Quadrature Encoder Phidget

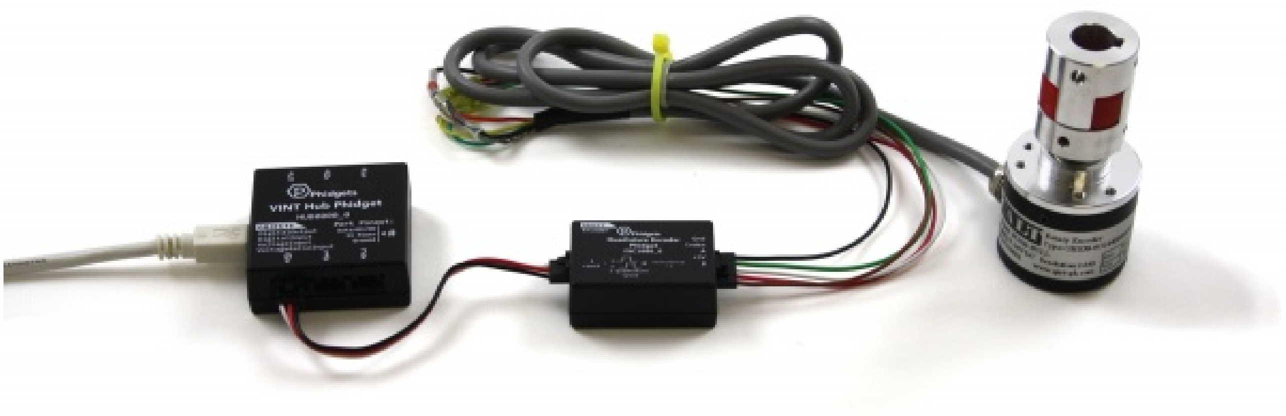
Contents

Getting Started

Welcome to the ENC1000 user guide! In order to get started, make sure you have the following hardware on hand:

- ENC1000 - Quadrature Encoder Phidget
- VINT Hub
- Phidget cable
- USB cable and computer
- Encoder

Next, you will need to connect the pieces:



1. Connect the ENC1000 to the VINT Hub using the Phidget cable.
2. Connect the encoder to the Phidget using an encoder cable.
3. Connect the VINT Hub to your computer using a USB cable.

TO TOP

Now that you have everything together, let's start using the ENC1000!

Using the ENC1000

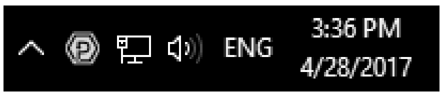
Phidget Control Panel

In order to demonstrate the functionality of the ENC1000, the Phidget Control Panel running on a Windows machine will be used.

The Phidget Control Panel is available for use on both macOS and Windows machines.

Windows

To open the Phidget Control Panel on Windows, find the icon in the taskbar. If it is not there, open up the start menu and search for Phidget Control Panel



macOS

To open the Phidget Control Panel on macOS, open Finder and navigate to the Phidget Control Panel in the Applications list. Double click on the icon to bring up the Phidget Control Panel.

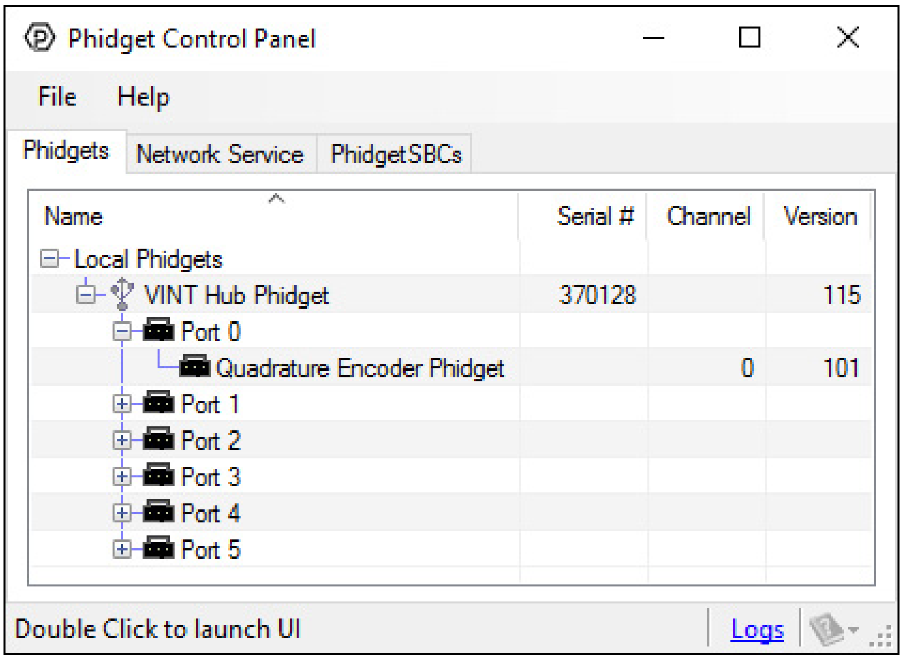
For more information, take a look at the getting started guide for your operating system:

- Getting started with Windows
- Getting started with macOS

Linux users can follow the getting started with Linux guide and continue reading here for more information about the ENC1000.

First Look

After plugging the ENC1000 into your computer and opening the Phidget Control Panel, you will see something like this:



Phidget Control Panel

Name	Serial #	Channel	Version
Local Phidgets			
VINT Hub Phidget	370128		115
Port 0			
Quadrature Encoder Phidget		0	101
Port 1			
Port 2			
Port 3			
Port 4			
Port 5			

Double Click to launch UI

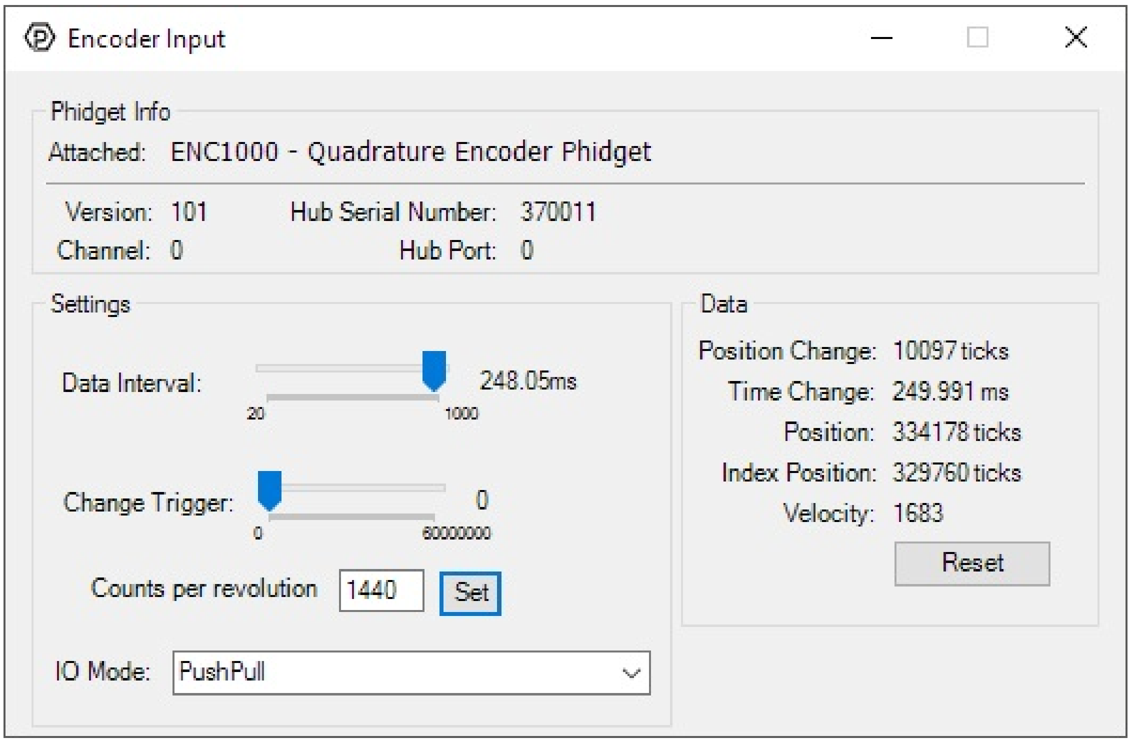
The Phidget Control Panel will list all connected Phidgets and associated objects, as well as the following information:

- **Serial number:** allows you to differentiate between similar Phidgets.
- **Channel:** allows you to differentiate between similar objects on a Phidget.
- **Version number:** corresponds to the firmware version your Phidget is running. If your Phidget is listed in red, your firmware is out of date. Update the firmware by double-clicking the entry.

The Phidget Control Panel can also be used to test your device. Double-clicking on an object will open an example.

Encoder

Double-click on the Encoder object, labelled Quadrature Encoder Phidget, in order to run the example:


Phidget Info

Attached: ENC1000 - Quadrature Encoder Phidget

Version: 101 Hub Serial Number: 370011

Channel: 0 Hub Port: 0

Settings

Data Interval: 248.05ms

Change Trigger: 0

Counts per revolution 1440

IO Mode: PushPull

Data

Position Change: 10097 ticks

Time Change: 249.991 ms

Position: 334178 ticks

Index Position: 329760 ticks

Velocity: 1683

General information about the selected object will be displayed at the top of the window. You can also experiment with the following functionality:

- **Position Change:** the number of ticks (or quadrature cycles) that have occurred since the last change event.
- **Time Change:** the amount of time in milliseconds that has elapsed since the last change event.
- **Position:** the total position in ticks relative to where the encoder was when the window was opened.
- **Index Position:** the position where the index channel was last encountered. Some encoders do not support index, check your encoder's datasheet for more information.
- **Velocity:** the average velocity in rotations per second. A CPR must be specified to enable this functionality.
- Specify a counts per revolution (CPR) value to enable velocity calculation.
- Modify the change trigger and/or data interval value by dragging the sliders. For more information on these settings, see the data interval/change trigger page.
- Modify the IO Mode with the drop-down menu. For more information on IO Mode, see the technical section.

Technical Details

General

The ENC1000 can be used with a wide assortment of mechanical and optical encoders. The encoder should be of incremental quadrature output type, indicating that there will be two output channels (usually labeled A and B).

The maximum rate of the ENC1000 is specified at 100,000 quadrature cycles per second. In your application, this number relates directly to the number of revolutions per second you wish to measure, and the number of counts per revolution specified for your encoder. If your encoder's wheel has 1000 cycles per revolution, then the limit on measurable revolutions per second is 100 (6000rpm).

Interfacing Encoders

The ENC1000 can connect to any of the encoders we sell without any modification just by setting the EncoderIOMode property to Push-Pull . If you're trying to use your own encoder, you may need to change the IO mode to Open Collector or Line Driver mode. Have a look at the Encoder Primer for more details on what to use.

Connector

The encoder input on the ENC1000 uses a 5-pin, 0.100 inch pitch locking connector. The connectors are commonly available - refer to the Table below for manufacturer part numbers.

Manufacturer	Part Number	Description
Molex	50-57-9405	5 Position Cable Connector
Molex	16-02-0102	Wire Crimp Insert for Cable Connector
Molex	70543-0004	5 Position Vertical PCB Connector
Molex	70553-0004	5 Position Right-Angle PCB Connector (Gold)
Molex	70553-0039	5 Position Right-Angle PCB Connector (Tin)
Molex	15-91-2055	5 Position Right-Angle PCB Connector - Surface Mount

Note: Most of the above components can be bought at Digikey.

Calculating Velocity

When your program captures an encoder change event, it will receive two variables: positionChange (measured in 'ticks', four of which equal one quadrature count for the ENC1000) and timeChange (measured in milliseconds). You can use these values to easily compute the instantaneous velocity of the encoder. For example, our C# encoder example implements this method of velocity calculation:

```
void enc_change(object sender, Phidget22.Events.EncoderEncoderChangeEventArgs e) {

...

// Convert time change from milliseconds to minutes
double timeChangeMinutes = e.TimeChange / 60000.0;
// Calculate RPM based on the positionChange, timeChange, and encoder CPR (specified by th
double rpm = (((double)e.PositionChange / CPR) / timeChangeMinutes);

...

}
```

This implementation may be useful if you're graphing the RPM on a line graph, but if it's being used to display the current RPM as a single number, it won't be very helpful because when the motor changes speed or direction frequently, it'll be hard to read the velocity as a meaningful value. This method can also be prone to variations in velocity if the encoder's CPR is low and the sampling rate is high. To solve these problems, you should decide on a time interval during which you'll gather data, and take a moving velocity calculation based on that data. You can use the `Queue` data type to make this easy:

```
Queue<double> positionChangeQueue = new Queue<double>();
Queue<double> timeChangeQueue = new Queue<double>();

void enc_change(object sender, Phidget22.Events.EncoderEncoderChangeEventArgs e) {

double totalPosition = 0;
double totalTime = 0;
int n = 500; // sampling window size, duration is 500*t where t is the data interval of th

// add the newest sample to the queue
positionChangeQueue.Enqueue(e.PositionChange);
timeChangeQueue.Enqueue(e.TimeChange);

// If we've exceeded our desired window size, remove the oldest element from the queue
if ( positionChangeQueue.Count >= n ) {
     positionChangeQueue.Dequeue();
     timeChangeQueue.Dequeue();
}

// Calculate totals for position and time
foreach( double positionChange in positionChangeQueue ) {

     totalPosition += positionChange;
}

foreach( double timeChange in timeChangeQueue ) {

     totalTime += timeChange;
}

// Convert time change from milliseconds to minutes
double timeChangeMinutes = e.TimeChange / 60000.0;
// Calculate RPM based on the positionChange, timeChange, and encoder CPR (specified by th
double rpm = (((double)e.PositionChange / CPR) / timeChangeMinutes);

}
```

Check out this project for even more information on encoder velocity.

What to do Next

- Software Overview - Find your preferred programming language here to learn how to write your own code with Phidgets!
- General Phidget Programming - Read this general guide to the various aspects of programming with Phidgets. Learn how to log data into a spreadsheet, use Phidgets over the network, and much more.
- Phidget22 API - The API is a universal library of all functions and definitions for programming with Phidgets. Just select your language and device and it'll give you a complete list of all properties, methods, events, and enumerations that are at your disposal.