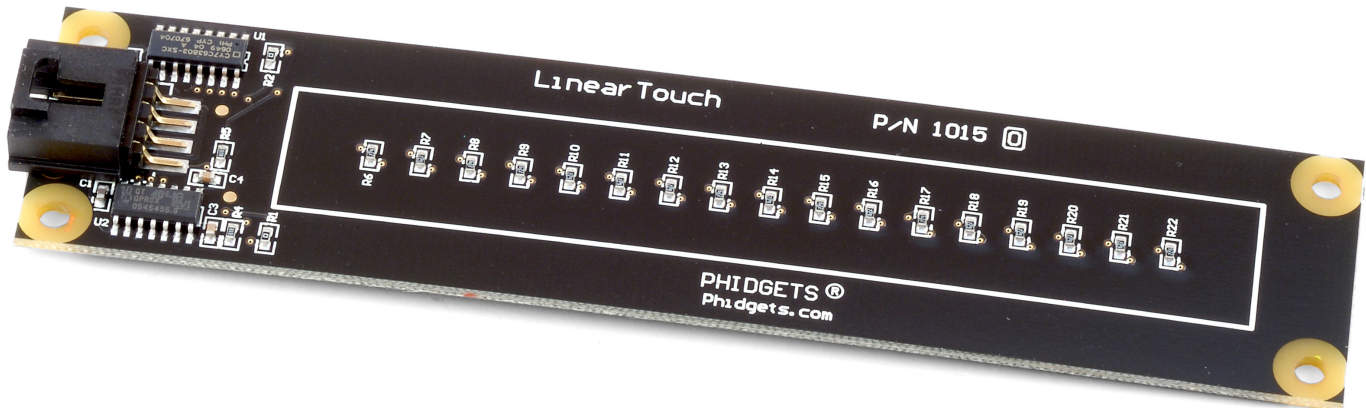# 1015 - PhidgetLinearTouch

Product Features

- Changes value when it is touched, detecting approximately 125 discrete positions.

- Works through 1/8 inch of glass or plastic.

- Recognizes both contact and proximity, and can be used as a slider or as an array of buttons.

- 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 PhidgetLinearTouch board.
- A custom USB cable
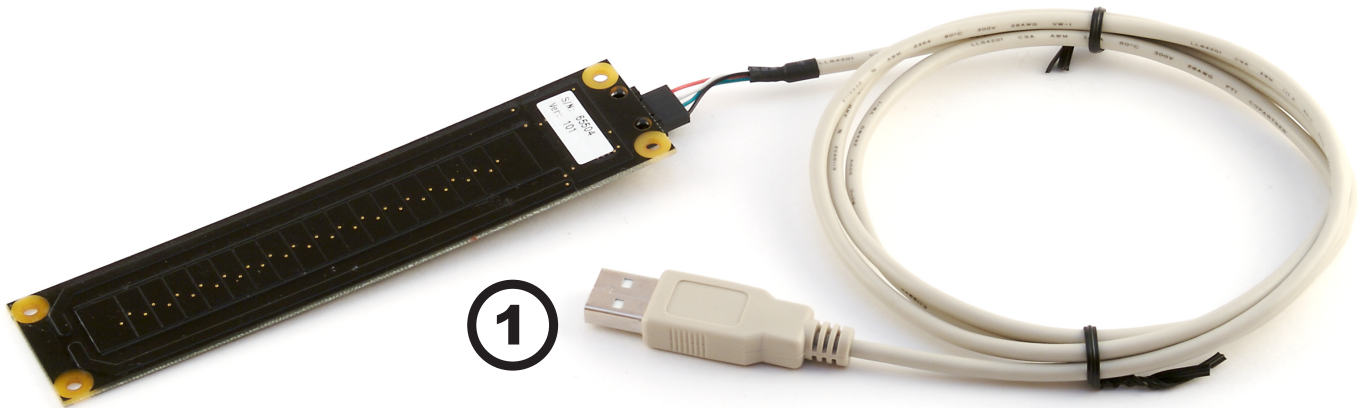
1. Connect the PhidgetLinearTouch board to the computer using the custom USB cable included.
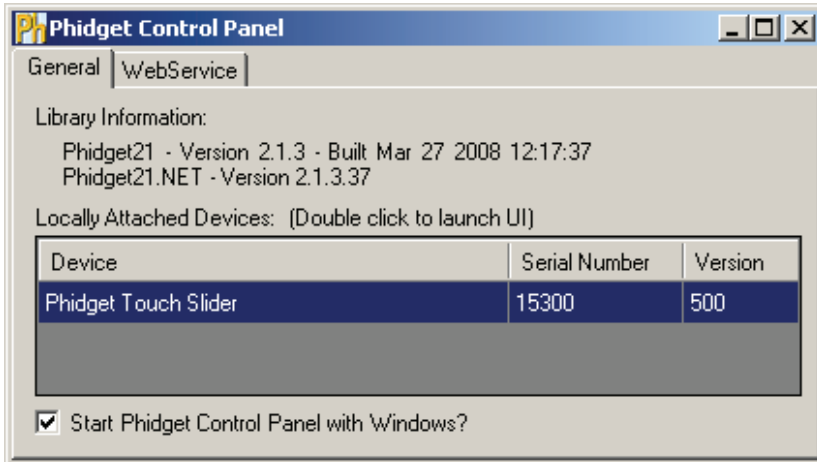
Downloading and Installing the software

If you are using Windows 2000/XP/Vista

Go to www.phidgets.com >> Drivers

Download and run Phidget21.MSI

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

Testing the PhidgetLinearTouch Functionality

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

1. Double Click on **Phidget Touch Slider** in the Phidget Control Panel to bring up InterfaceKit-full and check that the box labelled Attached contains the word True.

2. As you bring your finger closer to the board and touch the board, tick marks will appear in the Digital In boxes. The left one shows "touch" and the right one shows "proximity".

3. Move your finger along the back side of the PhidgetLinearTouch and watch the numbers in the Analog box change from 0 to 1000. The numbers are only significant when both Digital In boxes are "tick marked".

4. Adjusting the input sensitivity with the sensitivity slider changes the number of discrete steps that will fire the on-sensor-change event.
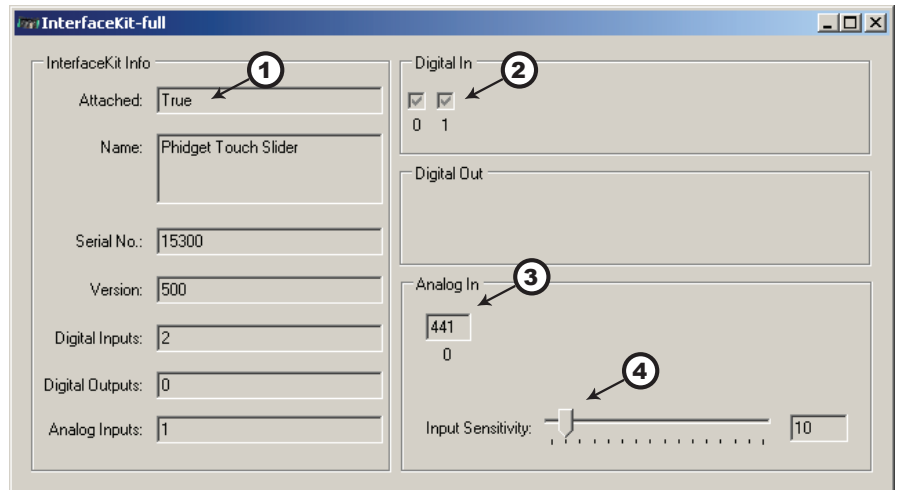
If you are using Mac OS X

Go to www.phidgets.com >> Drivers

- Download Mac OS X Framework

- Click on System Preferences >> Phidgets (under Other) to activate the Preference Pane.

- Make sure that your Phidget is properly attached.

- Double click on the attached Phidget to launch the Example.

If you are using 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 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.

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.

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 under Programming and are listed under the appropriate language.

API documentation

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 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 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.

Support

- 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 support@phidgets.com

Technical Section

The PhidgetLinearTouch is actually a capacitive-charge sensor, detecting changes in the capacitance between the on-board electrodes and the object making contact. The side of the circuit board opposite the connector and components is the side intended for contact. The internal sensor used for charge-detection is a Quantum Research Group QT401 Sensor.

Device Inputs

The PhidgetLinearTouch appears to the Phidget software libraries as an InterfaceKit. Sliding a finger along the touch sensor varies the Analog Input 0 value from 0 to 1000 in approximately 125 discrete steps. When the finger is removed, the final measured value is retained. Two Digital Inputs are also utilized to convey additional information: Digital Input 0 appears True when contact is made with the electrodes on the Phidget, and Digital Input 1 appears True when a finger or contacting object comes in close proximity to the electrodes. The two Digital Inputs are intended to be used as a quality measure, allowing the developer to trust the Analog Input value only when both Digital Inputs are true.

Input	Range	Description
Analog Input 0	0 - 1000	Analog value representing touch position
Digital Input 0	True/False	True indicates physical electrode contact
Digital Input 1	True/False	True indicates proximity to electrodes

For many projects it is required that the highest and lowest available values be more readily accessible than the full range. The PhidgetLinearTouch board has been designed so that the end-zones of the touch area have a greater contact area, allowing for effective maximum/minimum value control. If it is desired to use the touch slider as an array of buttons, or a combination of an array of buttons and a smaller slide-touch area, one must only interpret specific sub-ranges of sensor values differently in software depending upon the intended use. If sub-ranges of values are to be used as buttons, it is recommended that a small range of sensor values be left between the sub-ranges where a null-response is observed.

Dielectric Separation

The PhidgetLinearTouch has been left without components on the contact side so that it may be mounted behind a sheet of glass or plastic. The recommended thickness of separation material is 1/8 inch. Silicon adhesive is recommended when attaching the Phidget to the material; standing the PhidgetLinearTouch off or creating space between the separation material and the Phidget can cause false-triggering to occur. It should be noted that materials thicker than 1/8" may work, but will require a larger surface area of contact to ensure proper triggering (i.e.: two fingers instead of one).

API (Software Technical)

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, please refer to the device specifications.

Functions

int InputCount() [get] : Constant = 2

Returns the number of digital inputs supported. Please refer to the Device Inputs section for details on the abstractions used on the 1015.

bool InputState(int InputIndex) [get]

Returns the state of a particular digital input.

int SensorCount() [get] : Constant = 1

Returns the number of sensors (Analog Inputs) supported by this PhidgetInterfaceKit. Intrepreted together with the digital inputs, this represents the positiion of the user's finger on the slider.

int SensorValue(int SensorIndex) [get]

Returns the sensed value of a particular Analog Input. Please refer to the Device Inputs section for details on the abstractions used on the 1015.

double SensorChangeTrigger (int SensorIndex) [get,set]

Returns the change trigger for an analog input. This is the ammount that an inputs must change between successive SensorChangeEvents. This is based on the 0-1000 range provided by getSensorValue. This value is by default set to 10.
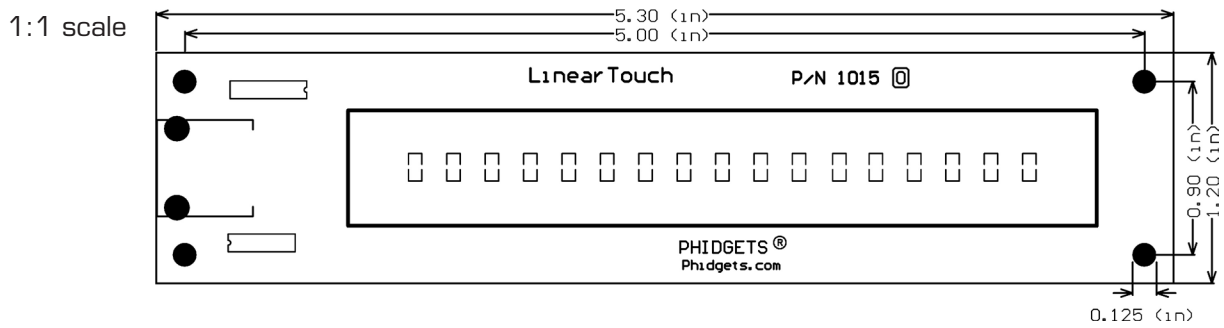
Events

OnInputChange(int InputIndex, bool State) [event]

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

OnSensorChange(int SensorIndex, int SensorValue), [event]

An event that is issued when the returned value from an Analog Input varies by more than the SensorChangeTrigger property.

Mechanical Drawing

Note: When printing the mechanical drawing, "**Page Scaling**" in the Print panel must be set to "**None**" to avoid re-sizing the image.

Device Specifications

Analog Input Update Rate	30 updates/second
Digital Input Update Rate	30 updates/second
Device Current Consumption	14mA max

Product History

Date	Product Revision	Comment
August 2005	DeviceVersion 100	Product Release
January 2006	DeviceVersion 101	Design migrated to Encore II Processor
January 2007	DeviceVersion 102	Bus Reset / Low Voltage Reset defined