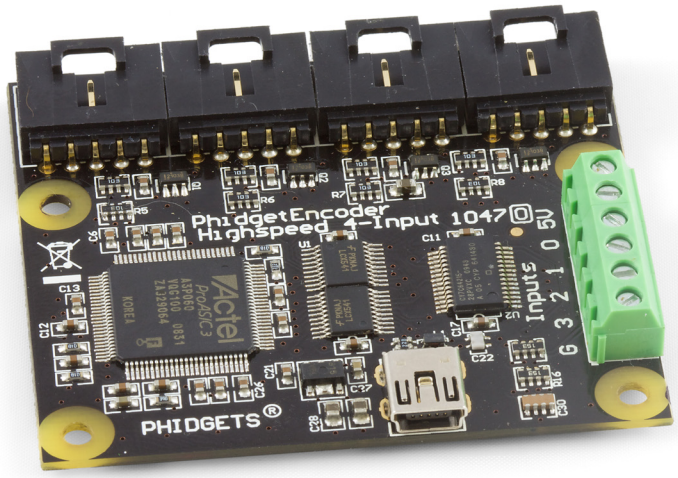
# Product Manual

## 1047 - PhidgetEncoder HighSpeed 4-Input



**Phidgets 1047 - Product Manual**

**For Board Revision 0**

# 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

6 Downloading the Phidgets drivers

6 Running Phidgets Sample Program

7 If you are 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 PhidgetEncoder Highspeed 4-Input

10 Functions

11 Events

## 12 Technical Section

12 General

12 Quadrature Encoder Fundamentals

13 Mechanical Drawing

13 Device Specifications

## 14 Product History

# 14 Support

# Product Features

- Reads up to four encoders simultaneously
- Ability to power down individual encoders
- Returns up to 250,000 counts per second
- Detects changes in incremental position
- Easily tracks changes with respect to time
- Accurately measures the time between encoder counts
- High resolution time measured in microseconds
- Can use the index signal to determine zero position or a complete revolution
- 4 Digital Inputs for detecting the state of switches and sensors

## 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:

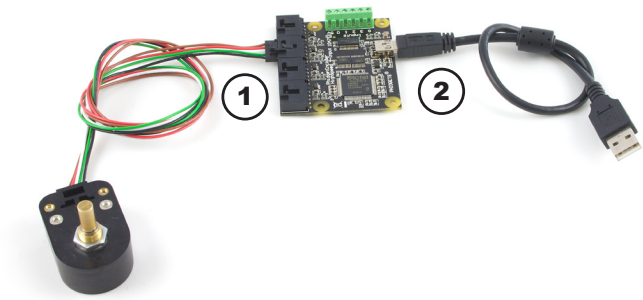
- A PhidgetEncoder Highspeed 4-Input
- A Mini-USB cable
- Four 5-wire encoder cables

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

- An incremental encoder with quadrature output and known pinout
- A piece of wire to test the digital input

## Connecting all the pieces

1. Connect the encoder to the PhidgetEncoder Highspeed 4-Input board using one of the 5-wire cables.
2. Connect the PhidgetEncoder board to the PC using the Mini-USB cable.




## Testing Using Windows 2000/XP/Vista

### 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)

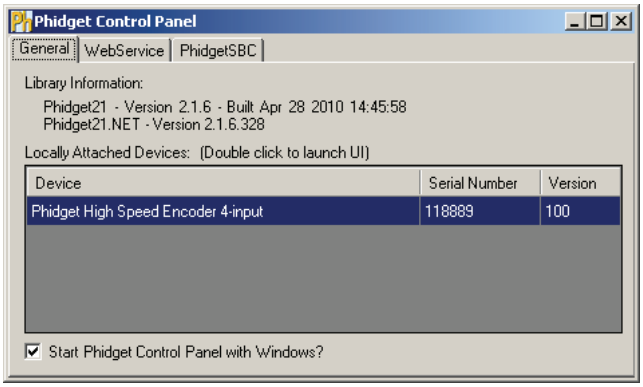
You should see the icon on the right hand corner of the Task Bar.

### 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.

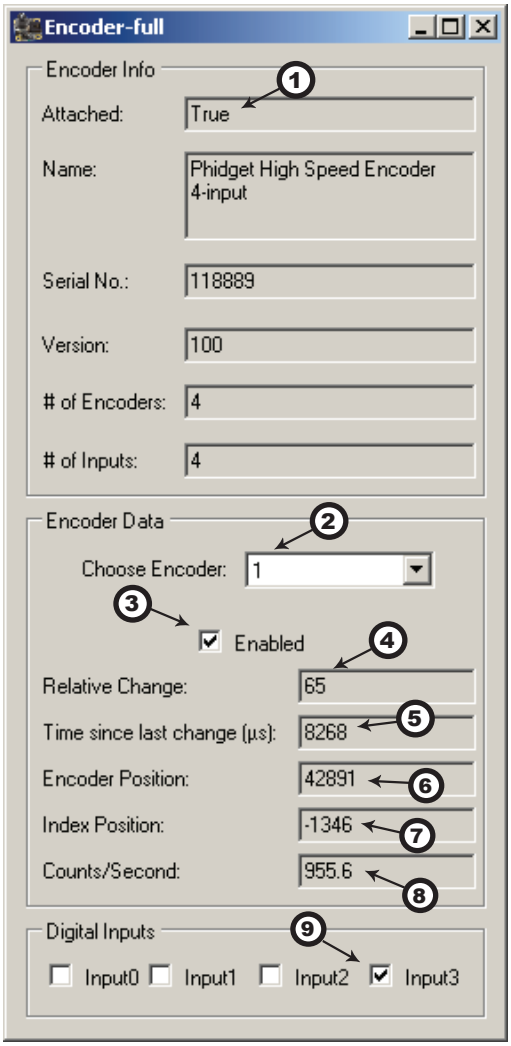
The source code for the Encoder-full sample program can be found under C# by clicking on phidgets.com >> Programming.

Double Click on the icon to activate the Phidget Control Panel and make sure that the **Phidget High Speed Encoder 4-Input** is properly attached to your PC.




1. Double Click on **Phidget High Speed Encoder 4-Input** in the Phidget Control Panel to bring up the Encoder-full example* and check that the box labelled Attached contains the word True.
2. Make sure your encoder is connected to the Encoder-0 input and select encoder 0 from the dropdown menu.
3. Enable the encoder by checking the Enabled check box.
4. Rotate the encoder and the relative change will display positive when rotated clockwise and will display negative when rotated counter-clockwise.
5. The example will display the time in microseconds that has elapsed since the last position change update received from the device.
6. Encoder position will display the absolute position of the encoder since it was initially enabled. Position will increase when the encoder is rotated clockwise and decrease when rotated counter-clockwise.
7. Index position will display the absolute position that the encoder was in when the last index was reached.
8. Counts/second displays the current velocity of the encoder measured in encoder counts per second.
9. Insert a piece of wire connecting ground and Digital Input 3 and a tick mark should appear in the Input3 box.

*Encoder-full is an example program and is not intended to be used in applications. It is meant to show how the API calls can be used and can be a starting point for your own application.




## Testing Using Mac OS X

- Click on System Preferences >> Phidgets (under Other) to activate the Preference Pane
- Make sure that the **Phidget High Speed Encoder 4-Input** is properly attached.
- Double Click on **Phidget High Speed Encoder 4-Input** in the Phidget Preference Pane to bring up the Encoder-full Example. This example 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.
- 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 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 PhidgetEncoder Highspeed 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.

## Functions

**int InputCount() [get] : Constant**

Returns the number of digital inputs supported by this PhidgetEncoder. On the 1047, there are 4 digital inputs.

**bool InputState (int EncoderIndex) [get]**

Returns the state of a particular digital input.

**int EncoderCount() [get] : Constant**

Returns the number of encoders supported by this PhidgetEncoder Highspeed 4-Input. The 1047 supports up to four optical quadrature encoders.

**int Position(int EncoderIndex) [get,set]**

Returns/sets the position of an encoder. This is an absolute position as calculated since the encoder was plugged in. Dividing position by the number of increments per revolution will give the number of rotations the encoder has travelled. Encoder position is returned in number of pulses, which is equivalent to four-times the number of encoder counts.

Position can be set, typically used when an encoder has reached an identifiable (through external means, such as a limit switch) home position. This call does not send information to the device, as an absolute position is maintained only in the library. After this call, position changes from the encoder will use the new value to calculate absolute position.

**int IndexPosition(int EncoderIndex) [get]**

Returns the index position of an enabled encoder. This will typically return the same value as Position(int EncoderIndex) except for on the event that an index has occurred, where it will return the exact position at which the encoder index was triggered.

**bool Enabled(int EncoderIndex) [get,set]**

Returns true if the requested encoder is enabled; returns false if the requested encoder is disabled.

Can be set true or false to decide whether the encoder is turned on or off. Disabled encoders will not draw power, detect position changes or trigger index events. Enabled encoders will draw power and trigger any requested events.

## Events

**OnPositionChange(int EncoderIndex, int Time, int PositionChange) [event]**

An event that is issued whenever a change in encoder position occurs on an enabled encoder. This event returns the length of time that the change took (in microseconds), and the amount of change (positive/negative encoder increments) for that particular encoder.

**OnIndex(int EncoderIndex, int Time, int PositionChange) [event]**

An event that is issued whenever the index is triggered from any one of the enabled encoders.

**OnInputChange(int InputIndex, bool State) [event]**

An event that is issued whenever the state of a digital input changes.

# Technical Section

## General

The PhidgetEncoder Highspeed 4-Input can be used with a wide assortment of mechanical and optical encoders. The encoder should be of quadrature output type, indicating that there will be two quadrature output channels (usually labeled A and B) and a third output channel (only on some encoders) to signal when the index pin (a reference point for zero position or a complete revolution) has been reached.

The PhidgetEncoder Highspeed 4-Input is able to read four encoders simultaneously. Encoders are not powered up until all initialization of the device is complete. It is possible to enable some or all encoders, depending on how many of the channels are being used. This can also be used to reduce power consumption when certain encoders are not needed.

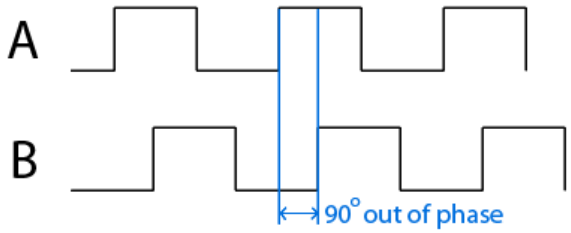
The PhidgetEncoder Highspeed 4-Input has the added ability to time the duration between a group of quadrature changes. The time is returned in microseconds. This time value can be used to calculate velocity and acceleration with very high precision.

If the number of quadrature counts per revolution is unknown for a particular encoder, this value can be determined by using the index signal. In addition, it is possible to monitor how many counts have occurred since the last index. The index signal is an output only on certain encoders. Refer to the encoder's description to check if this third output channel exists or not. If the encoder does not have this signal, it is still possible to use it with the PhidgetEncoder Highspeed 4-Input, but an event for the index will never get triggered.

The maximum rate of the PhidgetEncoder Highspeed 4-Input is specified at 250,000 counts 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 measures 1000 counts per revolution, then the limit on measurable revolutions per second is 250, or 15,000rpm.

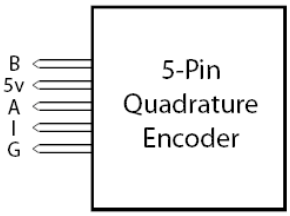
## Quadrature Encoder Fundamentals

Quadrature encoders are common, using two output channels to dictate both a change and the direction of change. In an encoder system, two parallel mechanical switches or optical slots are offset slightly. This way, as the slots passs by the sensor, the staggered output indicates both the number of pulses that have occurred (the change in position) as well as which output channel is leading the other (direction of change).




# Choosing Encoders

Both mechanical and optical encoders are available, with optical encoders dominating at > 100 counts per second. Review the data sheet for the encoder you are planning to use to ensure it is compatible with the PhidgetEncoder Highspeed 4-Input. Compatible encoders should operate from the +5VDC power provided by the PhidgetEncoder, and use a single wire for the A, B, and optional Index channels. Some types of encoders will use two wires (differential) for each channel - these are not compatible. Compatible encoders are often advertised as being 'single ended', and will have 4 or 5 connections.




Absolute encoders will not work with this device.

Mechanical encoders are effectively just two switches, and often have a push button switch on the shaft. This push button switch can be wired into a digital input on the PhidgetEncoder. Mechanical encoders do not have to be connected to +5V.

**Warning:** The PhidgetEncoder Highspeed 4-Input incorporates a 10 kOhm pull-up resistor on each line from the encoder input connector. If your encoder is mechanical, these pull-up resistors eliminate the requirement to add your own external pull-up resistors.

Some optical encoders will have a simple photo-transistor / open-collector output. The 10 kOhm pull-up resistor may have to be augmented with a stronger parallel resistor if your optical encoder datasheet calls for it. Some open-collector outputs will not be strong enough to pull this resistor to ground. These encoders are not compatible

with the 1047, and may only work initially, or not at all.  If you have any doubts, please contact us.

Most optical encoders have a push-pull output, and the pull-up resistor is irrelevant, but weak enough not cause problems.

We have reviewed the following encoders, and found that they can be used with the PhidgetEncoder Highspeed 4-Input.  This is not meant to be a comprehensive list but should be used as examples of the type of encoders that can be used with the 1047.

| Manufacturer | Web Page | Part Number |
|---|---|---|
| Grayhill | www.Grayhill.com | Series 63R, Series 61R Series 63Q TTL Output |
| US Digital (recommended) | www.USDigital.com | S4, S5, E2, E3, E4, E4P, etc. |
| Avago Technologies (Formerly Agilent) | www.avagotech.com | HEDS 5500 |
| CUI Inc. | www.cui.com | AMT103-V |

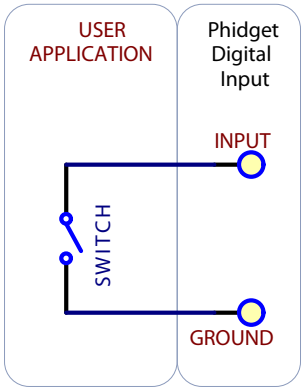
# Digital Inputs

## Using the Digital Inputs

Here are some circuit diagrams that illustrate how to connect various devices to the digital inputs on your Phidget.

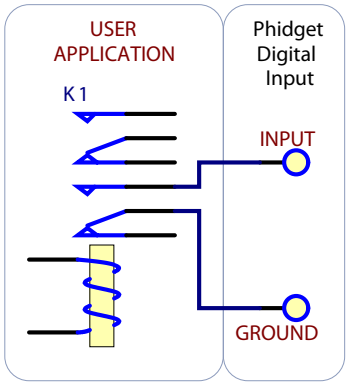
### Wiring a switch to a Digital Input

Closing the switch causes the digital input to report TRUE.




### Monitoring the position of a relay

The relay contact can be treated as a switch, and wired up similarly. When the relay contact is closed, the Digital Input will report TRUE.



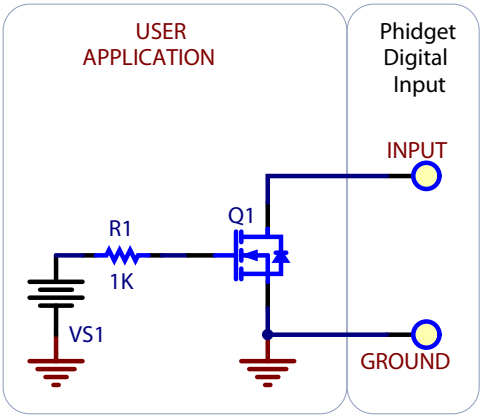

### Detecting an external Voltage with an N-Channel MOSFET

A MOSFET can be used to detect the presence of an external voltage. The external voltage will turn on the MOSFET, causing it to short the Digital Input to Ground.

If the MOSFET is conducting > 280µA, the Digital Input is guaranteed to report TRUE.

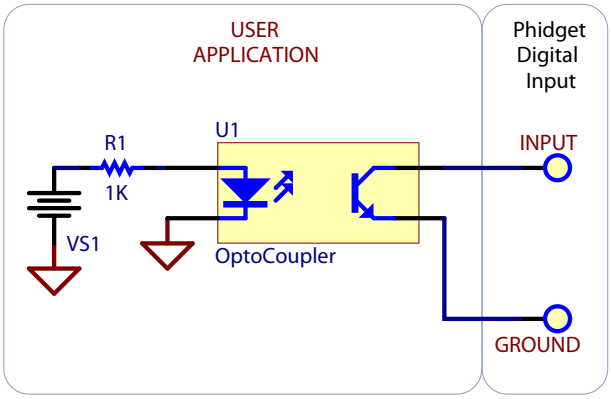
If the MOSFET is conducting < 190µA, the Digital Input is guaranteed to report FALSE.

The voltage level required to turn on the MOSFET depends on the make of of MOSFET you are using. Typical values are 2V-6V.



### Isolating a Digital Input with an Optocoupler

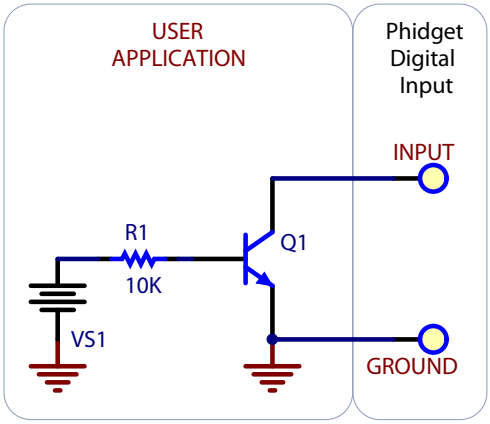
When driving current through the LED, the Digital Input will report TRUE. The amount of current required will depend on the optocoupler used. Design to sink at least 280µA to cause the digital input to report TRUE, and less than 190µA to report FALSE.




### Detecting an external Voltage with an NPN Transistor

This circuit can be used to measure if a battery is connected, or if 12V (for example) is on a wire.

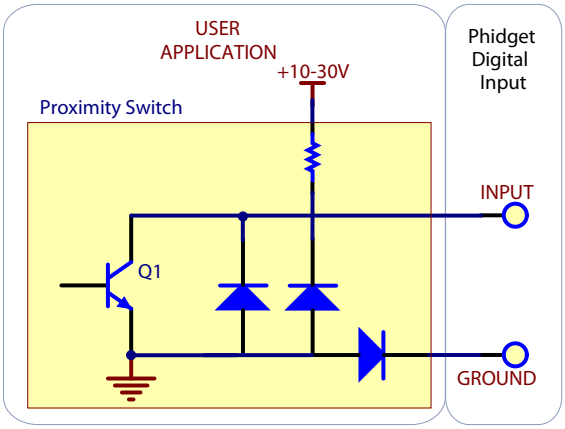
By designing to have Collector-Emitter current > 280µA, the digital input will report TRUE.




### Using a Capacitive or Inductive Proximity Switch

Capacitive proximity switches can detect the presence of nearby non-metallic objects, whereas inductive proximity switches can detect only the presence of metallic objects. To properly interface one of these proximity switches to the digital inputs, a 3-wire proximity switch is required, as well as an external power supply.

We have checked the following switch from Automation Direct to verify that it works with the Digital Inputs. Similar capacitive or inductive proximity switches from other manufacturers should work just as well.



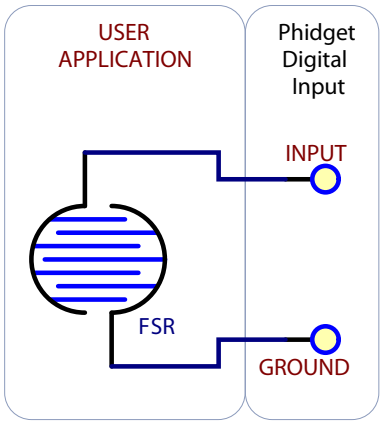

| Manufacturer | Web Page | Capacitive Part No | Inductive Part No |
|---|---|---|---|
| Automation Direct | www.automationdirect.com | CT1 Series | AM1 Series |

**Using an FSR or other variable resistor as a switch**

The digital inputs can be easily wired to use many variable resistors as switches.

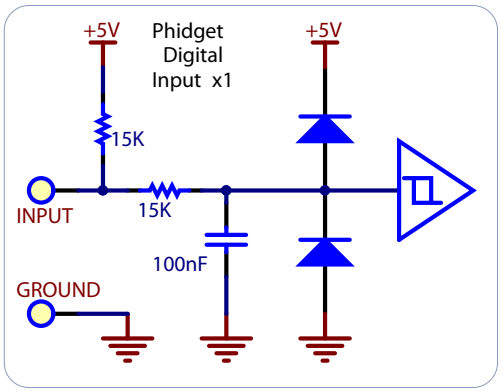
If the resistance falls below 2.8k Ohms, the Digital Input will go TRUE.

If the resistance rises above 11k Ohms, the Digital Input will go FALSE.




## Functional Block Diagram

The digital inputs have a built in 15K pull-up resistor. By connecting external circuitry, and forcing the input to Ground, the Digital Input in software will read as TRUE. The default state is FALSE - when you have nothing connected, or your circuitry (switch, etc) is not pulling the input to ground.




## Digital Input Hardware Filter

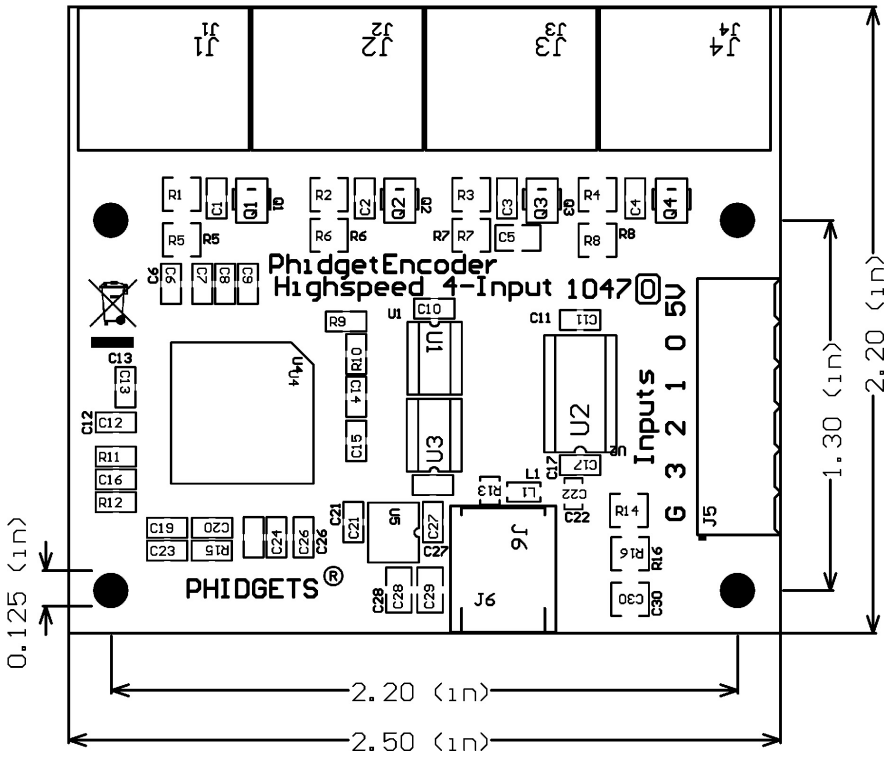
There is built-in filtering on the digital input, to eliminate false triggering from electrical noise. The digital input is first RC filtered by a 15K/100nF node, which will reject noise of higher frequency than 1Khz. This filter generally eliminates the need to shield the digital input from inductive and capacitive coupling likely to occur in wiring harnesses.

## Digital Input Sampling Characteristics

The state of the digital inputs are reported back to the PC periodically. During this sampling period, if a digital input was true for greater than 4.0ms, the digital input is guaranteed to be reported as true in software. This makes the digital input much more sensitive to reporting TRUE state, and makes it useful to watch for short events. Any Digital Input True events of less than 1.5ms are never reported.

## Mechanical Drawing

1:1 scale




## Device Specifications

| Characteristic | Value |
|---|---|
| **Encoder** | |
| Maximum Count Rate | 250,000 counts/second |
| Internal Output Pull-Up Resistance | 10 kilo Ohms |
| Software Update Rate (typical) | 8 milliseconds |
| USB Update Rate | 125 samples/second |
| Time Resolution | 1 µs |
| **Current** | |
| Max Device Current Consumption (all channels enabled) | 500 mA |
| Min Device Current Consumption (all channels disabled) | 30 mA |
| Maximum Current Consumption per Encoder | 200 mA |
| Maximum Current Consumption for all Encoders | 470 mA |
| **Voltage** | |
| Min/Max USB Supply Voltage | 4.75 - 5.25 VDC |
| Encoder Input 0 Voltage | < 1.4 V |
| Encoder Input 1 Voltage | > 1.8 V |
| **Digital Inputs** | |
| Digital Input Pull-Up Resistance | 15K ohms |
| Digital Output Update Rate | ~125 samples/second |
| Digital Input Recommended Wire Size | 16 - 26 AWG |
| Digital Input Wire Stripping | 5-6mm strip |

# Product History

| Date | Board Revision | Device Version | Comment |
|---|---|---|---|
| JUne 2010 | 0 | 100 | Product Release |

# Support

- Call the support desk at 1.403.282.7335 8:00 AM to 5:00 PM Mountain Time (US & Canada) - GMT-07:00

or

- E-mail us at: support@phidgets.com