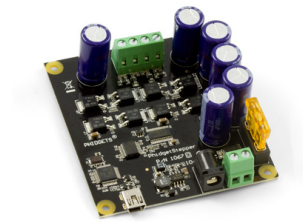


# 1067 User Guide

---



Go to this device's product page <sup>[1]</sup>

## Getting Started

### Checking the Contents

#### You should have received:

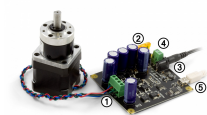
- A PhidgetStepperBipolar HC Controller Board
- A Mini-USB Cable
- A 5A or 10A fuse

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

- A 9 to 30V DC Power Supply (If you're using the barrel jack, make sure your power supply is **center positive** polarity)
- A bipolar stepper motor


### Connecting the Pieces

1. Connect the motor to the PhidgetStepper board. If you are having difficulty connecting your motor, refer to the technical section.
2. Push the fuse into the fuse holder located behind the power terminals.
3. Connect the power supply to the board using the barrel connector or the power terminal block.
4. Power supplies with higher current (more than 2.5 Amps) should be wired directly to the terminal block.
5. Connect the PhidgetStepper board to your PC using the USB cable.




### Testing Using Windows 2000 / XP / Vista / 7

Make sure you have the current version of the Phidget library installed on your PC. If you don't, follow these steps:

1. Go to the Quick Downloads section on the Windows page
2. Download and run the Phidget21 Installer (32-bit, or 64-bit, depending on your system)
3. You should see the  icon on the right hand corner of the Task Bar.

### Running Phidgets Sample Program

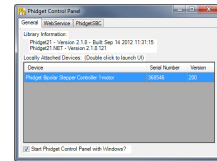
Double clicking on the  icon loads the Phidget Control Panel; we will use this program to ensure that your new Phidget works properly.

The source code for the **Stepper-full** sample program can be found in the quick downloads section on the C# Language Page. If you'd like to see examples in other languages, you can visit our Languages page.

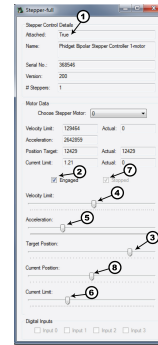
## Updating Device Firmware

If an entry in this list is red, it means the firmware for that device is out of date. Double click on the entry to be given the option of updating the firmware. If you choose not to update the firmware, you can still run the example for that device after refusing.

Double Click on the  icon to activate the Phidget Control Panel and make sure that **Phidget Bipolar Stepper Controller HC** is properly attached to your PC.



1. Double Click on **Phidget Bipolar Stepper Controller HC** in the Phidget Control Panel to bring up Stepper-full and check that the box labelled Attached contains the word True.
2. Check the Engaged box to power up the motor.
3. 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.
4. Use the Velocity Limit slider to set the maximum velocity. The motor will accelerate until the Actual velocity is equal to the Velocity Limit. Keep in mind that the maximum value on this slider is not actually the maximum velocity. See the API for more information.
5. Use the Acceleration slider to increase or decrease the acceleration.
6. Use the Current Limit slider to modify the current going through the motor and affect torque.
7. When the motor has reached the position target, a tick mark will appear in the Stopped box.
8. Move the Current Position slider to change the position the device is currently at.



## Testing Using Mac OS X

1. Go to the Quick Downloads section on the Mac OS X page
2. Download and run the Phidget OS X Installer
3. Click on System Preferences >> Phidgets (under Other) to activate the Preference Pane
4. Make sure that the Phidget Bipolar Stepper Controller HC is properly attached.
5. Double Click on Phidget Bipolar Stepper Controller HC in the Phidget Preference Pane to bring up the Stepper-full Sample program. This program will function in a similar way as the Windows version.

## Using Linux

For a step-by-step guide on getting Phidgets running on Linux, check the Linux page.

## Using Windows Mobile / CE 5.0 / CE 6.0

For a step-by-step guide on getting Phidgets running on Windows CE, check the Windows CE page.

## Technical Details

### How does the 1067 control Stepper Motors?

The 1067 can be used to control a bipolar stepper motor with 4, 6, or 8 wires.

The PhidgetStepper Bipolar is most effective when used with stepper motors designed to be driven with a chopper drive stepper controller. These motors are often large, with a rectangular industrial appearance. The specifications of these motors are often confusing and may seem contradictory. Our users are often confused attempting to relate the specifications of their motor to the capabilities of the 1067. If you are in doubt if your motor will work with the 1067, call us.

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.

Unlike the 1063, the 1067 will always micro-step, moving 1/16th of a step for each change in the position variable.

To avoid the use of floating point in the APIs, the position, velocity and acceleration parameters are expressed in microsteps (1/16th steps), instead of full steps. If your application wants to move the motor by 1 full step, you change the position by 16.

### Open and Closed Loop Control

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.

## How to Connect your Stepper to the 1067

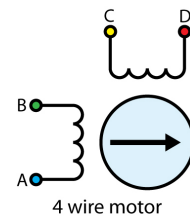
Bipolar Steppers motors are available in 4, 6 or 8 wire configurations. Driving a motor in the Bipolar Configuration produces maximum torque. The word Bipolar means that the controlling electronics have to be able to produce a current flow in either direction in each coil, to produce magnetic fields in opposite directions.

### 4 Wire Stepper Motors

A 4 wire Stepper motor can only be controlled as a bipolar. There are two coils, with the ends of each coil brought out. By applying current to the two coils in sequence, and changing the direction of current flow, the motor can be made to rotate.

Determining how to connect a 4 wire stepper to the 1067 can be done by following this procedure. Suppose we have four wires: red, green, yellow and blue.

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.



Wire Color	Green	Blue	Yellow	Red
Green		30	∞	∞
Blue			∞	∞
Yellow				30
Red				

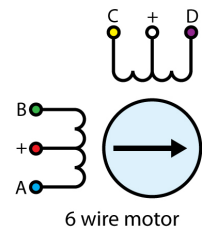
Looking at the table, you see that there are two pairs of wires (Green-Blue, and Yellow-Red), with roughly the same resistance measured between the wires in each pair. Pick one of the four wires and wire it to the A terminal. The other wire in this pair is connected to the B terminal. At random, wire the other pair to C and D. The motor will turn in clockwise or counter-clockwise rotation. Swapping the A and B wires or the C and D wires will change whether the motor considers “forward” to mean clockwise or counter-clockwise rotation (Remember, steppers can be run in forward or reverse by choosing a target position greater than or less than the current position).

6 Wire Stepper Motors

In a 6 – wire bipolar motor, there is 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 are left unconnected to operate as a bipolar. Let’s assume our six wire stepper motor wires are colored as follows: red, green, blue, white, purple, and yellow.

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



Wire Color	Red	White	Yellow	Purple	Green	Blue
Red		∞	∞	∞	10	10
White			10	10	∞	∞
Yellow				20	∞	∞
Purple					∞	∞
Green						20
Blue						

Looking at our table, we can see our pattern. The red wire has the same resistance to the green and blue wires. The white wire has the same resistance to the yellow and purple wire. Red, green, and blue are for one pole, and white, yellow, and purple are the other pole. The red and white wires are the center of their coils. Leave the red/white wires unconnected, and follow the instructions for connecting a 4-Wire Bipolar Motor as described earlier in this section.

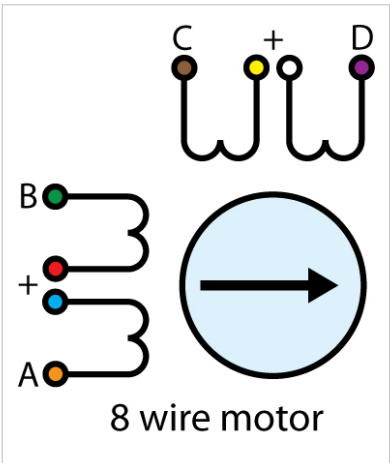
There are in fact two valid combinations, one that will produce a clockwise rotation in the stepper motor for increasing position and one that will produce counter-clockwise rotation. Swapping the A and B wires or the C and D wires will change whether the motor considers “forward” to mean clockwise or counter-clockwise rotation.

8 Wire Stepper Motors

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 we have to reconnect the coils, to reduce them to a 4 wire. Assume our eight wire stepper motor wires are colored as follows: red, yellow, purple, 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):



Wire Color	Orange	Red	Yellow	Purple	Blue	Green	Brown	White
Orange		∞	∞	∞	1	∞	∞	∞
Red			∞	∞	∞	1	∞	∞
Yellow				∞	∞	∞	1	∞
Purple					∞	∞	∞	1
Blue						∞	∞	∞
Green							∞	∞
Brown								∞
White								

This table tells us which wires are parts of a coil. From the table we can tell that red/green, yellow/brown, purple/white, and orange/blue 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 one wire from a pair 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. Do not connect the pairs to anything - as if you were using a 6-wire stepper in bipolar mode.

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.

If you are not using a Phidget stepper, we suggest consulting any manuals or data sheets that are associated with your particular motor in order to determine the proper wiring for your motor.

## Choosing a Power Supply Voltage

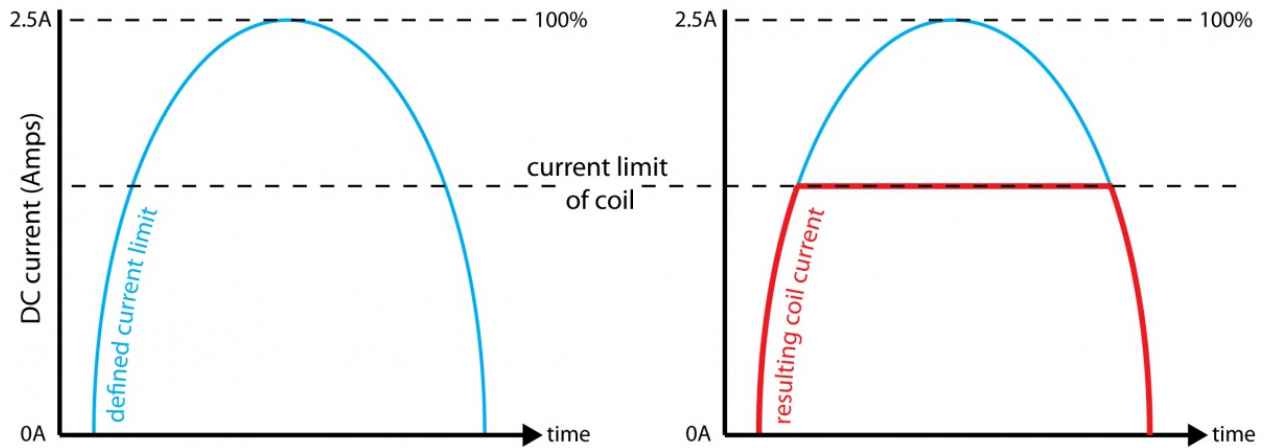
The 1067 can operate from 10 to 30 VDC. It is able to reduce the voltage during operation if your motor requires less, but it cannot increase the voltage. First, find the resistance of the coils in your motor. We will assume 4 Ohms. Add 0.5 Ohms to this to account for the electronics on the 1067. Decide the maximum current you want to push through the coils.

This current limit is per coil, and you set this through the software API when you write your application. The maximum current for your stepper is part of the motor specification - you can use the stepper with less current, to reduce power consumption, but you will get reduced torque. We'll assume the limit for this example is 2 Amps. Multiply 4.5 Ohms \* 2 Amps = 9 Volts. The minimum voltage you can operate this motor at will be 9 Volts + 0.5 Volts for the controller. If the voltage you provide is less than what is required for the motor (but still more than 9 volts), the motor will still run, but not at full spec, and may not step as smoothly as it could.

For more information about how supply voltage affects your motor's maximum speed and torque, check the Stepper Motor Primer.

## Setting Current Limit

The 1067 allows the current applied to the motor to be programmatically set. This is important - if the current limit is set too high, the motor's internal resistance will cause the sine-wave approximations used to implement microstepping to clip at the maximum current possible, given your motor/supply voltage. This clipping will cause rough operation, or prevent the motor from turning. If the limit is set too low, the motor may not be able to handle it's load, by missing steps, or not turning at all at high accelerations.



### How to determine CurrentLimit

- add method of finding optimal current limit without feedback

For more information about setting the current limit, check the Stepper Motor Primer.

## Continuous Rotation

A stepper motor can be caused to rotate continuously by simply setting the motor position property to an extremely large number of steps. 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 several years. You could also reset the current position variable whenever the motor reaches a certain position. If the target position is kept the same, this will also produce continuous rotation.

## Saving Power

Holding torque is the amount of torque required to rotate the motor 'against it's will' when the full rated current is flowing through the coils. If current consumption / heating is a problem, and full holding torque is not required, the Current Limit can be decreased in software when the motor is not moving. Holding torque will decrease linearly with the Current Limit.

There is also a small resistance to movement within the motor even when there is no current, called Detent Torque. This may be sufficient to prevent rotation in your application, particularly if the stepper has a gearbox on it.

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.

## High Precision Applications

Stepper motor precision is 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. Gearing a stepper motor down with a gearbox will reduce this error proportional to the reduction ratio of the gearbox.

## 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.

## Compatibility Guidelines

When looking for a motor that will be compatible with the 1067, check the motor's data sheet and make sure it meets the following specifications.

- **Bipolar motor** - While the 1067 can control some six-wire unipolar motors, we recommend you look for bipolar stepper motors.
- **4, 6, or 8-wire motor** - A 5-wire motor cannot be used with the 1067, because the coils of the motor are connected internally.
- **Rated/Recommended Voltage** - If the motor comes with a rated or recommended voltage, it should be no more than **30 volts**, and you should use a power supply that can output that voltage.
- **Rated Current** - The motor should be rated for a maximum of 4A per coil.

## Troubleshooting

If your stepper motor is not behaving correctly, you can use a multimeter to determine whether it is the controller or the motor that is at fault.

Disconnect the motor and set the current limit to 0. Set the target position to 7 and measure the voltage difference between the A and B terminals it should be either -12, or +12V assuming you are using a 12V power supply (just depending on which terminal you have the positive multimeter lead connected to). Then set the target position to 8, the voltage should drop to 0, and it will go to the opposite at 9 (-12 or +12). This pattern will repeat for 39,40,41 and 71,72,73 and 103,104,105 etc... in increments of 32. If this is behaving as expected then the problem is probably with how you have connected your motor, or your code.

## Further Reading

For more information about stepper motors, check the Stepper Motor and Controller Primer.

## API

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 in the Quick Downloads section for that language. 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 1067, this will always return 1.

This call does not return the number of motors actually connected.

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

Acceleration is the maximum change in velocity the PhidgetStepper uses when speeding up or slowing down the motor (But not when the motor is in mid-spin). This is specified in the same units used for Position - in the case of the 1067, 1/16th 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. If the acceleration is set too high, the motor may not be able to respond quickly enough and end up missing steps.
- The range of acceptable Acceleration is bounded by the software properties AccelerationMax and AccelerationMin.
- Acceleration should be set explicitly before beginning to use the PhidgetStepper, and will read as unknown until set by the application.
- VelocityLimit and Acceleration are set on the device in one discrete operation - if Acceleration is not set when VelocityLimit is set, it will default to (AccelerationMax / 2).

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

AccelerationMax is the maximum Acceleration the PhidgetStepper can accept and apply to the motor. In the case of the 1067, this will always return 10000000.

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

AccelerationMin is the minimum Acceleration the PhidgetStepper can accept and apply to the motor. In the case of the 1067, this will always return 2.

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

Velocity returns the current speed that a particular motor is being driven at, as the number of 1/16th steps per second. With the PhidgetStepper, there is no way of directly controlling the velocity of a motor. The Velocity is also returned from the device, so typically there will be a 30- 50 ms USB delay.

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

Sets the maximum velocity that the stepper controller will drive the motor. Please note that this is not necessarily the speed that the motor is being turned at. The motor is accelerated up to the VelocityLimit, and decelerated as it approaches the target. If the target position is close enough or Acceleration set low enough, the VelocityLimit might never be reached.

- The range of acceptable VelocityLimit is bounded by the software properties VelocityMax and VelocityMin.
- VelocityLimit should be set explicitly before beginning to use the PhidgetStepper.
- VelocityLimit and Acceleration are set on the device in one discrete operation. If VelocityLimit is not set when Acceleration is set, it will default to (VelocityMax / 2).
- Reducing the velocity limit to zero causes the controller to ramp down. This could be a good way of implementing a pause on a long move.

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

VelocityMax is the maximum VelocityLimit the PhidgetStepper can accept. In the case of the 1067, this will always return 250000. Functionally, it is the maximum speed that the 1067 can drive your motor at.

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



VelocityMin is the minimum VelocityLimit the PhidgetStepper can accept. In the case of the 1067, this will always return 0. Functionally, it is the minimum speed that the 1067 can drive your motor at. For very low speeds, you can increment position inside a slow software based timing loop. Note that if VelocityLimit is set to 0, the motor will not turn. The smallest velocity that will turn the motor is 1 for the 1067.

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

Get returns the current position of a motor in 1/16th step units. Note that there will be some delay (typically 30-50ms) between the PhidgetStepper reporting a position, and that position being read by your application. This property will return unknown when the motor is not engaged. When the motor is re-engaged, the property will return the last known position. Set allows you to overwrite the position that the PhidgetStepper is at right now. This can be useful for zeroing the position when a limit is reached. To keep accurate track of position, CurrentPosition should only be set when the Stopped property is true.

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

Sets the desired motor position. Note that setting TargetPosition will override the previous command, and the motor will begin tracking to the new position immediately. The velocity of the motor will be ramped appropriately. TargetPosition is bounded by PositionMin, and PositionMax.

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

MotorPositionMax is the maximum absolute MotorPosition the 1067 can accept. In the case of the 1067, this will always return 1E+15. Functionally, this is the largest position that the 1067 can move your motors toward. The initial MotorPosition is halfway between Min and Max. Behavior is undefined if MotorPosition is driven past Min or Max.

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

MotorPositionMin is the minimum MotorPosition the 1067 can move your motor toward. In the case of the 1067, this will always return 0.

#### **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 enabled, the coils may not be exactly aligned and the motor will snap to position.

- Engaged defaults to false.
- Engaged 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, Engaged should not be disabled until Stopped = True.

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

Stopped guarantees that the motor is not moving (without an external force), and that there are no commands in the pipeline to the motor. Note that virtually any API calls will cause Stopped to be temporarily false, including changing Acceleration or VelocityLimit on a stopped motor.

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

Sets the maximum current allowed through each coil of the stepper in amps. Current corresponds roughly to power consumption, and the torque produced by the motor. Reducing the current limit is often useful when the motor just has to hold a position. Please review the section of the user guide on determining an appropriate current limit.

- CurrentLimit should be set explicitly before beginning to use the motor.

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

CurrentMax is the largest CurrentLimit value the PhidgetStepper can accept. In the case of the 1067, this will always return 4.

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

CurrentMin is the smallest CurrentLimit the PhidgetStepper can accept. In the case of the 1067, this will always return 0.

## Events

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

Designates an event handler to run whenever the velocity changes on a motor.

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

Designates an event handler to run whenever the position changes on the motor. You are not guaranteed to receive events for every motor position as updates are throttled at approximately 30ms.

## Product History

The 1067 - PhidgetStepper HC replaced our previous bipolar stepper controller, the 1063 - PhidgetStepper Bipolar [2].

Date	Board Revision	Device Version	Comment
January 2013	0	200	Product Release

## References

[1] [http://www.phidgets.com/products.php?product\\_id=1067](http://www.phidgets.com/products.php?product_id=1067)

[2] [http://www.phidgets.com/products.php?product\\_id=1063](http://www.phidgets.com/products.php?product_id=1063)

# Article Sources and Contributors

**1067 User Guide** *Source:* [http://www.phidgets.com/wiki/index.php?title=1067\\_User\\_Guide](http://www.phidgets.com/wiki/index.php?title=1067_User_Guide) *Contributors:* Mparadis

## Image Sources, Licenses and Contributors

**Image:1067.jpg** *Source:* <http://www.phidgets.com/wiki/index.php?title=File:1067.jpg> *License:* unknown *Contributors:* Mparadis

**File:1067\_1\_Connecting\_The\_Hardware.jpg** *Source:* [http://www.phidgets.com/wiki/index.php?title=File:1067\\_1\\_Connecting\\_The\\_Hardware.jpg](http://www.phidgets.com/wiki/index.php?title=File:1067_1_Connecting_The_Hardware.jpg) *License:* unknown *Contributors:* Mparadis

**File:Ph.jpg** *Source:* <http://www.phidgets.com/wiki/index.php?title=File:Ph.jpg> *License:* unknown *Contributors:* Mparadis

**File:1067\_0\_Control\_Panel\_Screen.jpg** *Source:* [http://www.phidgets.com/wiki/index.php?title=File:1067\\_0\\_Control\\_Panel\\_Screen.jpg](http://www.phidgets.com/wiki/index.php?title=File:1067_0_Control_Panel_Screen.jpg) *License:* unknown *Contributors:* Mparadis

**File:1067\_0\_Stepper\_screen.jpg** *Source:* [http://www.phidgets.com/wiki/index.php?title=File:1067\\_0\\_Stepper\\_screen.jpg](http://www.phidgets.com/wiki/index.php?title=File:1067_0_Stepper_screen.jpg) *License:* unknown *Contributors:* Mparadis

**File:1063\_1\_Motor\_Types\_4wire.jpg** *Source:* [http://www.phidgets.com/wiki/index.php?title=File:1063\\_1\\_Motor\\_Types\\_4wire.jpg](http://www.phidgets.com/wiki/index.php?title=File:1063_1_Motor_Types_4wire.jpg) *License:* unknown *Contributors:* Mparadis

**File:1062\_1\_Motor\_Types\_6wire.jpg** *Source:* [http://www.phidgets.com/wiki/index.php?title=File:1062\\_1\\_Motor\\_Types\\_6wire.jpg](http://www.phidgets.com/wiki/index.php?title=File:1062_1_Motor_Types_6wire.jpg) *License:* unknown *Contributors:* Mparadis

**File:1062\_1\_Motor\_Types\_8wire.jpg** *Source:* [http://www.phidgets.com/wiki/index.php?title=File:1062\\_1\\_Motor\\_Types\\_8wire.jpg](http://www.phidgets.com/wiki/index.php?title=File:1062_1_Motor_Types_8wire.jpg) *License:* unknown *Contributors:* Mparadis

**File:1063\_1\_Current\_Limiting.jpg** *Source:* [http://www.phidgets.com/wiki/index.php?title=File:1063\\_1\\_Current\\_Limiting.jpg](http://www.phidgets.com/wiki/index.php?title=File:1063_1_Current_Limiting.jpg) *License:* unknown *Contributors:* Mparadis