-1024 Tech Support: (888) 997-8267

# 12-Volt Unipolar Stepper Motor (#27964)

#### Introduction

Stepper motors are electromechanical devices that convert a pattern of inputs and the rate-of-change of those inputs into precise rotational motion. The rotational angle and direction for each change (step) is determined by the construction of the motor as well as the step pattern input. The #27964 is a standard, four-phase unipolar stepper motor that is easily controlled with the BASIC Stamp or Javelin Stamp when buffered with an appropriate high-current driver (ULN2003 or similar suggested).

## **Applications**

- Robotics
- Motion Control and Industrial Equipment
- Techno-Art

# **Technical Specifications**

Rated Voltage 12 vdcRated Current/Phase 259 mA

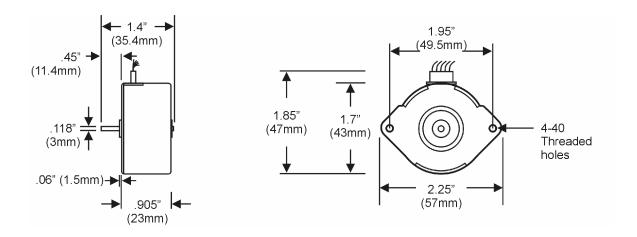
No. of Phase

• DC Coil Resistance 50  $\Omega$  / phase ±7% (100  $\Omega$  / coil)

• Step Angle 7.5° / phase

• Excitation Method 2-2 phase (unipolar)

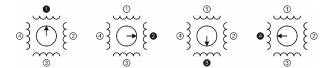
## **Mechanical Specifications**



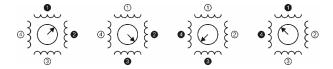
## **Stepper Basics**

Control of a stepper motor comes from applying a specific step sequence; rotational speed is controlled by the timing of the applied steps. The simplified diagrams below illustrate the effect of phase sequencing on rotational motion.

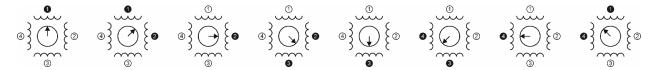
Full Step, Low Torque



Full Step, High Torque (standard application)

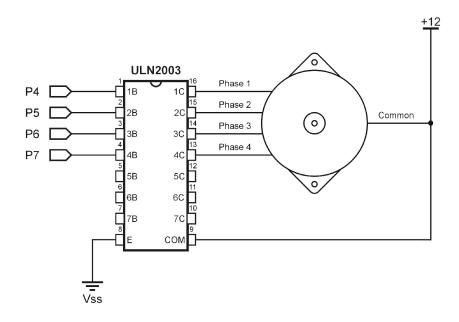


Half Step (best precision):



#### **Circuit Connections**

Use the circuit below to connect a 4-phase unipolar stepper motor to a BASIC Stamp or Javelin Stamp. The ULN2803 may also be used and has enough driver circuits to control two stepper motors (*be sure to verify motor current requirement versus ULN2x03 sink capability for multiple outputs*).



#### **Motor Connections**

Use the table below when connecting your stepper motor to the driver circuit.

Manufacturer	Mitsumi <sup>1</sup>	Howard Industries <sup>2</sup>
Degrees per Step	7.5	3.6
Steps per Revolution	48	100
Phase 1	Black	Brown
Phase 2	Orange	Green
Phase 3	Brown	Red
Phase 4	Yellow	White
Common	Red	Black

<sup>1</sup> Current motor supplied as PN #27964

## **BASIC Stamp 1 Application**

This program demonstrates basic full-step, high-torque control of a unipolar stepper motor. Step sequence data is stored in an EEPROM table to simplify program design and modification. Separate subroutines are used to manipulate the step index pointer that controls rotational direction. Note that the stepper update routine (Do\_Step) is designed to preserve the state of IO pins that are not being used by the stepper motor.

```
' {$STAMP BS1}
' {$PBASIC 1.0}
SYMBOL StpsPerRev = 48
                                               ' whole steps per rev
SYMBOL idx
                      = B2
                                               ' loop counter
                                               ' new phase data
SYMBOL phase
                      = B3
SYMBOL stpIdx
                                               ' step pointer
                      = B4
                                                ' delay for speed control
SYMBOL stpDelay
                       = B5
Full Steps:
  EEPROM 0, (%00110000, %01100000, %11000000, %10010000)
Setup:
                                                ' make P4..P7 outputs
  DIRS = %11110000
  stpDelay = 15
                                                ' set step delay
Main:
                                                ' one revolution
  FOR idx = 1 TO StpsPerRev
   GOSUB Step Fwd
                                                ' rotate clockwise
 NEXT
                                                ' wait 1/2 second
  PAUSE 500
  FOR idx = 1 TO StpsPerRev
                                                ' one revolution
                                                ' rotate counter-clockwise
   GOSUB Step Rev
 NEXT
```

Motor originally supplied with StampWorks kit

```
PAUSE 500
                                                 ' wait 1/2 second
  GOTO Main
  END
Step Fwd:
  stpIdx = stpIdx + 1 // 4
                                                 ' point to next step
  GOTO Do Step
Step Rev:
  stpIdx = stpIdx + 3 // 4
                                                 ' point to previous step
  GOTO Do_Step
Do Step:
                                                 ' read new phase data
  READ stpIdx, phase
 PINS = PINS & %00001111 | phase
                                                ' update stepper pins
                                                 ' pause between steps
 PAUSE stpDelay
 RETURN
```

## **BASIC Stamp 2 Application**

This program demonstrates basic full-step, high-torque control of a unipolar stepper motor using the BS2 family of microcontrollers. With the BS2 family, the programmer can take advantage of an IO structure (IO pins grouped as nibbles) which simplifies programming.

```
' {$STAMP BS2}
' {$PBASIC 2.5}
Phase
             VAR
                    OUTB
                                            ' phase control outputs
StpsPerRev CON 48
                                            ' whole steps per rev
idx
             VAR
                    Byte
                                            ' loop counter
stpIdx
                     Nib
                                            ' step pointer
             VAR
stpDelay
                                            ' delay for speed control
             VAR
                    Byte
            DATA %0011, %0110, %1100, %1001
Steps
Setup:
 DIRB = %1111
                                            ' make P4..P7 outputs
 stpDelay = 15
                                            ' set step delay
Main:
 FOR idx = 1 TO StpsPerRev
                                            ' one revolution
                                            ' rotate clockwise
  GOSUB Step Fwd
 NEXT
                                            ' wait 1/2 second
 PAUSE 500
                                            ' one revolution
 FOR idx = 1 TO StpsPerRev
```

```
' rotate counter-clockwise
    GOSUB Step Rev
  NEXT
  PAUSE 500
                                                 ' wait 1/2 second
  GOTO Main
  END
Step Fwd:
  stpIdx = stpIdx + 1 // 4
                                                 ' point to next step
  GOTO Do Step
Step Rev:
  stpIdx = stpIdx + 3 // 4
                                                 ' point to previous step
 GOTO Do Step
Do Step:
  READ (Steps + stpIdx), Phase
                                                 ' output new phase data
  PAUSE stpDelay
                                                 ' pause between steps
 RETURN
```

### **Javelin Stamp Application**

This program demonstrates basic, full-step control of a unipolar stepper motor using the Javelin Stamp microcontroller. The Stepper class handles the connection details and contains methods that allow the programmer to specify the number of steps, as well as the delay (in milliseconds between steps).

#### Stepper Motor Class

```
package stamp.misc;
import stamp.core.*;
public class Stepper {
 private int stpIdx = 0;
                                                // current step index
  private int phlPin;
                                                // phase 1 control pin
                                                // phase 2 control pin
  private int ph2Pin;
  private int ph3Pin;
                                                // phase 3 control pin
                                                // phase 4 control pin
  private int ph4Pin;
  public Stepper(int ph1Pin, int ph2Pin, int ph3Pin, int ph4Pin) {
    this.ph1Pin = ph1Pin;
    this.ph2Pin = ph2Pin;
    this.ph3Pin = ph3Pin;
    this.ph4Pin = ph4Pin;
  private void setFullStep(int theStep) {
```

```
switch (theStep) {
     case 0:
       CPU.writePin(ph1Pin, true);
       CPU.writePin(ph2Pin, true);
       CPU.writePin(ph3Pin, false);
       CPU.writePin(ph4Pin, false);
       break;
     case 1:
       CPU.writePin(ph1Pin, false);
       CPU.writePin(ph2Pin, true);
       CPU.writePin(ph3Pin, true);
       CPU.writePin(ph4Pin, false);
       break;
     case 2:
       CPU.writePin(ph1Pin, false);
       CPU.writePin(ph2Pin, false);
       CPU.writePin(ph3Pin, true);
       CPU.writePin(ph4Pin, true);
       break;
     case 3:
      CPU.writePin(ph1Pin, true);
       CPU.writePin(ph2Pin, false);
       CPU.writePin(ph3Pin, false);
       CPU.writePin(ph4Pin, true);
       break;
public void stepFFwd(int steps, int msDelay) {
  while (steps-- > 0) {
   stpIdx = (stpIdx + 1) % 4;
    setFullStep(stpIdx);
    CPU.delay(msDelay * 10);
  }
}
public void stepFRev(int steps, int msDelay) {
  while (steps-- > 0) {
   stpIdx = (stpIdx + 3) % 4;
   setFullStep(stpIdx);
    CPU.delay(msDelay * 10);
  }
}
```

## **Test Program for Stepper Class**