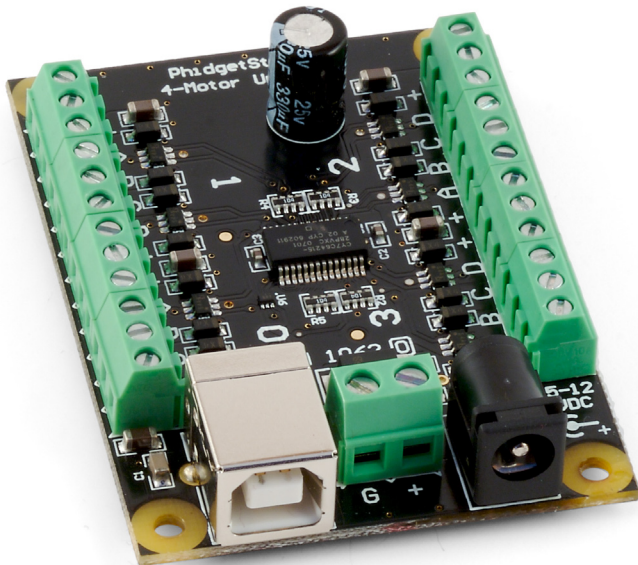


# Product Manual

## 1062 - PhidgetStepper Unipolar 4-Motor



**Phidgets 1062 - Product Manual**  
**For Board Revision 0**  
**© Phidgets Inc. 2009**

# Contents

## 5 Product Features

- 5 Programming Environment
- 5 Connection

## 6 Getting Started

- 6 Checking the Contents
- 6 Connecting all the pieces
- 6 Testing Using Windows 2000/XP/Vista
  - 6 Downloading the Phidgets drivers
  - 6 Running Phidgets Sample Program
- 7 Testing Using Mac OS X
- 8 If you are using Linux
- 8 If you are using Windows Mobile/CE 5.0 or 6.0

## 9 Programming a Phidget

- 9 Architecture
- 9 Libraries
- 9 Programming Hints
- 9 Networking Phidgets
- 10 Documentation
  - 10 Programming Manual
  - 10 Getting Started Guides
  - 10 API Guides
- 10 Code Samples
- 10 API for the PhidgetStepper Unipolar 4-Motor
  - 10 Functions
  - 12 Events

## 13 Technical Section

- 13 Introduction to Stepper Motors
- 13 How to connect your Stepper to the 1062
  - 15 Controlling Steppers - Open and Closed Loop
- 16 Stepping Mechanism
- 16 Continuous Rotation

- 16 Disabling the PhidgetStepper Unipolar
- 16 Starting the motor
- 17 Mechanical Drawing
- 17 Device Specifications

## **18 Product History**

## **18 Support**

# Product Features

---

- The PhidgetStepper Unipolar 4-Motor allows you to control the position, velocity, and acceleration of up to 4 unipolar stepper motors.
- The 1062 can be used in applications that require precise positioning and continuous rotation, at low cost.
- Requires an external 5 to 12VDC power supply.

**Note:** Make sure that power supply you are using matches your motor specifications. Giving a 5V stepper a 12V supply would cause the motor to run a 6 times higher power than it is rated for, and would likely destroy it.

## Programming Environment

**Operating Systems:** Windows 2000/XP/Vista/7, Windows CE, Linux, and Mac OS X

**Programming Languages (APIs):** VB6, VB.NET, C#.NET, C++, Flash 9, Flex, Java, LabVIEW, Python, Max/MSP, and Cocoa.

**Examples:** Many example applications for all the operating systems and development environments above are available for download at [www.phidgets.com](http://www.phidgets.com) >> Programming.

## Connection

The board connects directly to a computer's USB port.

# Getting Started

---

## Checking the Contents

### You should have received:

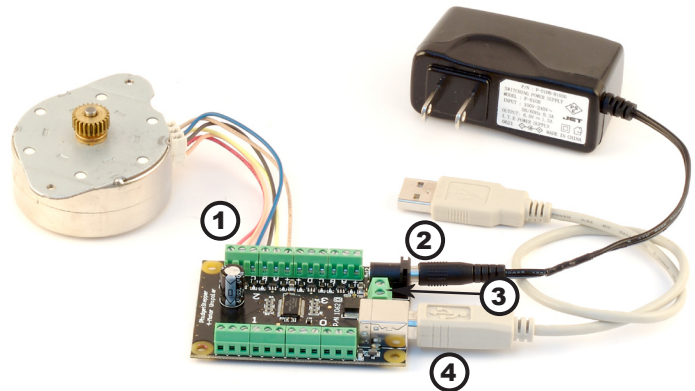
- A PhidgetStepper Unipolar 4-Motor
- A USB Cable

### In order to test your new Phidget you will also need:

- A 5 to 12V DC Power Supply (make sure the power supply does not exceed your motor voltage ratings.)
- A stepper motor (5, 6, or 8 wire)

## Connecting all the pieces

1. Connect the motor to the PhidgetStepper board. If you are having difficulty connecting your motor, refer to the Technical Section in this manual.
2. Connect the power supply to the board using the barrel connector.
3. Power supplies with higher current (more than 2.5 Amps) should be wired directly to the terminal block.
4. Connect the PhidgetMotorControl board to your PC using the USB cable.




## Testing Using Windows 2000/XP/Vista

### Downloading the Phidgets drivers

Make sure that you have the current version of the Phidget library installed on your PC. If you don't, do the following:

Go to [www.phidgets.com](http://www.phidgets.com) >> Drivers

Download and run Phidget21 Installer (32-bit, or 64-bit, depending on your PC)

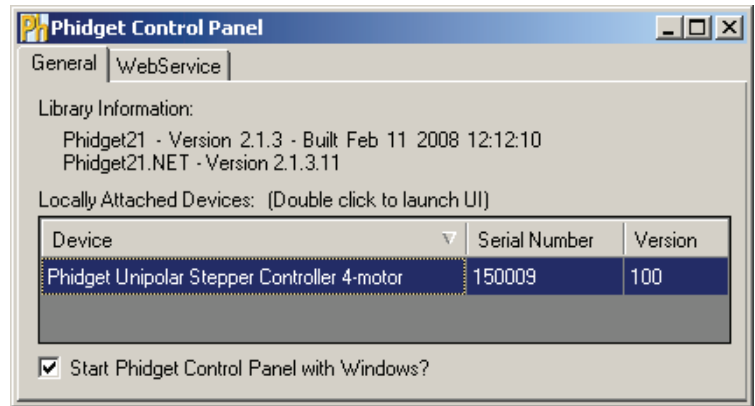
You should see the  icon on the right hand corner of the Task Bar.

### Running Phidgets Sample Program

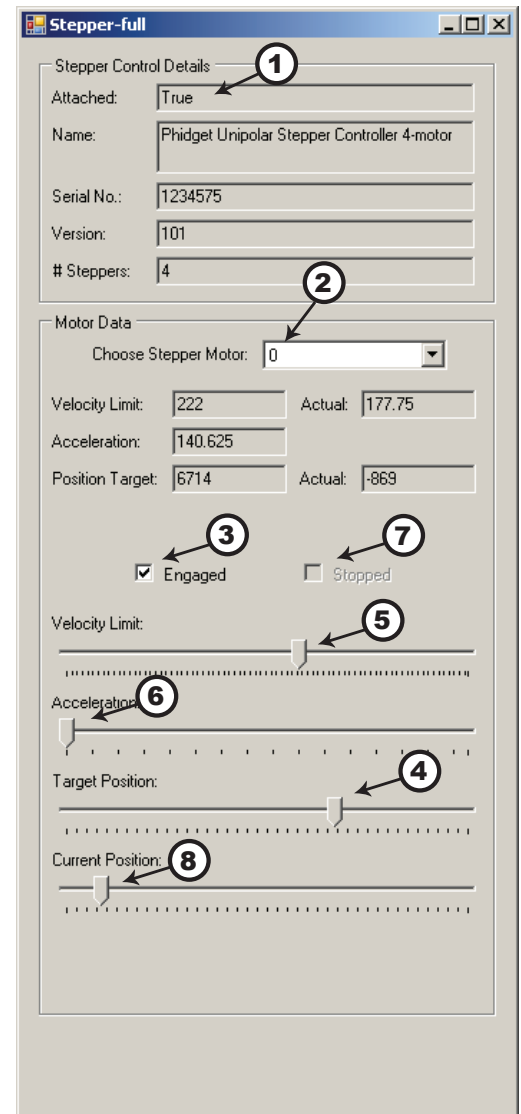
Double clicking on the  icon loads the Phidget Control Panel; we will use this program to make sure that your new Phidget works properly.

The source code for the Stepper-Full sample program can be found under C# by clicking on Phidget.com > Programming.

Double Click on the  icon to activate the Phidget Control Panel and make sure that the **Phidget Unipolar Stepper Controller 4-motor** is properly attached to your PC.



1. Double Click on **Phidget Unipolar Stepper Controller 4-motor** in the Phidget Control Panel to bring up Stepper-full and check that the box labelled Attached contains the word True.
2. Select the connected motor. If you have connected your motor at the same place as the one in the picture on page 3, it should be at position 2.
3. Check the Engaged box to power up the motor.
4. Move the Target Position slider to the right or the left. The target motor position will be displayed in the Position Target box and the motor will start turning until the Actual position is the same as the target.
5. Use the Velocity Limit slider to set the maximum velocity. The motor will accelerate until the Actual velocity is equal to the Velocity Limit.
6. Use the Acceleration slider to increase or decrease the acceleration.
7. When the motor has reached the position target, a tick mark will appear in the Stopped box.
8. When the motor is stopped, you can reset the current motor Position by using the Current Position slider.



## Testing Using Mac OS X

- Click on System Preferences >> Phidgets (under Other) to activate the Preference Pane
- Make sure that the **Phidget Unipolar Stepper Controller 4-motor** is properly attached.
- Double Click on **Phidget Unipolar Stepper Controller 4-motor** in the Phidget Preference Pane to bring up the Stepper-full example. This example will function in a similar way as the Windows version, but note that it does not include an Advanced Sensor Display.

## If you are using Linux

There are no sample programs written for Linux.

Go to [www.phidgets.com](http://www.phidgets.com) >> Drivers

Download Linux Source

- Have a look at the readme file
- Build Phidget21

The most popular programming languages in Linux are C/C++ and Java.

Notes:

Many Linux systems are now built with unsupported third party drivers. It may be necessary to uninstall these drivers for our libraries to work properly.

Phidget21 for Linux is a user-space library. Applications typically have to be run as root, or udev/hotplug must be configured to give permissions when the Phidget is plugged in.

## If you are using Windows Mobile/CE 5.0 or 6.0

Go to [www.phidgets.com](http://www.phidgets.com) >> Drivers

Download x86, ARMV4I or MIPSII, depending on the platform you are using. Mini-itx and ICOP systems will be x86, and most mobile devices, including XScale based systems will run the ARMV4I.

The CE libraries are distributed in .CAB format. Windows Mobile/CE is able to directly install .CAB files.

The most popular languages are C/C++, .NET Compact Framework (VB.NET and C#). A desktop version of Visual Studio can usually be configured to target your Windows Mobile Platform, whether you are compiling to machine code or the .NET Compact Framework.



# Programming a Phidget

---

Phidgets' philosophy is that you do not have to be an electrical engineer in order to do projects that use devices like sensors, motors, motor controllers, and interface boards. All you need to know is how to program. We have developed a complete set of Application Programming Interfaces (API) that are supported for Windows, Mac OS X, and Linux. When it comes to languages, we support VB6, VB.NET, C#.NET, C, C++, Flash 9, Flex, Java, LabVIEW, Python, Max/MSP, and Cocoa.

## Architecture

We have designed our libraries to give you the maximum amount of freedom. We do not impose our own programming model on you.

To achieve this goal we have implemented the libraries as a series of layers with the C API at the core surrounded by other language wrappers.

## Libraries

The lowest level library is the C API. The C API can be programmed against on Windows, CE, OS X and Linux. With the C API, C/C++, you can write cross-platform code. For systems with minimal resources (small computers), the C API may be the only choice.

The Java API is built into the C API Library. Java, by default is cross-platform - but your particular platform may not support it (CE).

The .NET API also relies on the C API. Our default .NET API is for .NET 2.0 Framework, but we also have .NET libraries for .NET 1.1 and .NET Compact Framework (CE).

The COM API relies on the C API. The COM API is programmed against when coding in VB6, VBScript, Excel (VBA), Delphi and Labview.

The ActionScript 3.0 Library relies on a communication link with a PhidgetWebService (see below). ActionScript 3.0 is used in Flex and Flash 9.

## Programming Hints

- Every Phidget has a unique serial number - this allows you to sort out which device is which at runtime. Unlike USB devices which model themselves as a COM port, you don't have to worry about where in the USB bus you plug your Phidget in. If you have more than one Phidget, even of the same type, their serial numbers enable you to sort them out at runtime.
- Each Phidget you have plugged in is controlled from your application using an object/handle specific to that phidget. This link between the Phidget and the software object is created when you call the .OPEN group of commands. This association will stay, even if the Phidget is disconnected/reattached, until .CLOSE is called.
- The Phidget APIs are designed to be used in an event-driven architecture. While it is possible to poll them, we don't recommend it. Please familiarize yourself with event programming.

## Networking Phidgets

The PhidgetWebService is an application written by Phidgets Inc. which acts as a network proxy on a computer. The PhidgetWebService will allow other computers on the network to communicate with the Phidgets connected to that computer. ALL of our APIs have the capability to communicate with Phidgets on another computer that has the PhidgetWebService running.

The PhidgetWebService also makes it possible to communicate with other applications that you wrote and that are connected to the PhidgetWebService, through the PhidgetDictionary object.

# Documentation

## Programming Manual

The Phidget Programming Manual documents the Phidgets software programming model in a language and device unspecific way, providing a general overview of the Phidgets API as a whole. You can find the manual at [www.phidgets.com](http://www.phidgets.com) >> Programming.

## Getting Started Guides

We have written Getting Started Guides for most of the languages that we support. If the manual exists for the language you want to use, this is the first manual you want to read. The Guides can be found at [www.phidgets.com](http://www.phidgets.com) >> Programming, and are listed under the appropriate language.

## API Guides

We maintain API references for COM (Windows), C (Windows/Mac OSX/Linux), Action Script, .Net and Java. These references document the API calls that are common to all Phidgets. These API References can be found under [www.phidgets.com](http://www.phidgets.com) >> Programming and are listed under the appropriate language. To look at the API calls for a specific Phidget, check its Product Manual.

## Code Samples

We have written sample programs to illustrate how the APIs are used.

Due to the large number of languages and devices we support, we cannot provide examples in every language for every Phidget. Some of the examples are very minimal, and other examples will have a full-featured GUI allowing all the functionality of the device to be explored. Most developers start by modifying existing examples until they have an understanding of the architecture.

Go to [www.phidgets.com](http://www.phidgets.com) >> Programming to see if there are code samples written for your device. Find the language you want to use and click on the magnifying glass besides "Code Sample". You will get a list of all the devices for which we wrote code samples in that language.

## API for the PhidgetStepper Unipolar 4-Motor

We document API Calls specific to this product in this section. Functions common to all Phidgets and functions not applicable to this device are not covered here. This section is deliberately generic. For calling conventions under a specific language, refer to the associated API manual. For exact values, refer to the device specifications.

### Functions

#### **int MotorCount() [get]**

Returns the number of motors this PhidgetStepper can control. In the case of the 1062, this will always return 4. This call does not return the number of motors actually connected - on the 1062, there is no way of programmatically finding out if motors are connected.

#### **double Acceleration(int MotorIndex) [get,set]**

Acceleration is the maximum change in velocity the PhidgetStepper uses when speeding up/ slowing down the motor. This is specified in the same units used for MotorPosition - in the case of the 1062, half-steps.

- If your motor is heavily loaded, or not supplied with a high enough voltage, there will be a practical limit on how fast it can accelerate.
- The range of valid Acceleration permitted is bounded by the software properties AccelerationMax/ AccelerationMin.
- This property should be set by the user as part of initialization. If not set, this value will remain unknown, and could be any of: Minimum acceleration, mid-point acceleration, or any value previously set by another application.

#### **double AccelerationMax(int MotorIndex) [get] : Constant**

AccelerationMax is the Maximum Acceleration the 1062 can accept, and apply to the motor. That does not mean that your motor can accelerate that fast!

### **double AccelerationMin(int MotorIndex) [get] : Constant**

AccelerationMin is the Minimum Acceleration the 1062 can accept, and apply to the motor.

### **double Velocity(int MotorIndex) [get]**

Velocity returns the current speed that a particular motor is being driven at. In the case of the 1062, the unit is half-steps per second. With the PhidgetStepper, there is no way of directly controlling the velocity of a motor, because of acceleration curves, however the maximum velocity (VelocityLimit) can be set. The Velocity is returned from the 1062 - so there will be a delay - typically 30-50ms.

### **double VelocityLimit(int MotorIndex) [get,set]**

Sets the maximum velocity that the stepper controller will move the motor. Please note that this is not necessarily the speed that the motor is being turned at. The motor is accelerated to the VelocityLimit, and then decelerated as it approaches the target. If the target is close enough, you may never reach the VelocityLimit.

- VelocityLimit is bounded by VelocityMax/VelocityMin.
- This property should be set by the user as part of initialization. If not set, this value will remain unknown, and could be any of: 0 (motor won't move), mid-point velocity, or any value previously set by another application.
- Note that when VelocityLimit is set to 0, the motor will not move.

### **double VelocityMax(int MotorIndex) [get] : Constant**

VelocityMax is the Maximum VelocityLimit the 1062 can accept. Functionally, this is the maximum speed that the 1062 can drive your motors at.

### **double VelocityMin(int MotorIndex) [get] : Constant**

VelocityMin is the Minimum Velocity Limit the 1062 can accept.

### **int64 CurrentMotorPosition(int MotorIndex) [get,set]**

Returns the current position of a motor. Note that there will be some lag (typical 30-50ms) between the PhidgetStepper reporting a position, and that position being read by your application. CurrentMotorPosition is fixed-point - an increment of one is one half-step - the smallest step that the 1062 can move the motor.

Sets the position that the PhidgetStepper is at right now. This is useful for zeroing the position when a limit switch is reached, for example. To keep accurate track of position, CurrentMotorPosition should only be set when the MotorStopped property is true, because if this property is set while the motor is moving, the motor will have to decelerate to stop moving, before setting the current position.

### **int64 TargetMotorPosition(int MotorIndex) [get,set]**

Sets the desired motor position. Note that setting TargetMotorPosition will override a previous set TargetMotorPosition, and the motor will begin tracking to the new position immediately. The velocity of the motor will be ramped appropriately. TargetMotorPosition is bounded by MotorPositionMin, and MotorPositionMax.

Returns the last set TargetMotorPosition.

### **int64 MotorPositionMax(int MotorIndex) [get] : Constant**

MotorPositionMax is the Maximum MotorPosition the 1062 can accept. Functionally, this is the largest position that the 1062 can move your motors toward. The initial MotorPosition is halfway between Min and Max. Behaviour is undefined if MotorPosition is driven past Min or Max.

### **int64 MotorPositionMin(int MotorIndex) [get] : Constant**

MotorPositionMin is the Minimum MotorPosition the 1062 can move your motors toward.

### **bool Engaged(int MotorIndex) [get,set]**

Enables a particular stepper to be positioned. If this property is false, no power is applied to the motor. Note that when the motor is first Enabled, the coils may not be exactly aligned, and the motor will snap to position.

MotorOn is useful to reduce the power consumed by a motor once it's reached a given position. If you are concerned about keeping accurate track of position, MotorOn should not be disabled until MotorStopped = True.

### **bool Stopped(int MotorIndex) [get]**

MotorStopped guarantees that the motor is not moving (unless you are moving it by hand), and that there are no commands in the pipeline to the motor. Note that virtually any API calls will cause MotorStopped to be temporarily false, even changing Acceleration or VelocityLimit on a stopped motor.

## **Events**

### **VelocityChange(int MotorIndex, double Velocity) [event]**

An event issued when the velocity changes on a motor.

### **PositionChange(int MotorIndex, int64 Velocity) [event]**

An event issued when the position changes on a motor. You are not guaranteed to receive events for every motor position - updates are throttled at approximately 16ms.

# Technical Section

## Introduction to Stepper Motors

Stepper motors are broadly available motors commonly used for positioning. DC Motors are controlled by simply applying power which sends them blindly spinning. Steppers Motors are controlled in a series of discrete steps, allowing them to be sent to a very precise position. By repeating the set of steps over and over, the motor can be made to rotate while your system tracks the position.

Some steppers will require hundreds of steps to do a full rotation, while others, like the air core motor, can do a full rotation with only 4 steps.

Many stepper controllers rely on the resistance of the coils of wire inside the stepper motor to control the amount of current. In fact, the 1062 PhidgetStepper uses this simpler (and cheaper) method.

The vast majority of steppers are built using two coils of wire, whose magnetism pulls or repels a rotating magnet attached to an exposed shaft. Depending on how these two coils are available to wire up, we get 4, 5, 6 or 8 wire stepper motors.

Unipolar Stepper motors are available in 5, 6 or 8 wire configurations. By making the middle of the coils available for connection, the cost of the controlling electronics can be reduced. Driving a motor in the Unipolar configuration reduces the maximum torque the motor can produce.

The word Unipolar means that the coils have current passing in one direction only. When the center tap of each coil is connected to the power supply, the ends of each coil can be grounded or left floating (ungrounded) in sequence to generate the magnetic fields to rotate the motor.

## How to connect your Stepper to the 1062

Unipolar Stepper motors are available in 5, 6 or 8 wire configurations.

### 5 Wire Stepper Motors

In a 5 wire motor, the center taps of the coils are connected together. This scheme prevents this motor from being controlled as a bipolar motor.

To use a 5 wire motor as a unipolar, the center tap wire is connected to the power supply.

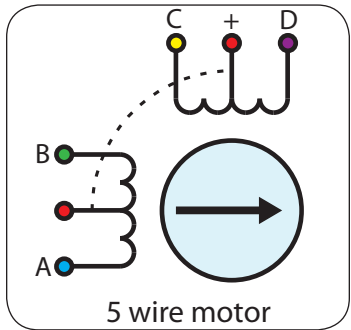
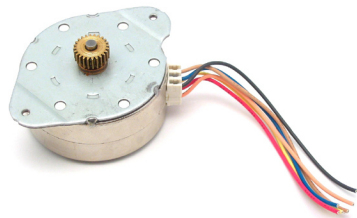
Determining how to connect a 5 wire stepper to a Unipolar Stepper Controller, like the 1062 can be done by following this procedure.

Start by measuring the resistance between all the wires. Below is a sample table of resistance data, in ohms. This table contains example values, your readings may be different but should still produce a similar pattern.

	Red	Green	Black	White	Brown
Red		147	74	147	147
Green			74	147	147
Black				74	74
White					147
Brown					

Looking at the table, you should notice a pattern; the black wire has the same resistance to the other four wires. This tells us that black is our + (center tap) wire, and should be wired to the power supply connection. On the 1062 PhidgetStepper, the power supply connection is labelled as (+). There are two power supply connections available on the 1062 for each motor - either can be used.

Pick one of the remaining four wires and wire it to the A terminal. Now carefully try connecting the remaining three wires to the B, C, D terminals until you find a sequence that results in the motor turning.



There are in fact two valid combinations, one that will produce a clockwise rotation in the stepper motor for increasing position and one to produce counter-clockwise rotation. In order to reverse this rotation, simply swap the (A, B) wire pair and the (C, D) wire pair.

6 Wire Stepper Motors - Unipolar

The process is similar to a 5 wire motor. On a 6 – wire motor, there will be two + wires, one for each coil, which are the center taps for each coil. You will need to isolate which are the center tap wires and the corresponding wires for their coil.

These center taps must be wired together to the power supply.

Let’s assume our six wire stepper motor wires are colored as follows: red, green, black, white, brown, and yellow.

We measure the resistance between all wires and are presented with the following values in ohms (these are simply example values) :

Looking at our table, we can see our pattern. The red wire has the same resistance to the brown and yellow wires. The green wire has the same resistance to the black and white wire. Red, brown, and

	Red	Green	Black	White	Brown	Yellow
Red		$\infty$	$\infty$	$\infty$	10	10
Green			10	10	$\infty$	$\infty$
Black				20	$\infty$	$\infty$
White					$\infty$	$\infty$
Brown						20
Yellow						

yellow bring out one coil, and green, black, and white are the other coil. The red and green wires are the center of their coils.

Connect red and green to the (+) terminal block connections on the PhidgetStepper. Pick one of the remaining four wires and wire it to the A terminal. Now carefully try connecting the remaining three wires to the B,C,D terminals until you find a sequence that results in the motor turning.

There are in fact two valid combinations, one that will produce a clockwise rotation in the stepper motor for increasing position and one to produce counter-clockwise rotation. In order to reverse this rotation, simply swap the (A, B) wire pair and the (C, D) wire pair.

8 Wire Stepper Motors - Unipolar

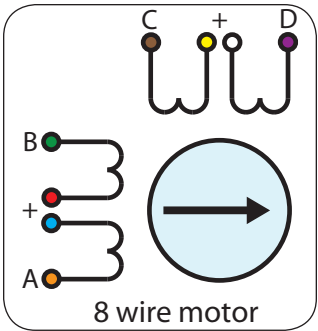
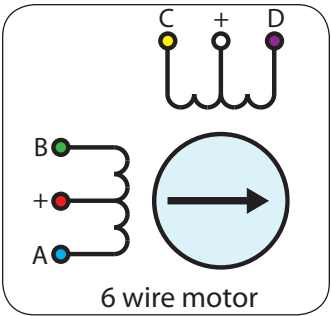
8 Wire Motors are very difficult to wire up if you do not have a schematic showing how the wires are connected to the internal coils. Only follow these instructions if you are **really** desperate.

In an 8 wire motor, the coils are split, and to operate it as a unipolar, we have to reconnect the coils to reduce it to a 6 wire unipolar.

Assume our eight wire stepper motor wires are colored as follows: red, yellow, black, orange, blue, green, brown, and white. In an 8-wire stepper motor, these wires would be part of 4 coils, 2 wires per coil. We need to determine the cable pairings.

We measure the resistance between each wire and are presented with the following values in ohms (these are simply example values):

This table tells us which wires are parts of a coil. From the table we can tell that red/



	Red	Yellow	Black	Orange	Blue	Green	Brown	White
Red		$\infty$	$\infty$	$\infty$	1	$\infty$	$\infty$	$\infty$
Yellow			$\infty$	$\infty$	$\infty$	1	$\infty$	$\infty$
Black				$\infty$	$\infty$	$\infty$	1	$\infty$
Orange					$\infty$	$\infty$	$\infty$	1
Blue						$\infty$	$\infty$	$\infty$
Green							$\infty$	$\infty$
Brown								$\infty$
White								

blue, yellow/green, black/brown, and orange/white are the coils.

We are now left with the following situation; we need to determine the proper orientation of the wires to determine our connections. Of each pair, one of the wires will be assigned to A, B, C, or D, and the other wire will be connected to another pair. The number of combinations to be tried to see if they produce rotation is large, but can be reduced to a maximum of 96 possibilities by following these steps:

1. Choose Red/Blue to connect to A. (2 possibilities)
2. Choose one wire of the other pairs (6 possibilities) and connect to B. The other wire from this pair is connected to the wire from Step 1 not connected to A.
3. Choose one wire from the two remaining pairs (4 possibilities) and connect to C.
4. Choose one wire from the remaining pair (2 possibilities) and connect to the wire from Step 3 not connected to C. The remaining wire from this pair is connected to D.
5. After trying each permutation, engage the motor from software and try to rotate it. Since you are driving the motor as Unipolar, the connected pairs should be connected to the (+) on the PhidgetStepper Controller.
6. If you attempt to use this algorithm, build a table of permutations beforehand and proceed in a systematic way.

There are a total of 96 wiring combinations, of which there are 2 valid combinations where one will cause a clockwise motor rotation and the other will cause a counter-clockwise rotation.

In order to properly determine the proper wiring for your motor we suggest consulting any manuals or data sheets that are associated with your particular motor.

## Controlling Steppers - Open and Closed Loop

Because stepper motors do not have the inherent ability to sense their actual shaft position, they are considered open loop systems. This means that the value contained in the current position property is merely a count of the number of steps that have occurred towards the target value; it can not be relied upon as a measure of the actual shaft angle, as external forces may also be affecting the motor.

There are several ways of overcoming this drawback. The simplest is to allow the motor load to depress a limit switch located at a known position. This can be used to fire an event in software to recalibrate the shaft position values. A more elegant solution might involve the mounting of an optical encoder on the shaft and the development of a control system.



# Stepping Mechanism

Example stepper motor with 15° step angle

The 1062 PhidgetStepper Unipolar controls stepper motors in half-step increments. A Position increment of one corresponds to one half-step. A stepper motor with 15 degree step increments will rotate in 7.5 degree steps. The 1062 accomplishes this by alternating the number of powered coils between one and two, always at least one coil powered. In this way, the rotor is positioned at both full steps and half steps. The table below describes the order in which coils are powered to achieve this.

Step Number	Coil(s) Powered				Shaft Angle
	A	B	C	D	
1	1	0	1	0	0°
2	1	0	0	0	7.5°
3	1	0	0	1	15°
4	0	0	0	1	22.5°
5	0	1	0	1	30°
6	0	1	0	0	37.5°
7	0	1	1	0	45°
8	0	0	1	0	52.5°

After step number 8 in the table, the order the coils are powered in simply repeats from the beginning. As the motor approaches the requested position, it is decelerated according to the value of the acceleration property. When the desired position has been reached, the 1062 stops the motor and holds it at that position.

## Continuous Rotation

A stepper motor can be caused to rotate continuously by simply setting the motor position property to an extremely large number. The valid range of values for the motor position property is large enough to be able to cause the motor to continuously turn at maximum velocity for 45 years.

## Disabling the PhidgetStepper Unipolar

When the stepper motor has rotated a requested number of half steps, and is stopped, the coils will remain energized to hold it in position. This is necessary to allow the motor to support a load on it's shaft without rotating to an unknown position.

The amount of current required to hold a motor shaft in place is called the holding current, and is based mainly on the resistance of the motor coils and the load being supported by the motor. The current required to produce the holding torque can often be large enough to cause the motor to generate heat from the power dissipated in the coils. If the motor is not supporting a load or is not required to maintain a specific angle, it is recommended to set the Enable property to false. This will allow the motor shaft to rotate freely, but the present angle may be lost if forces on the motor-shaft are greater than can be resisted by the detent torque of the unpowered motor.

## Starting the motor

When the steppers are first engaged from software, the stepper motor likely will not be at the same state as the default output state of the controller. This will cause the stepper to 'snap' to the position asserted by the controller - potentially moving by 2 full steps.

## High precision applications

Stepper motors precision are limited by the manufacturing process used to build them. Errors in the rotor and coils will cause some degree of inaccuracy. In our experience, inexpensive stepper motors will often have positioning errors approaching a half-step.

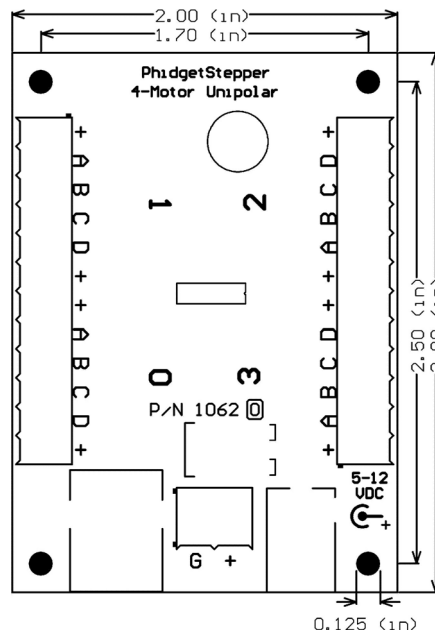
## Synchronization of multiple motors

Many applications call for several steppers motors operating in unison - for example, operating a CNC table, or a robot arm. Highly precise synchronization of steppers using the PhidgetStepper is not possible, as the sequencing will be affected by the real-time performance of your operating system. Each stepper is controlled as a independent unit, so there is no way of arranging for a particular action to happen to all motors at the same time. Typical jitter can be 10-30mS.



## Mechanical Drawing

1:1 scale



**Note:** When printing the mechanical drawing, “**Page Scaling**” in the Print panel must be set to “**None**” to avoid re-sizing the image.

## Device Specifications

Characteristic	Value
Output Controller Update Rate	62.5 updates/second/motor
Position Resolution	1/2 step (40-bit signed)
Upper Position Limit	$2^{39} - 1$ 1/2 steps
Lower Position Limit	$-(2^{39} - 1)$ 1/2 steps
Velocity Resolution	0.75 1/2 steps/second (9-bit)
Velocity Limit	383.25 1/2 steps/second
Acceleration Resolution	140.625 1/2 steps/second <sup>2</sup> (6-bit)
Acceleration Limit	8859.375 1/2 steps/second <sup>2</sup>
Minimum Power Supply Voltage	5V
Maximum Power Supply Voltage	12V
Max Current Per Coil	1A
USB-Power Current Specification	100mA max
Device Quiescent Current Consumption	23mA

**Note:** Current from USB supply is not available for motors

## Product History

Date	Board Revision	Device Version	Comment
April 2008	0	101	Product Release

## Support

- Call the support desk at 1.403.282.7335 8:00 AM to 5:00 PM Mountain Time (US & Canada) - GMT-07:00 or
- E-mail us at: [support@phidgets.com](mailto:support@phidgets.com)