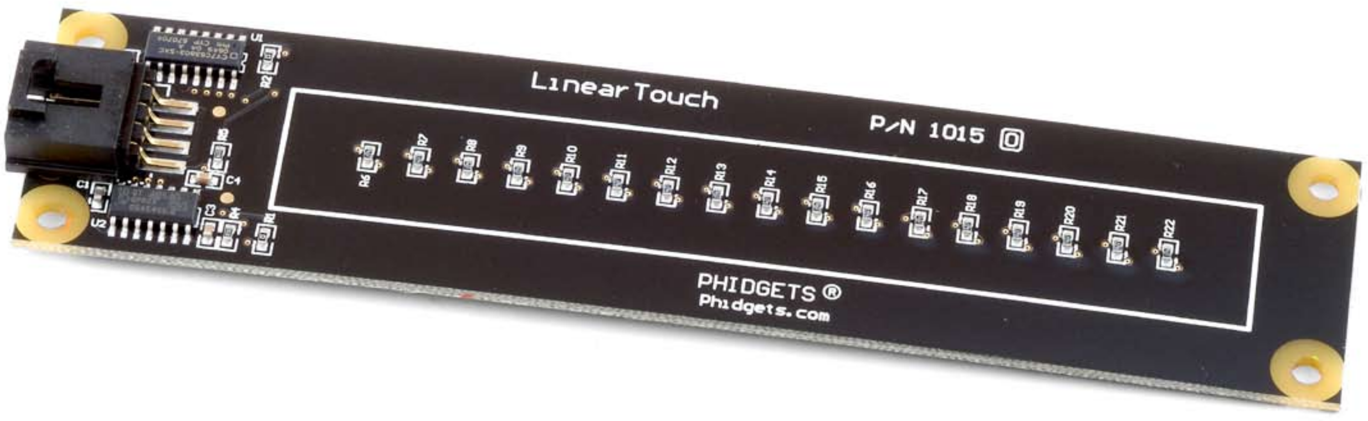
PhidgetLinearTouch



Operating Systems:

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

Application Programming Interfaces (APIs):

Visual Basic, VB.NET, C, C++, C#, Flash 9, Flex, Java, LabVIEW, and Matlab

Examples:

You will find program examples in the download section of www.phidgets.com

What Can the PhidgetLinearTouch Do?

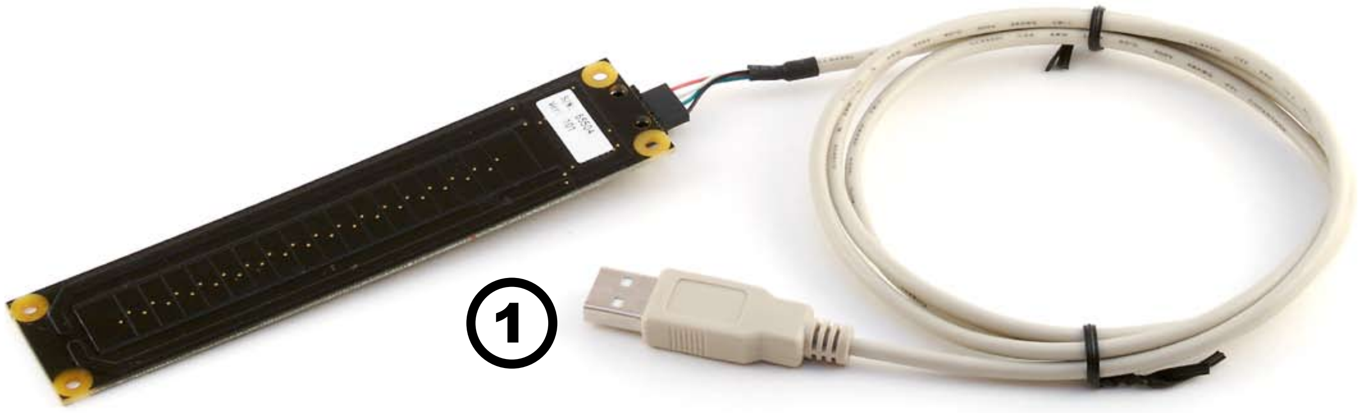
The PhidgetLinearTouch changes value when it is touched, detecting approximately 125 discrete positions. The PhidgetLinearTouch will work through 1/8 inch of glass or plastic, and appears to the Phidget software libraries as an InterfaceKit. Touch sensors are an intuitive way for people to interface with a computer's graphical user interfaces. The PhidgetLinearTouch can recognize both contact and proximity, and can be used as a slider or as an array of buttons.

Getting Started

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.

Download and Install the software

Go to www.phidgets.com >> downloads

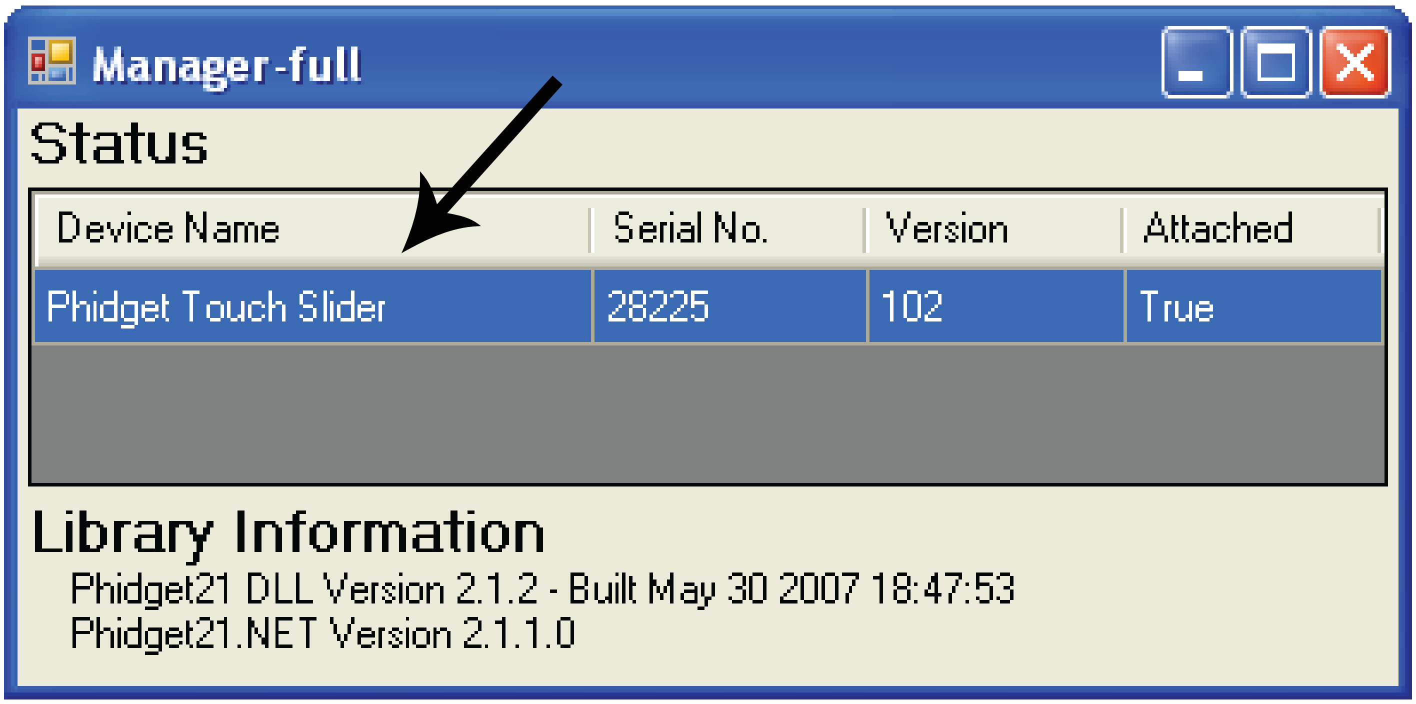
Select your operating system (Windows, Linux, MAC OS)

Select the language you want to use and download the appropriate examples and Libraries.

Install the Libraries and decompress the Example file.

Testing the PhidgetLinearTouch using Windows

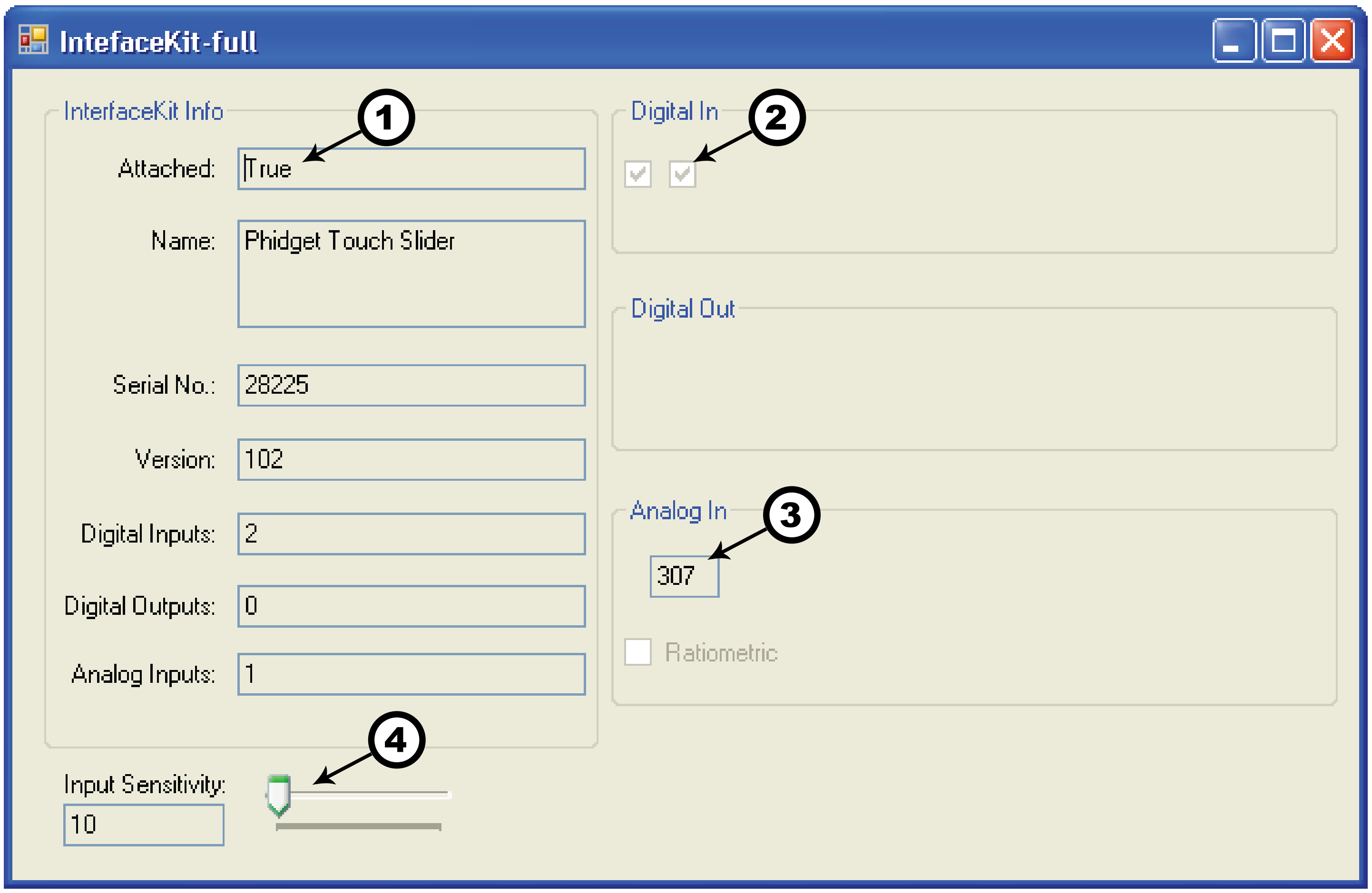
- Note that some examples are not available for Linux, Mac OSX or Windows CE.
- Make sure that you have installed the libraries and decompressed your example file.



Device Name	Serial No.	Version	Attached
Phidget Touch Slider	28225	102	True

Phidget21 DLL Version 2.1.2 - Built May 30 2007 18:47:53
Phidget21.NET Version 2.1.1.0

Run the program ***Manager-full*** to make sure that the ***PhidgetLinearTouch*** is properly connected to your PC.



InteІaceKit-full

InterfaceKit Info

Attached: True

Name: Phidget Touch Slider

Serial No.: 28225

Version: 102

Digital Inputs: 2

Digital Outputs: 0

Analog Inputs: 1

Input Sensitivity: 10

Analog In: 307

1. Run the program ***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.

Programming a Phidget

Where to get information

- Go to www.phidgets.com >> downloads
- Select the Operating System and the language you want to use.
- Download the appropriate API manual and read the section under the ***InterfaceKit*** heading.
- Have a look at the source code of the ***InterfaceKit-full*** program.

- Have a look at the C# example below.
- Modify an existing program or write your own program from scratch.

Simple example written in C#

```
/* - InterfaceKit simple -
 ****************************************************************************************
 * This simple example creates an Interfacekit object, hooks the event handlers, and
 * opens it for connections to IntefaceKit Phidgets.  It will then wait for user input to
 * terminate, and in the meantime, display event generated data from the InterfaceKit.
 * For a more detailed example, please see the InterfaceKit-full example.
 *
 * Please note that this example was designed to work with only one Phidget InterfaceKit
 * connected. For an example using multiple Phidget InterfaceKits, please see a
 * "multiple" example in the InterfaceKit 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;
//Needed for the InterfaceKit class, phidget base classes, and the PhidgetException class
using Phidgets;
//Needed for the event handling classes
using Phidgets.Events;
namespace InterfaceKit_simple
{
    class Program
    {
        //Declare an InterfaceKit object
        static InterfaceKit ifKit;

        static void Main(string[] args)
        {
            try
            {
                //Initialize the InterfaceKit object
                ifKit = new InterfaceKit();

                //Hook the basica event handlers
                ifKit.Attach += new AttachEventHandler(ifKit_Attach);
                ifKit.Detach += new DetachEventHandler(ifKit_Detach);
                ifKit.Error += new ErrorEventHandler(ifKit_Error);

                //Hook the phidget spcific event handlers
                ifKit.InputChange += new InputChangeEventHandler(ifKit_InputChange);
                ifKit.OutputChange += new OutputChangeEventHandler(ifKit_OutputChange);
                ifKit.SensorChange += new SensorChangeEventHandler(ifKit_SensorChange);

                //Open the object for device connections
                ifKit.open();

                //Wait for an InterfaceKit phidget to be attached
                Console.WriteLine("Waiting for InterfaceKit to be attached...");
                ifKit.waitForAttachment();

                //Wait for user input so that we can wait and watch for some event data
                //from the phidget
                Console.WriteLine("Press any key to end...");
                Console.Read();

                //User input was rad so we'll terminate the program, so close the object
                ifKit.close();

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

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

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

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

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

        //Input Change event handler...Display the input index and the new value to the
        //console
```

Learning more ...

- check out the forums
- check out the Phidgets projects

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

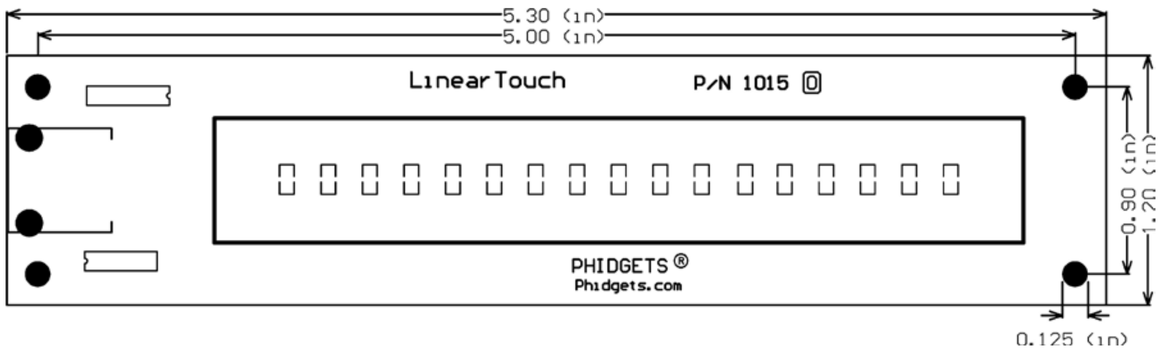
The PhidgetLinearTouch has been left without components on the contact side so that it may be mounted behind a sheet of glass or plastic. The recommended thickness of separation material is 1/8 inch. Silicon adhesive is recommended when attaching the Phidget to the material; standing the PhidgetLinearTouch 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

Device Specifications

Analog Input Update Rate	30 updates/second
Digital Input Update Rate	30 updates/second
USB Power Current Specification	500mA max
Device Quiescent Current Consumption	14mA
Device Active Current Consumption	14mA max

Mechanical Drawing

1:1 scale



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