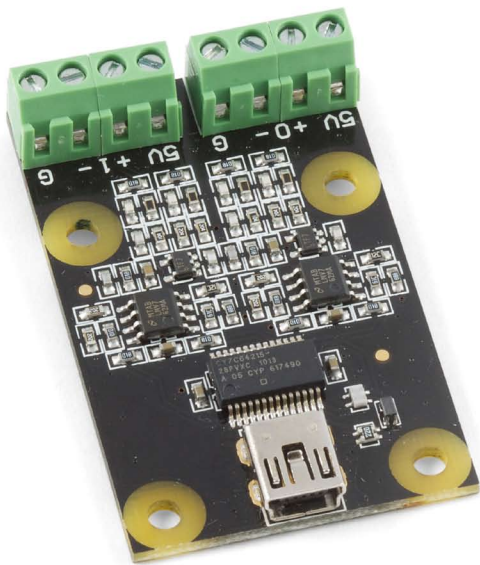


Product Manual

1054 - PhidgetFrequencyCounter



Phidgets 1054 - 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 PhidgetFrequencyCounter
 - 10 Enums
 - 10 Properties
 - 11 Functions
 - 11 Events

12 Technical Section

- 12 Logic-Level Frequencies
- 12 Zero-Centered Frequencies
- 12 Differential Inputs

12 Output Voltage

13 Device Specifications

13 Product History

13 Support

Product Features

The 1054 Frequency Counter is a device designed to count events from an analog signal over time and calculate a frequency from it. The product can count a logic level or signal centered around zero volts. Signals with a different ground can be counted, provided they are within the common mode range ($\pm 10V$). It can power small devices, such as Hall Effect or flow rate sensors, tachometers, and other sensors.

- Allows you to measure the frequency of an analog or digital signal.
- Can power small isolated devices such as Hall Effect or flow rate sensors, tachometers, ...
- Can measure frequencies from 0 to 1 MHz.
- Provides a 5VDC power supply for your connected devices.
- Can measure zero-centered signals with a minimum amplitude of 110mV peak-to-peak
- Can measure logic-levels (3.3V and 5V) with a maximum difference in the ground voltage of $\pm 10V$
- Connects directly to a computer's USB port.

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:

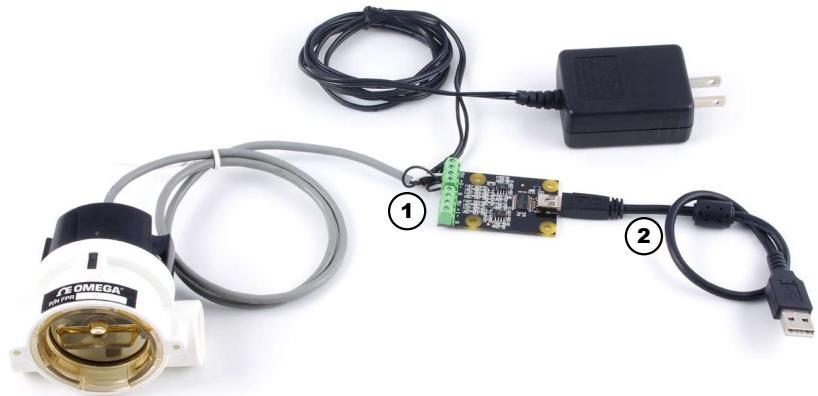
- A PhidgetFrequencyCounter
- A Mini-USB cable

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

- A Frequency producing device such as a flow meter or tachometer

Connecting all the pieces

1. Connect your sensor to the PhidgetFrequencyCounter channel 0. We are using a Flow Meter that requires an external +12V power supply. The ground of the power supply, sensor and 1054 are all connected together. If your sensor requires +5V, it can be powered directly by the 1054.
2. Connect the PhidgetFrequencyCounter 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)

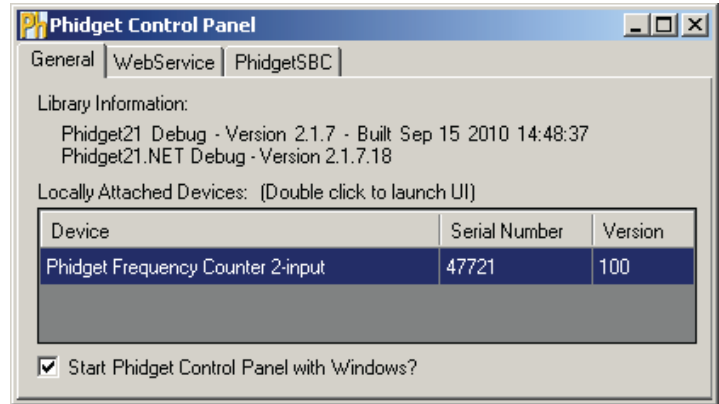
You should see the  icon on the right hand corner of the Task Bar.

Running Phidgets Sample Program

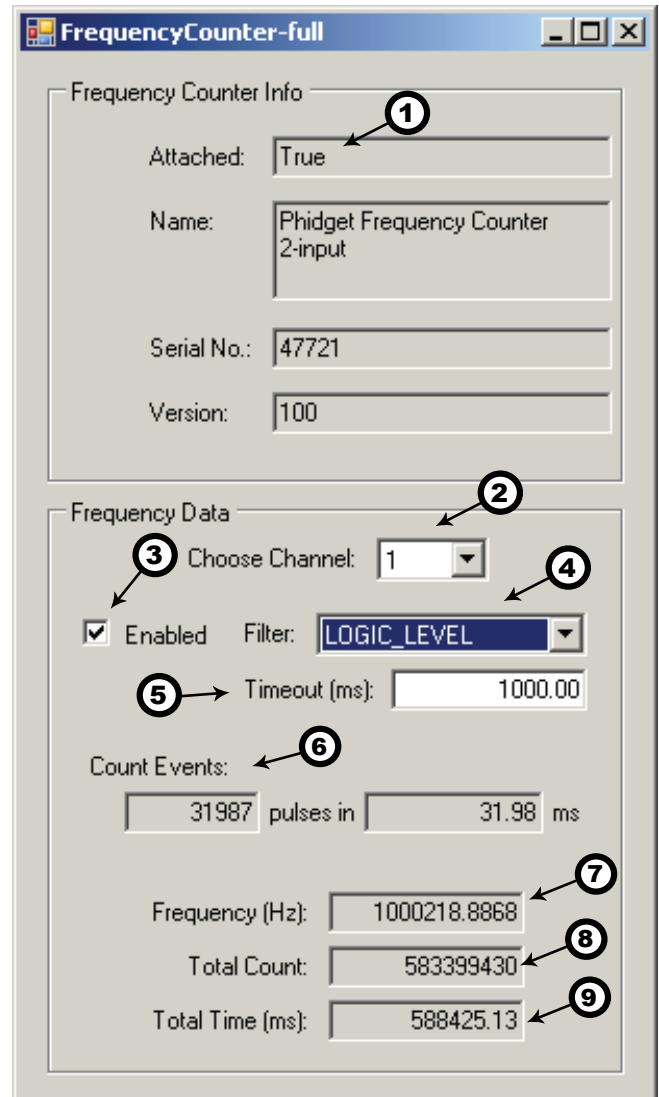
Double clicking on the  icon loads the Phidget Control Panel; we will use this program to make sure that your new Phidget works properly.

The source code for the FrequencyCounter-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 **PhidgetFrequencyCounter** is properly attached to your PC.



1. Double Click on **PhidgetFrequencyCounter** in the Phidget Control Panel to bring up FrequencyCounter-full and check that the box labelled Attached contains the word True.
2. Select channel 0.
3. Click in the box to enable the device.
4. Select Logic Level or Zero Crossing, depending on your sensor. See the Technical Section for help.
5. The Timeout default of 1000 ms is fine for most applications.
6. Count Events shows the number of pulses in the last measurement interval.
7. Frequency is calculated from the number of pulses in the last measurement interval.
8. Total count is the total number of pulses counted since the application started or the 1054 reset.
9. Total Time is the elapsed time since the application started or the 1054 reset.



Testing Using Mac OS X

- Click on System Preferences >> Phidgets (under Other) to activate the Preference Pane
- Make sure that the PHidgetFrequencyCounter is properly attached.
- Double Click on Phidget PhidgetFrequencyCounter in the Phidget Preference Pane to bring up the FrequencyCounter-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

There is no Control Panel written for Linux, but there are C/C++ and Java code samples available for all Phidgets which will compile and run on Linux without modification.

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 PhidgetFrequencyCounter

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.

Enums

```
enum {  
    ZERO_CROSSING = 1,  
    LOGIC_LEVEL  
} FilterType
```

Determines the signal type that the PhidgetFrequencyCounter responds to.

Properties

double Frequency(int ChannelIndex) [get]

Gets the last calculated frequency on the specified channel, in Hz. This function will return 0 if The Timeout value elapses without detecting a signal. Frequency is recalculated up to 31.25 times a second, depending on the pulse rate.

int64 TotalCount(int ChannelIndex) [get]

Gets the total number of pulses detected on the specified channel since the Phidget was opened, or since the last reset.

int64 TotalTime(int ChannelIndex) [get]

Gets the total elapsed time since Phidget was opened, or since the last reset, in microseconds. This property complements the TotalCount property.

int Timeout[get,set]

Gets or set the Timeout value, in microseconds. This value is used to set the time to wait without detecting a signal before reporting 0 Hz. The valid range is 0.1 - 100 seconds (100,000 - 100,000,000 microseconds). 1/Timeout represents the lowest frequency that will be measurable.

int Filter(int ChannelIndex, FilterType filter) [get, set]

Gets or set the channel filter mode. This controls the type of signal that the frequency counter will respond to - either a zero-centered signal, or a logic level signal.

bool Enabled(int ChannelIndex) [get, set]

Gets or sets the enabled state on the specified channel. When a channel is disabled, it will no longer register counts. TotalTime and TotalCount properties will not be incremented until the channel is re-enabled.

Functions**void reset(int ChannelIndex)**

Resets the TotalCount and TotalTime counters to 0 for the specified channel. For best precision/reliability, this should be called when the channel is disabled.

Events**Count(int ChannelIndex, int time, int counts) [event]**

An event that is issued whenever some counts have been detected. This event will fire at up to 31.25 times a second, depending on the pulse rate. The time is in microseconds and represents the amount of time in which the number of counts occurred. This event can be used to calculate frequency independently of the phidget21 library frequency implementation.

This event will fire with a count of 0 once, after the Timeout time has elapsed with no counts for a channel, to indicate 0Hz.

Technical Section

The PhidgetFrequencyCounter contains two channels to sense two different inputs. Each channel has two different circuits to sense for logic level-frequencies or zero-centered frequencies. The measurable frequency is accurate to 0.25% up to 1MHz. The Frequency Counter may measure frequencies past 1 Mhz, but the input voltage specifications will not hold.

The PhidgetFrequencyCounter can measure frequencies down to $\sim 0.01\text{Hz}$. However the response time of these measurements is directly related to the frequency, thus it could take 1 or 2 periods(100-200s) to detect the input frequency.

Logic-Level Frequencies

The logic-level sensing circuit has a hysteresis range from 0.9V to 2.4V. This will allow the circuit to count both 3.3V and 5V logic levels. In addition to logic-level signals, this will also accept the pulses from sensors with open collector outputs.

When the input signals are either 3.3V or 5V, the maximum sensed frequency is 1.5Mhz.

Most digital sensors that are powered from a signal positive power supply will output a logic-level frequency.

Zero-Centered Frequencies

The zero-centered sensing circuit can be used for input signals where you want to count as it crosses zero volts. A hysteresis of 30mVpp filters noise. At maximum frequency (1Mhz), a signal of 400mVPP is required for reliable counting.

A common application that uses a zero centered frequency output is a simple magnetic tachometer, which produces a sine wave around 0 volts.

Differential Inputs

The PhidgetFrequencyCounter uses differential inputs - that is, the voltage being compared is the difference between the (+) and (-) inputs. If your application has a slightly different ground from your USB ground, the common mode rejection in the 1054 will handle small differences in ground.

For many applications, the signal being measured is single ended - that is, your sensor outputs only one signal, and you can directly connect the ground of the sensor to the ground of the 1054. In this case, ensure the (-) input is tied to ground. Never allow either input to be left unconnected.

Output Voltage

USB Voltage is passed directly to the +5V terminal on the green blocks.

Device Specifications

Characteristic	Value
Number of channels	2
Operating Temperature	0 - 70°C
USB Voltage Range	4.75 - 5.25V
USB Current	42mA
Max Output Current (using +5V from terminal block)	450mA
Input Impedance	332 KOhm / 30pF
Frequency Error	0.25%
Maximum Input Frequency	1MHZ
Common mode input range vs. GND	-6.25 to +10.25V
Maximum Input Voltage (measured to ground)	±20V
Zero Centered Input Hysteresis Band	±30mV
Minimum Input Voltage @ 1Mhz, Zero-Centered Input	400mVpp
Logic Input, Logic Level 0 voltage	0.8V
Logic Input, Logic Level 1 voltage	3V

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