

1001 - PhidgetServo 4-Motor



Product Features

- Controls 4 Remote Control (RC) servo motors.
- The motor at position 0 is powered directly from the USB port
- Motors 1, 2, and 3 are powered by an external power supply.
- Step accuracy of 0.1 degrees
- Requires a 6 to 12VDC external power supply.
- Connects directly to a computer's USB port.

Programming Environment

Operating Systems: Windows 2000/XP/Vista, 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.

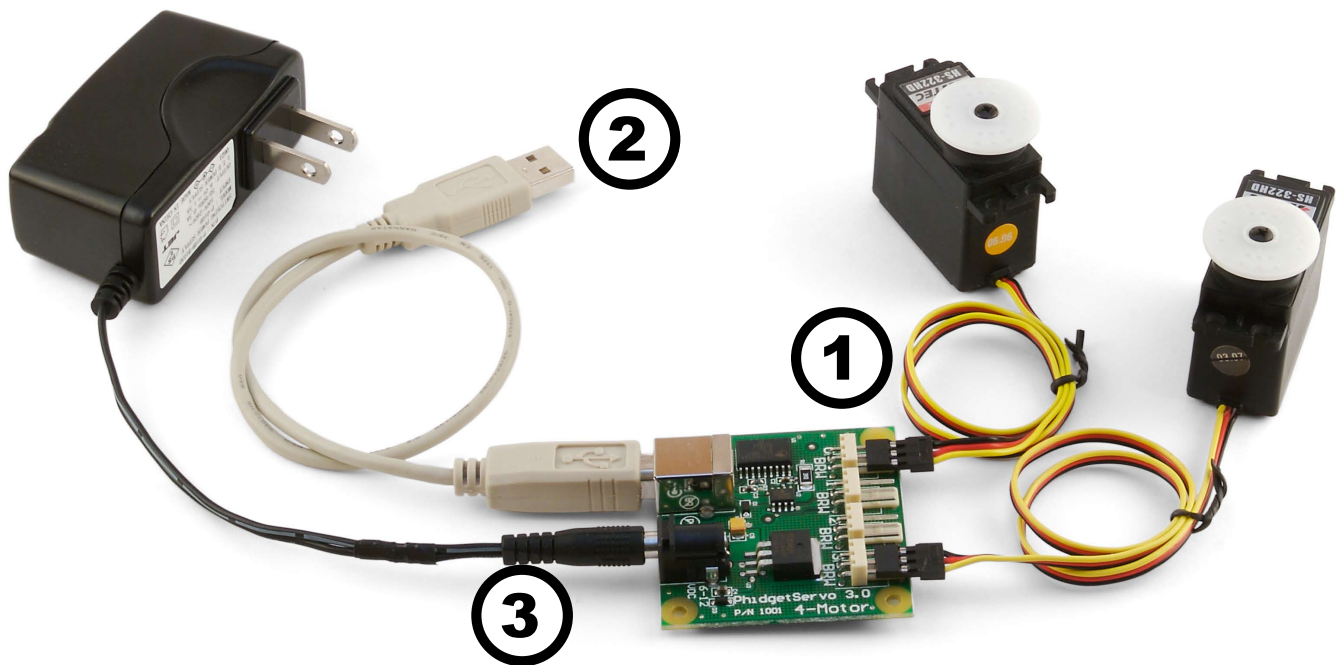
Installing the hardware

The kit contains:

- A PhidgetServo 4-Motor
- A USB cable

You will also need:

- 2 servo motors
- A 6 to 12VDC power supply,




1. Attach the connector from the servo motor onto the PhidgetServo board. The board is labeled with B R W (Black Red White) to match the wire colors from servo motors. If you connect it backwards, it will not work! Many servo motors have a yellow wire instead of a white wire. Use the board connector labelled 0 for the first motor and the connector labelled 4 for the second one.
2. Connect the PhidgetServo to your PC using the USB cable.
3. Connect the power supply to the PhidgetServo using the barrel connector.

Downloading and Installing the software

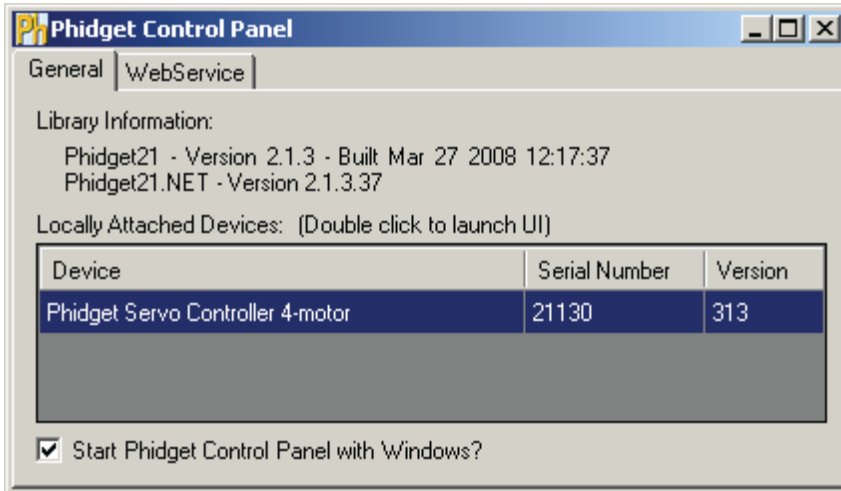
If you are using Windows 2000/XP/Vista


Go to www.phidgets.com >> Downloads >> Windows

Download and run Phidget21.MSI

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

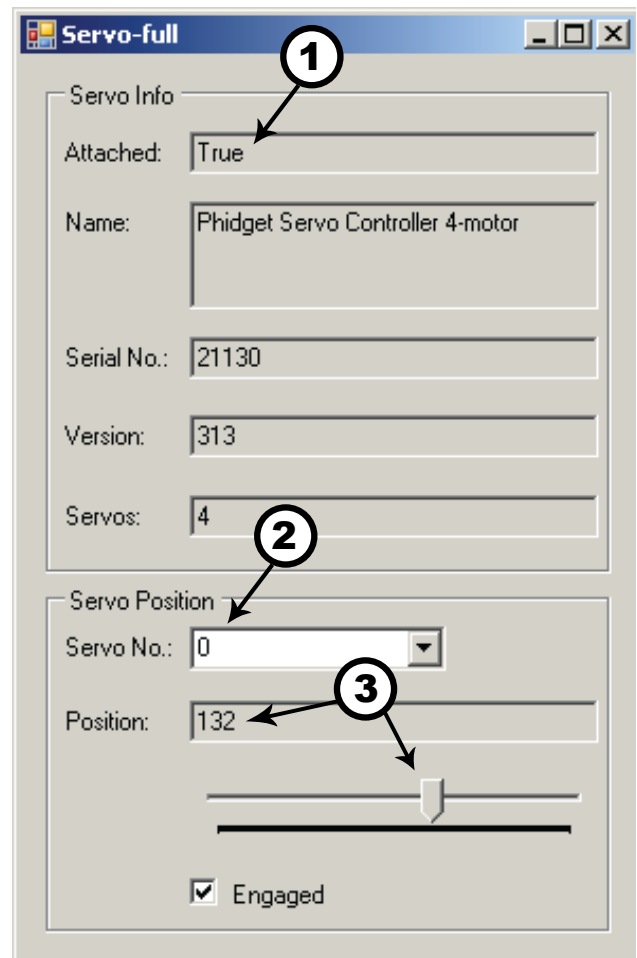
Testing the PhidgetServo 4-Motor Functionality



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

1. Double Click on **Phidget Servo Controller 4-motor** in the Phidget Control Panel to bring up Servo-full and check that the box labelled Attached contains the word True.
2. Select Servo motor 0.
3. Move the slider to make the motor turn. The motor position is displayed in the box above the Slider.

Note: To check the second motor just switch the Servo No to 4 (step 2) and redo step 3.

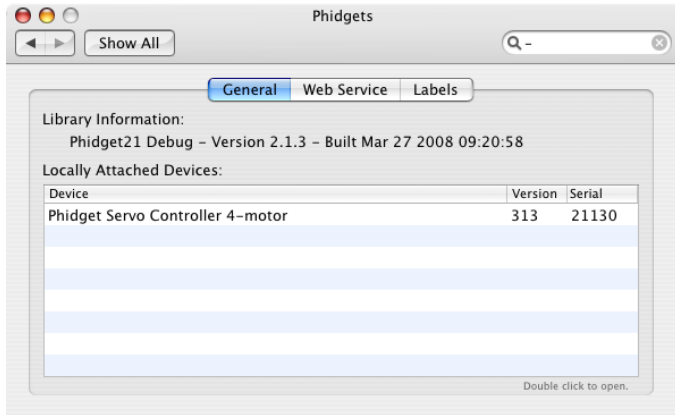


If you are using Mac OS X

Go to www.phidgets.com >> downloads >> Mac

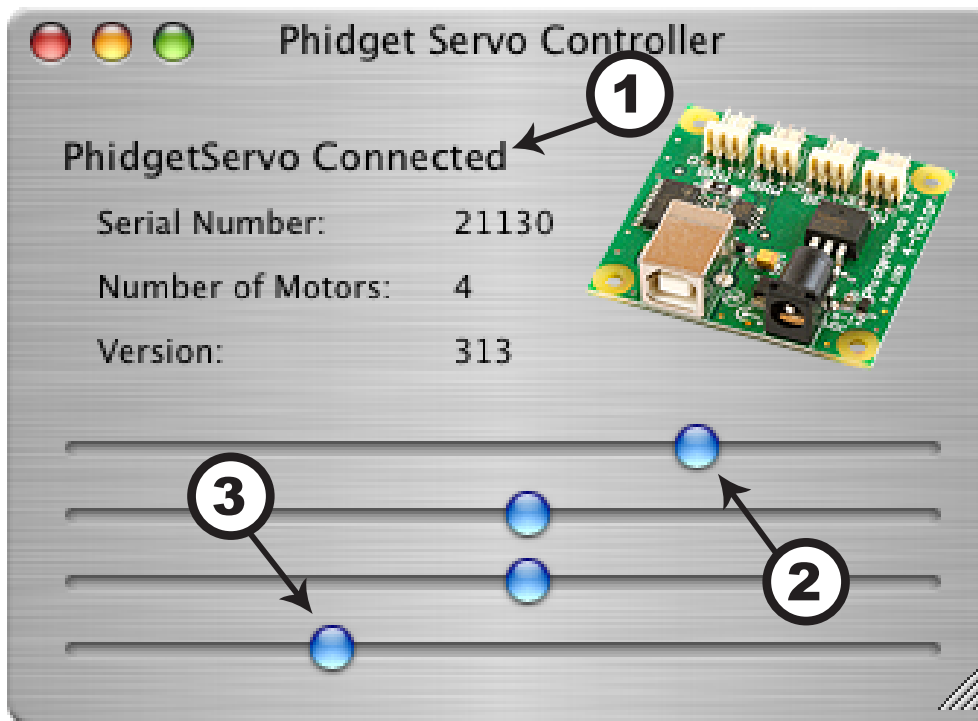
Download Mac OS X Framework

Testing the PhidgetServo 4-Motor functionality



Click on System Preferences >> Phidgets (under Other) to activate the Phidgets Preference Pane. Make sure that the **Phidget Servo Controller 4-motor** is properly attached.

1. Double Click on **Phidget Servo Controller 4-motor** in the Phidget Preference Pane to bring up the Phidget Servo Controller Example and check that the Phidget Servo Controller is attached.



2. Move the slider to make the first motor turn.
3. Move the slider to make the second motor turn.

If you are using Linux

Go to www.phidgets.com >> Downloads >> Linux

- Download Linux Source
- Have a look at the readme file
- Build Phidget21

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

Note: 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.

Note: 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 >> Downloads >> Windows Mobile/CE

Download x86 or ARMV4I, 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 computers 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.

API documentation

We maintain API manuals for COM (Windows), C (Windows/Mac OSX/Linux), Action Script, .Net and Java. Look at the section that corresponds to the Phidget you are using. These manuals can be accessed in different ways:

Using Downloads on main menu

Click on Downloads >> Operating System (i.e. Windows) >> Platform (i.e. C#) >> API Document (i.e. Net API Manual)

Using Products on Home Page

Click on InterfaceKits (under Products) >> 1018 PhidgetInterfaceKit 8/8/8 >> API Manual (Under Software Information)

Using Information on Home Page

Click on Information (under Main Menu) >> Your API Manual (under Phidgets API Manuals)

Examples

We have written examples to illustrate how the APIs are used. Examples for the C#.NET programming language include .exe files for each of the examples in the directory root.

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.

To get the examples, go to www.phidgets.com and click on Downloads. Under Step 2: click on your platform of choice and click on the File Name besides Examples.

Support

- Click on Live Support on www.phidgets.com to chat with our support desk experts
- Call the support desk at 1.403.282.7335 8:00 AM to 5:00 PM Mountain Time (US & Canada) - GMT-07:00
- E-mail sales@phidgets.com

Simple example written in C#

```
/* - Servo simple -
*****
* This simple example sets up a Servo objectm hooks the event handlers and opens it for
* device connections. Once a Servo is attached with a motor in motor 0 it will simulate
* moving the motor from position 15 to 231, displaying the event details to the console.
* For a more detailed example, see the Servo-full example.
*
* Please note that this example was designed to work with only one Phidget Servo
* connected. For an example using multiple Phidget Servos, please see a "multiple"
* example in the Servos Examples folder.
*
* Copyright 2007 Phidgets Inc.
* This work is licensed under the Creative Commons Attribution 2.5 Canada License.
* To view a copy of this license, visit http://creativecommons.org/licenses/by/2.5/ca/
*/
using System;
using System.Collections.Generic;
using System.Text;
//Needed for the Servo class, Phidget base classes, and the PhidgetException class
using Phidgets;
//Needed for the Phidget event handling classes
using Phidgets.Events;
//Using this simply for the sleep() method so that the for loop will wait for the motor
//to finish moving to the previous new position before setting a new position
using System.Threading;

namespace Servo_simple
{
    class Program
    {
        static void Main(string[] args)
        {
            try
            {
                //Declare a Servo object
                Servo servo = new Servo();

                //Hook the basic event handlers
                servo.Attach += new AttachEventHandler(servo_Attach);
                servo.Detach += new DetachEventHandler(servo_Detach);
                servo.Error += new ErrorHandler(servo_Error);

                //hook the phidget specific event handlers
                servo.PositionChange += new PositionChangeEventHandler
                    (servo_PositionChange);

                //open the Servo object for device connections
                servo.open();

                //Get the program to wait for a Servo to be attached
                Console.WriteLine("Waiting for Servo to be attached...");
                servo.waitForAttachment();

                //Set the initial position for the servo. I set it to 15 here just
                //since I am going to cycle through the positive values to show a basic
                //full movement
                if (servo.Attached)
```



```

    {
        servo.servos[0].Position = 15.00;
    }
    double i;

    //Wait for the motor to finish getting into position and let the user
    //continue
    Console.WriteLine("Press any key to continue...");
    Console.Read();

    //Move the motor from position value 15 to 231m we sleep for
    //10 milliseconds between each step to give the motor enough time to
    //move to the set position
    if (servo.Attached)
    {
        for (i = 15.00; i < 232.00; i++)
        {
            Thread.Sleep(10);
            servo.servos[0].Position = i;
        }
    }

    //Wait for the events to fire and display, user input will continue the
    //program
    Console.WriteLine("Press any key to end...");
    Console.Read();

    //user input was read so we can terminate the program now, close the
    //Servo object
    servo.close();

    //set the object to null to get it out of memory
    servo = null;

    //if no exceptions were thrown at this point it is safe to terminate
    Console.WriteLine("ok");
}
catch (PhidgetException ex)
{
    Console.WriteLine(ex.Description);
}
}

//Attach event handler...Display te serial number of the attached servo device
static void servo_Attach(object sender, AttachEventArgs e)
{
    Console.WriteLine("Servo {0} attached!", e.Device.SerialNumber.ToString());
}

//Detach event handler...Display the serial number of the detached servo device
static void servo_Detach(object sender, DetachEventArgs e)
{
    Console.WriteLine("Servo {0} detached!", e.Device.SerialNumber.ToString());
}

//Error event handler....Display the error description to the console
static void servo_Error(object sender, ErrorEventArgs e)
{

```

```

        Console.WriteLine(e.Description);
    }

    //Position CHange event handler...display which motor changed position and
    //its new position value to the console
    static void servo_PositionChange(object sender, PositionChangeEventArgs e)
    {
        Console.WriteLine("Servo {0} Position {1}", e.Index, e.Position);
    }
}
}

```

Technical Section



Servo Motors

Servos are motors that are typically used when shaft position needs to be controlled. Internally, a servo motor's shaft is mechanically connected to a potentiometer; this tells the motor's integrated electronics the present position of the shaft. A pulse-code-modulated signal sent from the PhidgetServo on the control wire tells the motor the desired position of the shaft, which is set in software. The motor is then powered until the current-position and desired-position match.

Pulse Code Modulation (PCM)

A PCM signal has a defined period (typically 20ms in servo applications) and a specified ON- and OFF-time. The ON-time is the amount of time during the period that the signal is at 5V; the rest of the time the signal is at 0V (the OFF-time). When using PCM with servo motors, one specific duration of ON-time will represent the minimum shaft position, and a different and longer duration ON-time will represent the maximum shaft position. The ON-time in between these two bounds is the setting where the shaft-position is centered. These values are defined by the motor manufacturer and vary between servo motors.

Using the PhidgetServo with a Servo Motor

The PhidgetServo has been designed to be used with a variety of RC servo motors independent of the motor-specific position, velocity and torque limits. Select a motor that suits your application and falls within the PhidgetServo device specifications (see page 9). To use a servo motor, simply set the desired shaft position in software. It should be noted that the PhidgetServo can not sense the actual position of a servo on its own. The servo motor at position 0 is powered by the USB bus from the PC; an external power supply is only required to power motors as positions 1, 2, and 3.

Calculating Servo Motor Pulse Codes

The width of the pulse sent from the PhidgetServo to the motor translates to an angular position of the motor shaft. In many cases, a pulse width of 1.5 milliseconds signifies the center position of the shaft; the upper and lower timing bounds depend on the manufacturer and model of the servo used. The timing of the pulse code can be calculated with the following formula:

$$\text{Pulse Width (in microseconds)} = (\text{Software MotorPosition} + 23) * 10.6$$

Servo Motor Gear Slop

Although the PhidgetServo can position to an accuracy of 0.1 degrees, the repeatability of positioning the shaft of a servo motor is affected by the servo motor's *gear slop*. Gear slop is the amount of play between the interlocking teeth of gears within the servo motor. More expensive servo motors, built with precision gears, will have a smaller amount of gear slop, while cheaper RC servo motors constructed with plastic gears may have one degree or more.

Using the PhidgetServo with Continuous Rotation Servos

A continuous rotation servo is a servo motor that has had its headgear-stop removed and potentiometer replaced by two matched-value resistors. This has the effect of allowing the motor to rotate freely through a full range of motion, but disables the motor's ability to control shaft position.

When using the PhidgetServo with a servo motor modified in this way, position control in software becomes the motor's speed control. Because the two resistors that replace the motor's potentiometer are matched in value, the motor will always think its shaft is at center position. If the target position in software is set to center, the motor will believe it has achieved the target and will therefore not rotate. The further away from center the target position is set to, the faster the motor will rotate (trying to reach that position, but never doing so). Changing the value above or below center changes the direction of rotation.

Using the PhidgetServo with PCM-to-DC Motor Controllers

Some DC motor controllers accept a servo motor PCM signal as valid input, and use the signal to control the speed of a DC motor. Examples of these include Victor and Thor series motor controllers from IFI Robotics. Operation of these are similar to the way the PhidgetServo is used to control continuous rotation servos, however DC motors with much higher voltage/current ratings can be driven. Note: a buffer on the control line is sometimes required when interfacing to these types of motor controllers, and can typically be purchased from the motor controller manufacturer.

RC Servo Motors

The PhidgetServo 4-Motor will work with a variety of small to medium sized 3-wire servo motors. A few motors are listed below.

Manufacturer	Part Number	Description
Hitec	HS-55	Feather Series RC Servo Motor
Hitec	HS-322HD	Deluxe Series RC Servo Motor (shown)
Hitec	HS-805BB	Mega Quarter Scale RC Servo Motor

The Hitech HS-322HD is available for purchase at www.phidgets.com. Many RC servo motors are available directly from manufacturers like Hitec or at local distributors.

API Section

We document API Calls specific to the 1001 Phidgetservo 4-Motor. Functions common to all Phidgets are not covered here. This section is deliberately generic - for calling conventions in a specific language, refer to that languages' API manual.

Functions

int MotorCount() [get] : Constant = 4

Returns the number of servos that can be controlled by this PhidgetServo. Note that there is no way of determining the number of servos actually attached.

double Position(int ServoIndex) [get,set] : Degrees

Sets/returns the desired servo motor position for a particular servo motor.

Note that reading Position will not tell you where the servo really is. RC Servos are open loop – the PhidgetServo can command them to travel to a position, but there is no feedback available for if they arrived, or their position.

If the servo is not engaged, the position is unknown and calling this function will throw an exception.

The range is between PositionMin and PositionMax, and corresponds approximately to an angle in degrees. Note that most servos will not be able to operate across this entire range. Typically, the range might be 25 - 180 degrees, but this depends on the servo.

On the 1001 PhidgetServo 4-Motor, 0 is the servo motor powered by USB, and servos 1-3 are powered by an external power supply.

double PositionMax(int ServoIndex) [get] : Constant

Returns the maximum position that the PhidgetServo will accept, or return.

double PositionMin(int ServoIndex) [get] : Constant

Returns the minimum position that a PhidgetServo will accept, or return.

bool Engaged(int ServoIndex) [get,set]

If Engaged is set to false, no PWM signals will be sent to the servo. This engages or disengages the servo. The motor is engaged whenever you set a position, or use this function to disengage and reengage without setting a position.

Events

OnPositionChange(int ServoIndex, double Position) [event]

An event that is issued whenever the position of a PhidgetServo changes.

Device Specifications

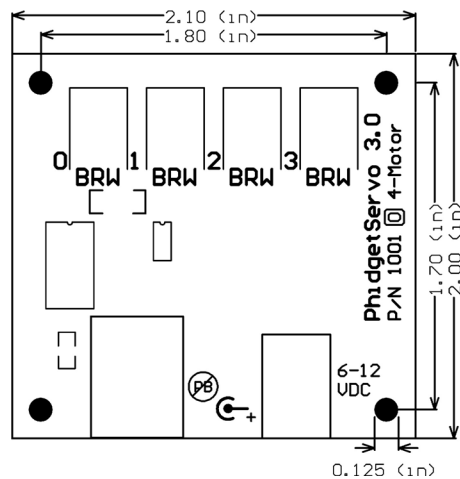
Pulse Code Period	20 ms
Minimum Pulse Width	10 us
Maximum Pulse Width	2.55 ms
Time Resolution	1 us
Output Controller Update Rate	50 updates / second
Output Impedance (control)	600 Ohms
Lower Position Limit	- 23.00°
Upper Position Limit	232.99°
Operating Motor Voltage	5.0 V
External Power Supply Voltage	6 VDC - 12 VDC
External Power Current Consumption	1500 mA max (500 mA / motor)
USB-Power Current Specification	500 mA max
Device Quiescent Current Consumption	13 mA
Device Active Current Consumption	500 mA max

Hitech HS-322HD RC Servo Specifications

Torque @ 4.8V	41.66 oz.in
Speed @ 4.8V	190ms/60°
Size L x W x H	1.57" x 0.78" x 1.43"
Weight	1.51 oz.

Mechanical Drawing

1:1 scale



Product History

Date	Product Revision	Comment
June 2001	DeviceVersion200	1 Degree Position Resolution
June 2002	DeviceVersion300	0.1 Degree Position Resolution
January 2004	DeviceVersion313	State Echoing Added