1052 - PhidgetEncoder

Product Features

- Uses a two-bit mechanical encoder with a built-in momentary-action push-button switch.
- Returns 80 counts for 360 degrees of rotation.
- Detects changes in incremental and absolute position.
- Easily track changes with respect to time.
- Intended to be used as a human interface, not as a device to measure shaft speed.
- Connects directly to a computer's USB port

Programming Environment

Operating Systems: Windows 2000/XP/Vista, Windows CE, Linux, and Mac OS X

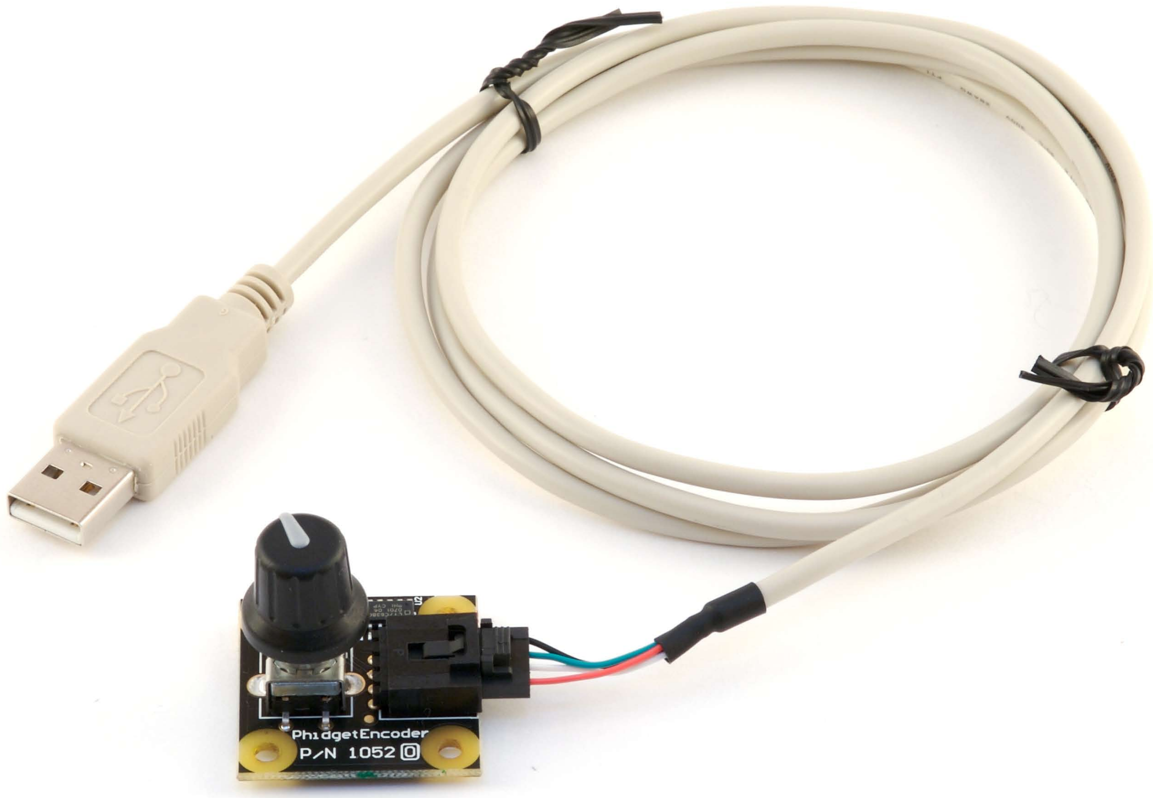
Programming Languages (APIs): VB6, VB.NET, C#.NET, C++, Flash 9, Flex, Java, LabVIEW, Python, Max/MSP, and Cocoa.

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 PhidgeEncoder board
- A custom USB cable



Connect the PhidgeEncoder board to the computer using the USB cable.

Downloading and Installing the software

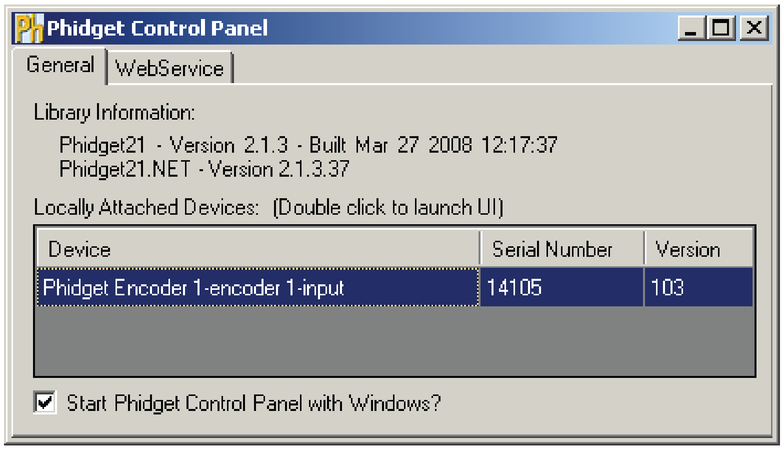
If you are using Windows 2000/XP/Vista

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

Download and run Phidget21.MSI

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

Testing the PhidgetEncoder Functionality



Library Information:

Phidget21 - Version 2.1.3 - Built Mar 27 2008 12:17:37
Phidget21.NET - Version 2.1.3.37

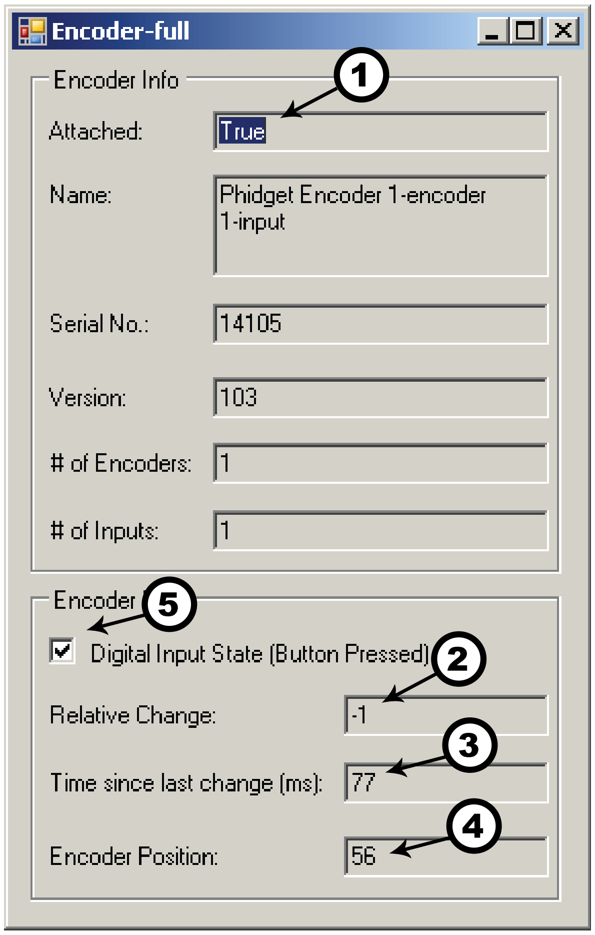
Locally Attached Devices: (Double click to launch UI)

Device	Serial Number	Version
Phidget Encoder 1-encoder 1-input	14105	103

☑ Start Phidget Control Panel with Windows?

Double Click on the Ph icon to activate the Phidget Control Panel and make sure that the ***PhidgetEncoder*** is properly attached to your PC.

1. Double Click on ***Phidget Encoder*** in the Phidget Control Panel to bring up Encoder-full and check that the box labelled Attached contains the word True.
2. As you turn the knob the relative change will be positive when turning counterclockwise and negative when turning clockwise.
3. Displays the time in milliseconds since the last time you turned the knob.
4. Returns 80 counts for each knob revolution. The number will decrease when turning the knob clockwise and increase when turning counterclockwise.
5. Tick mark appears when the knob is depressed.



Encoder Info

Attached:	True
Name:	Phidget Encoder 1-encoder 1-input
Serial No.:	14105
Version:	103
# of Encoders:	1
# of Inputs:	1

Encoder

☑ Digital Input State (Button Pressed)

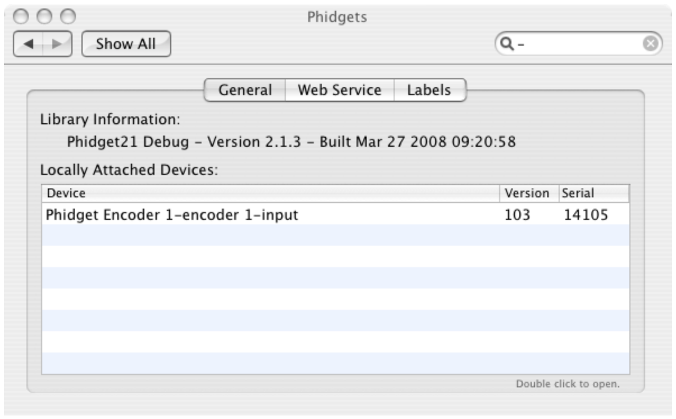
Relative Change:	-1
Time since last change (ms):	77
Encoder Position:	56

If you are using Mac OS X

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

Download Mac OS X Framework

Testing the PhidgetEncoder functionality



Library Information:
Phidget21 Debug - Version 2.1.3 - Built Mar 27 2008 09:20:58

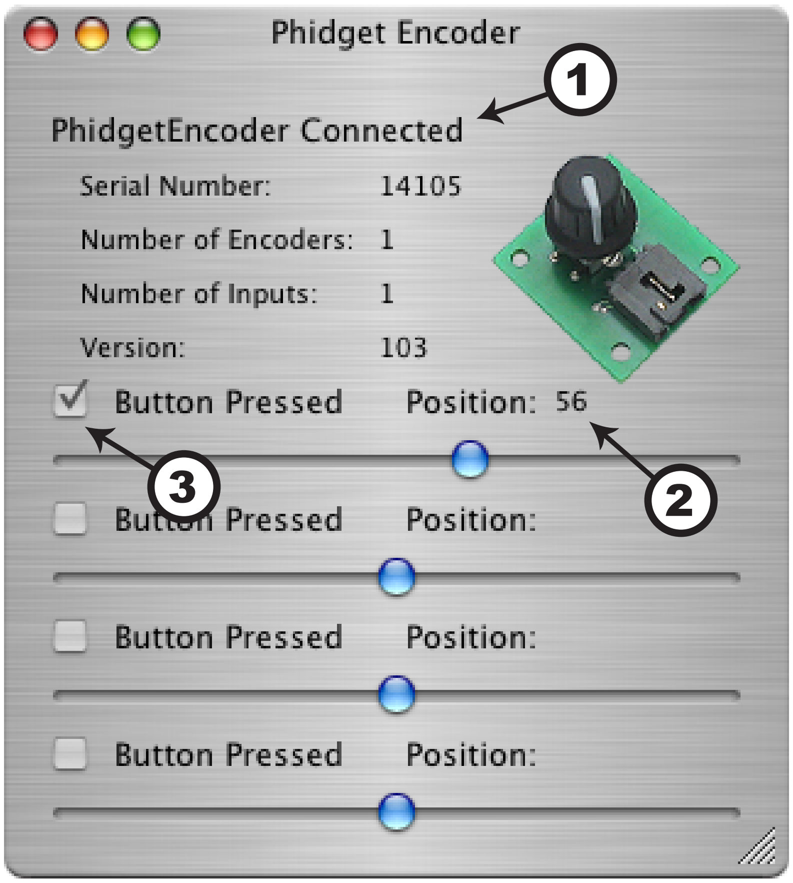
Locally Attached Devices:

Device	Version	Serial
Phidget Encoder 1-encoder 1-input	103	14105

Double click to open.

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

1. Double Click on ***Phidget Encoder*** in the Phidgets Preference Pane to bring up Phidget Encoder Example and check that the PhidgetEncoder is properly connected.
2. Returns 80 counts for each knob revolution. The number will decrease when turning the knob clockwise and increase when turning counterclockwise.
3. Tick mark appears when the knob is depressed.



PhidgetEncoder Connected

Serial Number: 14105
Number of Encoders: 1
Number of Inputs: 1
Version: 103

✓ Button Pressed Position: 56

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 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.

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, including .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 beside 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#

```
/* - Encoder simple -
 ****************************************************************************************
 * This example simply creates an Encoder object, hooks the event handlers, and then
 * waits for an encoder is attached. Once it is attached, the program will wait for user
 * input so that we can see the event data on the screen when using the encoder.
 * For a more detailed example, please see the Encoder-full example.
 *
 * Please note that this example was designed to work with only one Phidget Encoder
 * connected.  For an example using multiple Phidget Encoders, please see a “multiple”
 * example in the Encoder 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;
using Phidgets; //Needed for the Encoder class and the PhidgetException class
using Phidgets.Events; //Needed for the event handling classes

namespace Encoder_simple
{
    class Program
    {
        //Declare a Phidgets.Encoder object (there is a naming ambiguitity in .NET since
        //there is an Encoder object in the .NET library. So need to specify it as a
        //phidgets one here in this case
        static Phidgets.Encoder encoder;

        static void Main(string[] args)
        {
            try
            {
                //Initialize the Encoder object
                encoder = new Phidgets.Encoder();

                //Hook the basic event handlers
                encoder.Attach += new AttachEventHandler(encoder_Attach);
                encoder.Detach += new DetachEventHandler(encoder_Detach);
                encoder.Error += new ErrorEventHandler(encoder_Error);

                //Hook the phidget specific event handlers
                encoder.InputChange += new InputChangeEventHandler(encoder_InputChange);
                encoder.PositionChange += new EncoderPositionChangeEventHandler
                                                    (encoder_PositionChange);

                //open the object for connections
                encoder.open();

                //Wait for an encoder device to be attached before continuing
                Console.WriteLine(“Waiting for Encoder to be attached....”);
                encoder.waitForAttachment();

                //Wait for user input before continuing so that we can see the events
                //being sent when using the encoder
                Console.WriteLine("Press any key to end...");
                Console.Read();

                //Since user input is read we can terminate the program, so we'll close
                //the encoder object
                encoder.close();

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

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

        //Attach event handler...Display the serial number of the attached Encoder to
        //the console
        static void encoder_Attach(object sender, AttachEventArgs e)
        {
            Console.WriteLine("Encoder {0} attached!",
                                 e.Device.SerialNumber.ToString());
        }

        //Detach event handler...Display the serial number of the detached Encoder to
        //the console
        static void encoder_Detach(object sender, DetachEventArgs e)
        {
            Console.WriteLine("Encoder {0} detached!", e.Device.SerialNumber.ToString());
        }

        //Error event handler...Display the error description to the console
        static void encoder_Error(object sender, ErrorEventArgs e)
        {
            Console.WriteLine(e.Description);
        }

        //Input Change event handler...DIsplay the new value of te digital input
        //(the pushbutton) to the console
        static void encoder_InputChange(object sender, InputChangeEventArgs e)
        {
            Console.WriteLine("Input {0} value {1}", e.Index, e.Value);
        }

        //Position change event handler...Display the index of the encoder that change,
        //the position change value, the time between changes, and the current encoder
        //position value to the console
        static void encoder_PositionChange(object sender,
                                               EncoderPositionChangeEventArgs e)
        {
            Console.WriteLine("Encoder {0} Change {1} Time {2} Position {3}", e.Index,
                                   e.PositionChange, e.Time, encoder.encoders[e.Index]);
        }
    }
}
```

Technical Section

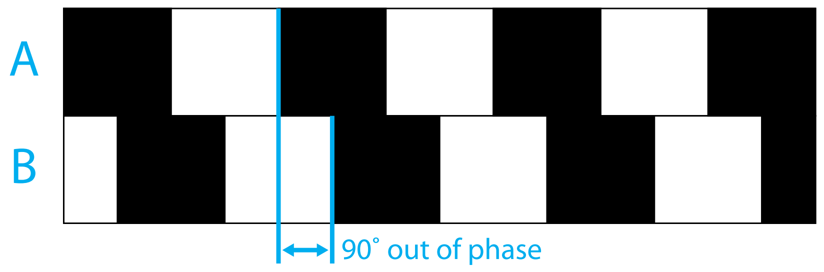
The PhidgetEncoder uses a two-bit quadrature mechanical encoder with a built-in momentary pushbutton SPST switch. It returns 80 counts for 360 degrees of rotation. With It you can:

- Detect changes in incremental and absolute position
- Easily track these changes with respect to time

The encoder that comes with the PhidgetEncoder is manufactured by CTS, part number 290UAA5F201B1.

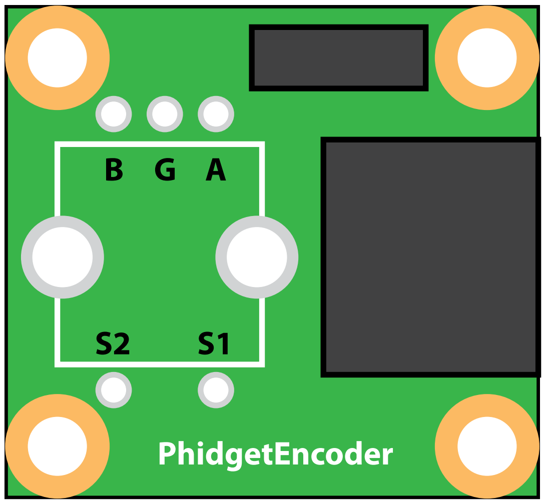
Quadrature Encoder Fundamentals

Quadrature encoders are the most popular type of encoder, using two output channels to dictate both angular displacement and direction of rotation. The term quadrature relates to a system whose components exist at an angle of 90° to each other. In an encoder system, two parallel mechanical disks or optical gratings are set 90° out of phase. In this way, as the two disks spin in unison, the output can signify both the number of pulses that have occured (the angular displacement) as well as which output channel is leading the other (direction of rotation). In the diagram below, the black segments represent a closed switch and the white segments represent an open switch.



Using Other Encoders

If you decide that the mounted encoder does not meet your requirements it is possible to use other two-bit mechanical encoders. Optical encoders with large counts per revolution should not be expected to work with this device. To change the encoder the user will have to carefully desolder and remove the existing encoder. However, if you attempt to remove the encoder the warranty willl be void and we can not offer technical support. For assistance on desoldering techniques we recommend using any search engine on the Internet. Since your new encoder may not have the exact same pin configuration, you should use wire to connect your new encoder to the appropriate pins on the PhidgetEncoder board. We have tested the following encoders, and found that they can be used as replacements:



Manuf.	Part Number	Description
CTS	290UAA5F201B2	2-Bit 9mm 20CPR Encoder Detent w/MO SPST
CTS	290UAA5F201B1	2-Bit 9mm 20CPR Encoder Non-Detent w/MO SPST
CTS	288V232R161B2	2-Bit 16mm 16CPR Encoder Detent w/MO SPST

Note: Most of the above components can be bought at www.digikey.com

API Section

We document API Calls specific to the 1052. 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 InputCount() [get] : Constant

Returns the number of digital inputs supported by this PhidgetEncoder. On the 1052, there is one digital input - the push button on the shaft of the encoder.

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. On the 1052, there is one encoder - the unit mounted on the board.

int Position(int EncoderIndex) [get,set]

Returns/sets the position of an encoder. This is an absolute position as calcutated since the encoder was plugged in. Dividing position by the number of increments per revolution (80) will give the number of rotations the encoder has travelled.

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

Events

OnDigitalInputChange(int InputIndex, bool State) [event]

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

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

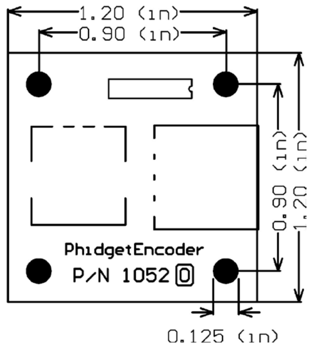
An event that is issued whenever a change in encoder position occurs. This event returns the length of time that the change took (in milliseconds), and the amount of change (positive/negative encoder increments)

Device Specifications

Position Update Rate	30 Hz
Encoder Output Pull-Up Resistance	470 Ω
Device Current Consumption	20mA

Mechanical Drawing

1:1 scale



Product History

Date	Product Revision	Comment
January 2003	DeviceVersion 100	Product Release
July 2004	DeviceVersion 110	Protocol modified to allow multiple encoder input expansion