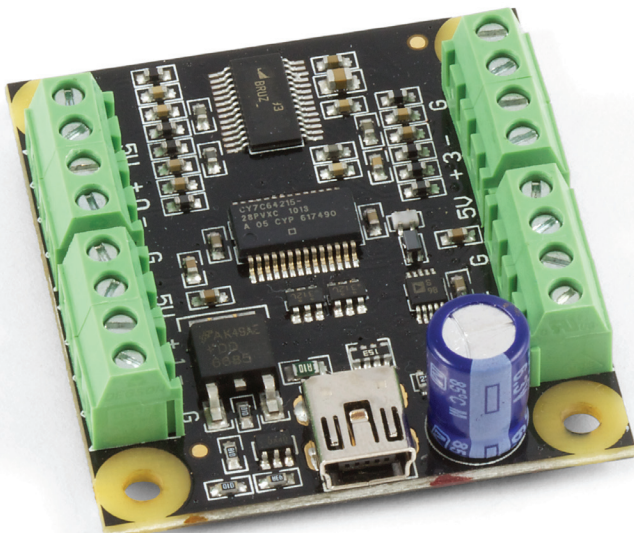


Product Manual

1046 - PhidgetBridge 4-Input



Phidgets 1046 - Product Manual
For Board Revision 0
© Phidgets Inc. 2011

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/7
 - 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 PhidgetBridge 4-input
 - 10 Properties
 - 11 Events

12 Technical Section

- 12 How to calibrate the bridge
- 12 Gain Setting vs Resolution
- 12 Connecting your strain gauge/load cell
- 12 Things to watch for
- 13 Changing the Data Rate

13 Device Specifications

13 Product History

13 Support

Product Features

- Interfaces to up to 4 un-amplified Wheatstone bridges
- Supports specific data rates (from 1 to 125 samples/sec)
- Supports gains from 1 - 128.
- Software configurable amplification
- Great interface for Load Cells, Strain Gauges, Pressure Sensors/Barometers, Magneto-resistive sensors (Compasses)

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 >> Programming.

Connection

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

Getting Started

Checking the Contents

You should have received:

- A PhidgetBridge 4-Inputs
- A Mini-USB cable

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

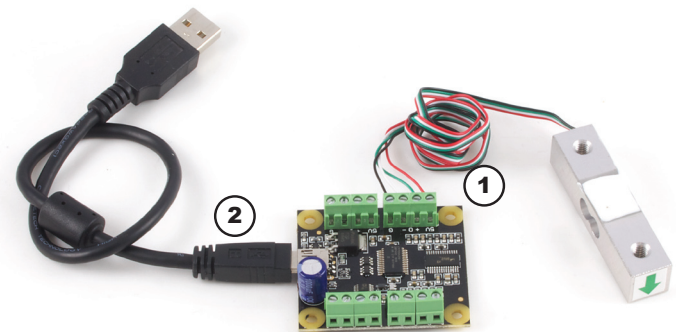
- A wheatstone bridge based sensor

Connecting all the pieces

1. Connect the load cell to the PhidgetBridge - use bridge 0. We are using a 3133 - Micro Load Cell (0-5kg) - CZL635. Connect the red wire to 5V, the green wire to +, the white wire to -, and the black wire to G.

If your Load Cell is not documented, refer to the technical section of this manual for instructions on how to connect it.

2. Connect the PhidgetBridge to your computer using the Mini-USB cable.




Testing Using Windows 2000/XP/Vista/7

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 >> Drivers


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

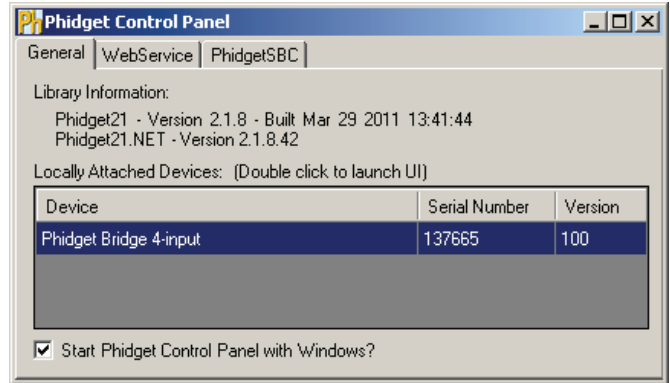
You should see the  icon on the right hand corner of the Task Bar.

Running Phidgets Sample Program

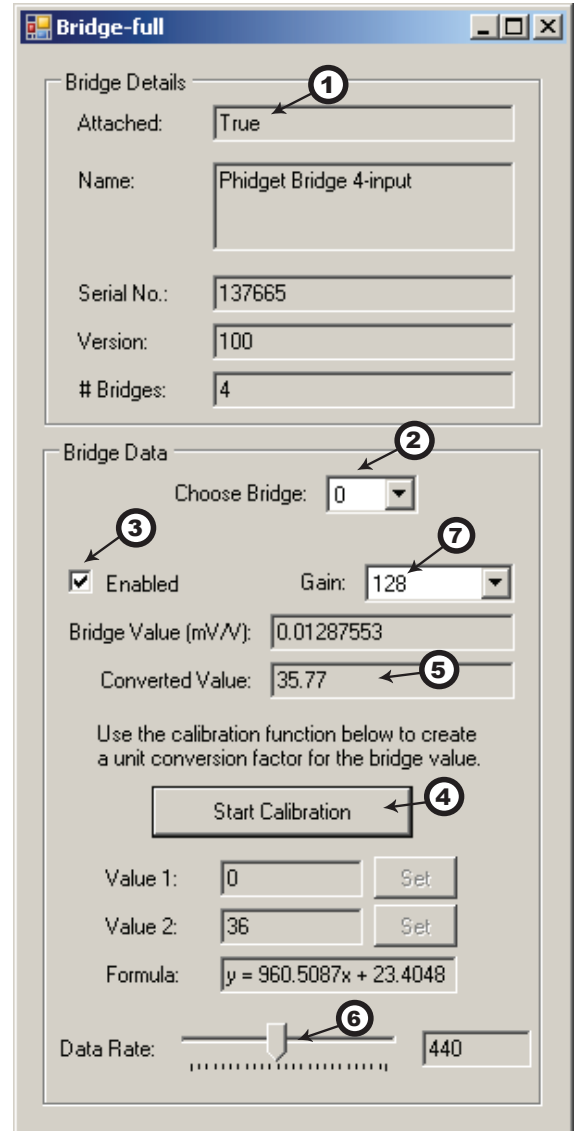
Double clicking on the  icon loads the Phidget Control Panel; we will use this program to make sure that your new Phidget works properly.

The source code for the Bridge-full sample program can be found under C# by clicking on www.phidgets.com >> Programming.

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



1. Double Click on **PhidgetBridge 4-input** in the Phidget Control Panel to bring up Bridge-full and check that the box labelled Attached contains the word True.
2. If you have connected your device to the same bridge as we did, select bridge 0.
3. Click to enable the bridge.
4. Click on the Start Calibration Button. Enter 0 for Value 1 (no weight on the load cell). Put a known weight on the load cell and enter the number in Value 2 box. The calibration formula is displayed in the formula box. If your sensor is not a load cell, you can still use the Calibration functionality - you just need two known values for your sensor.
5. Put a different weight on the load cell and the bridge value gets converted using the calibration formula.
6. You can use the slider to adjust the data rate from 8ms to 1000ms in increments of 8ms.
7. You can set the gain to 1, 8, 16, 32, 64, 128; in general a higher gain value gives you lower noise and higher resolution.



Testing Using Mac OS X

- Click on System Preferences >> Phidgets (under Other) to activate the Preference Pane
- Make sure that the PhidgetBridge 4-input is properly attached.
- Double Click on Phidget PhidgetBridge 4-input in the Phidget Preference Pane to bring up the Bridge-full Sample program. This program will function in a similar way as the Windows version.

If you are using Linux

There are no sample programs written for Linux.

Go to 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 >> 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.
- For full performance, the Phidget APIs are designed to be used in an event driven architecture. Applications that require receiving all the data streaming from the device will have to use event handlers, instead of polling.

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 >> 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 >> 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 >> 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 >> 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 PhidgetBridge 4-input

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.

Properties

int InputCount [get] : Constant = 4

Returns the number of bridges supported by this PhidgetBridge.

double BridgeValue(int index) [get]

Returns the value of the selected input, in mV/V. If the input is not enabled, this will throw an `EPHIDGET_UNKNOWNVAL` exception. If the bridge is saturated, this will be equal to `BridgeMax` or `BridgeMin` and an error event will be fired - in this case, gain should be reduced.

double BridgeMax(int index) [get]

Returns the maximum value that the selected Bridge can measure, in mV/V. This value will depend on the selected gain. At a gain of 1, `BridgeMax` == 1000mV/V.

double BridgeMin(int index) [get]

Returns the minimum value that the selected Bridge can measure, in mV/V. This value will depend on the selected gain. At a gain of 1, `BridgeMin` == -1000mV/V.

boolean Enabled(int index) [get,set]

Gets / Sets the enabled state of a Bridge. This applies power between +5v and Ground and starts measuring the differential on the +/- pins. By default, all Bridges are disabled, and need to be explicitly enabled on startup.

Gains Gain(int index) [get,set]

Gets / Sets the gain for a selected bridge. Supported gains are 1, 8, 16, 32, 64 and 128. Note that increasing the gains will reduce the measurable voltage difference by the gain factor, with +-1000mV/V being the maximum, with no gain.

int DataRate [get,set]

Gets / Sets the data rate, in ms. Data rate applies to all 4 bridges simultaneously. Setting a slower data rate will reduce noise at the cost of sample time. Also note that each bridge is being sampled only 1/4 of the time - this is probably ok for very stable signals, but for changing signals, it's may be best to set a higher sampling rate and do averaging in software.

Data rate must be a multiple of 8ms. Trying to set something between multiples of 8 will cause an EPHIDGET_INVALIDARG exception to be thrown.

int DataRateMax [get] : Constant = 8

Gets the maximum supported data rate, in ms.

int DataRateMin [get] : Constant = 1000

Gets the minimum supported data rate, in ms.

Events

OnBridgeData(int index, double value) [event]

An event that is issued at the specified DataRate, for each enabled bridge. Value is the bridgeValue, in mV/V.

OnError(int ErrorCode, String ErrorDescription)

The PhidgetBridge will throw error events under certain circumstances:

ErrorCode = EEPHIDGET_OUTOFRANGE - A bridge input has gone out of range. This indicates either an overrange or underrange condition. If possible, gain should be reduced.

See the ErrorDescription string for specific error details.

Technical Section

How to calibrate the bridge

We have observed a 1.5% difference between gain=1 and gain=8. This may require that each system (PhidgetBridge + sensors) are calibrated as a whole. For maximum accuracy, decide on and keep with a chosen gain before calibrating the system.

Expensive sensors will ship with a certificate of calibration specifying, often in mV/V, how the sensor responds to stimulus. Less expensive will have to be calibrated, which requires having at least two points where you know accurately what is being measured. In the case of weight measurement, this would be a known force or weight. Record the output from the PhidgetBridge at one known point, and at a second known point. It helps if the two values are reasonably far apart. Use the values to make a linear equation to convert the PhidgetBridge output in mV/V (called X) to the appropriate unit you are measuring (called Y). Two calibration coefficients (a,b) set the slope and offset for the calibration: $(Y = aX + b)$. It's possible to use more than two points, if available.

The C# Bridge-full example shows how to do a 2-point calibration and apply the coefficients programmatically.

Gain Setting vs Resolution

We report the measured voltage in a ratiometric unit known as mV/V. This is how the maximum range of sensors that use strain gauges is usually specified. mV/V is the output value in mV of the measured sensor, scaled for a 1V sensor supply voltage. This value will correspond to the physical quantity that the sensor is measuring, regardless of the actual voltage supplied to the sensor.

Gain	Resolution	Range
1	119nV/V	+/-1000mV/V
8	14.9nV/V	+/-125mV/V
16	7.45nV/V	+/-62.5mV/V
32	3.72nV/V	+/-31.25mV/V
64	1.86nV/V	+/-15.625mV/V
128	0.93nV/V	+/-7.8125mV/V

When choosing the Gain setting, it's best to use the highest gain possible that can still measure the full range of your sensor. For an individual unit, you can apply the maximum stimulus to the sensor, and ensure the BridgeValue reported is well within the range for the Gain setting you have chosen. If many units are being deployed, it's best to consult the data sheet for the strain gauge and look for maximum offset.

Some wheatstone bridges - most often those produced from silicon and used in pressure sensors, will have a very wide offset, and large manufacturing variation in the offset. This will restrict the gain to lower settings, particularly if the application must support a number of deployed systems with the expected variation. Fortunately, the very high precision electronics used in the PhidgetBridge means that in many application, higher gain is not necessary to get adequate accuracy and resolution.

Connecting your strain gauge/load cell

If no documentation is available for your strain gauge, it's possible to use a multimeter to determine how to connect it, provided there are no electronics in the sensor. First, measure resistance between the 4 wires. There are 6 combinations - two combinations will have a resistance 20-40% higher than the other four. Choose one of these high-resistance combinations, and wire it into 5V and G on the PhidgetBridge. Connect the other two wires into +/- . Apply a load - if the mV/V responds in the opposite way to your expectations, flip the +/- wires.

Things to watch for

The PhidgetBridge is designed to measure voltages as a ratio of the supply voltage - it's not practical to make measurements of absolute voltages with this product.

For maximum accuracy, all wires from the PhidgetBridge to the sensor should be the same length and thickness. Changes in temperature will change the resistance of the wires - if they are all the same, the errors will cancel out.

Changing the Data Rate

Using a slower sampling rate will reduce the noise in the measurements dramatically. The noise figures are specific to individual applications and sensors. The lowest noise level achievable is 5nV/V RMS.

Other

Each bridge input can be powered down, reducing power consumption with Bridge-Sensors, and useful for reducing heating of sensors, which can introduce errors.

Device Specifications

Characteristic	Value
USB Voltage	4.5 - 5.25VDC
USB Current	500mA
USB Quiescent Current	35mA
Total Current available to Bridge Outputs	465mA
Recommended wire size	16 - 26AWG
Differential Voltage resolution per channel	24 bits
Data Rates (affects all channels)	8ms to 1000ms in 8ms increments
Gain Settings (affects all channels)	1, 8, 16, 24, 64, 128
Input Current (Max)	+/-3nA
Operational Input Voltage Range	GND + 0.25V to 5V Supply - 0.25V
Operating Temperature	0 - 70°C

Product History

Date	Board Revision	Device Version	Comment
May 2011	0	100	Product Release

Support

Call the support desk at 1.403.282.7335 9:00 AM to 5:00 PM Mountain Time (US & Canada) - GMT-07:00

or

E-mail us at: support@phidgets.com