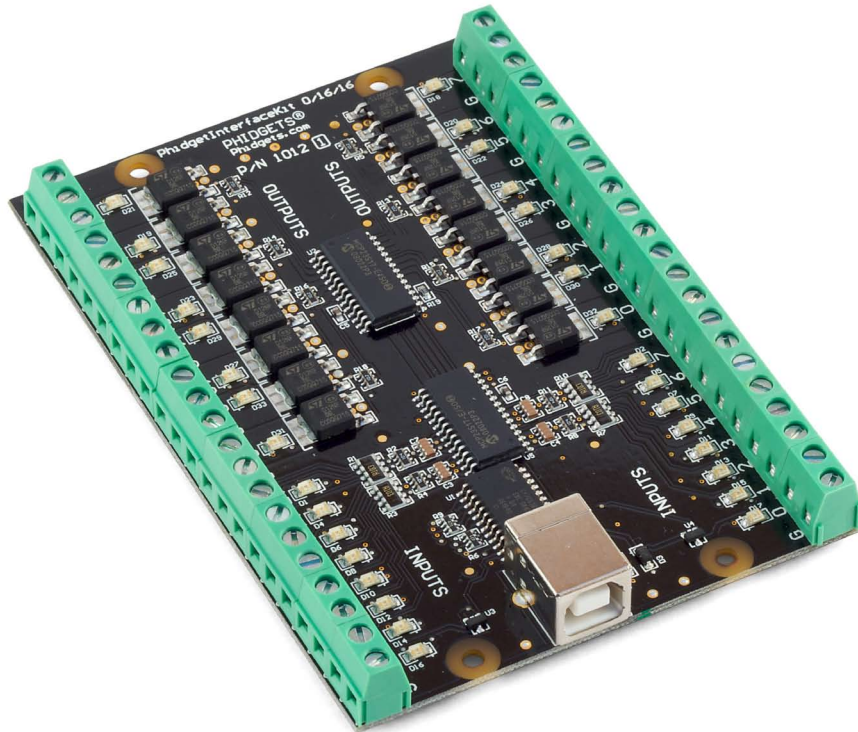


1012 - PhidgetInterfaceKit 0/16/16



Product Features

- 16 Digital Inputs that are activated by an external voltage source, triggering on a wide voltage range - 4 to 30VDC
- They can be used to convey the state of on/off devices, such as push buttons, limit switches, relays.
- 16 Digital Outputs switching up to 30VDC at up to 2 Amps.
- They can be used to directly control devices requiring substantial power such as incandescent lights, high power LEDs, relays, solenoids, motors.
- The Digital Output acts as a switch to ground, and is protected from transient voltages typical when switching inductive devices - relays, solenoids, motors.
- LED Indicators on all I/O channels
- 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.

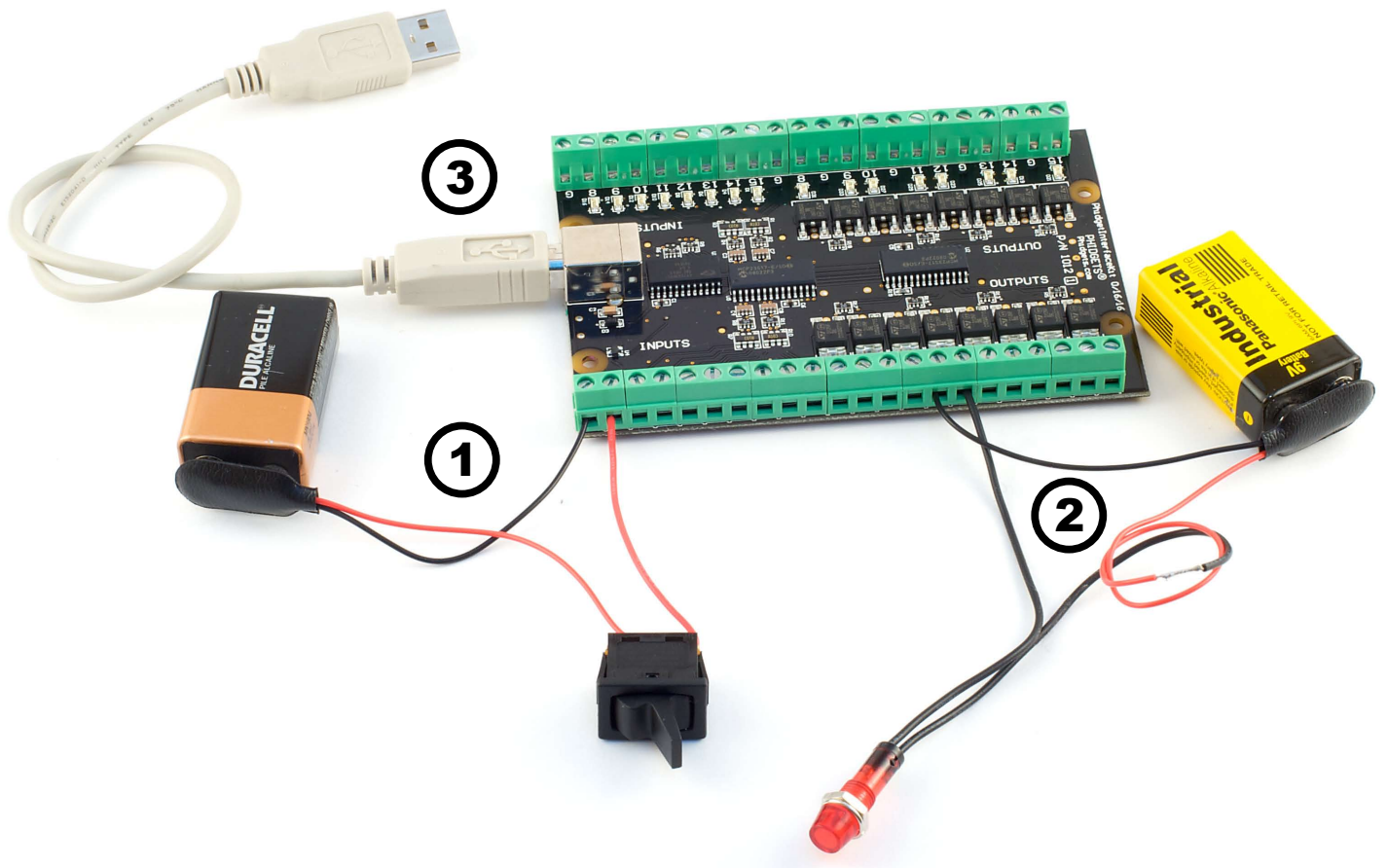
Installing the hardware

The kit contains:

- A Phidget Interface Kit 0/16/16
- A USB cable

You will also need:

- Two 9V batteries
- Two 9V battery connectors
- A switch, a piece of wire
- An incandescent bulb




1. Connect the black wire (-) from the battery to the ground terminal (G). Connect the red wire (+) to one of the switch terminals. Connect the other switch terminal to the digital input terminal block number 0 using a piece of wire.
2. Connect the black wire (-) from the battery to the ground terminal (G). Connect the red wire (+) to one of the bulb wires. Connect the other bulb wire the digital output terminal block number 3.
3. Connect the board to the PC using the USB cable.

Downloading and Installing the software

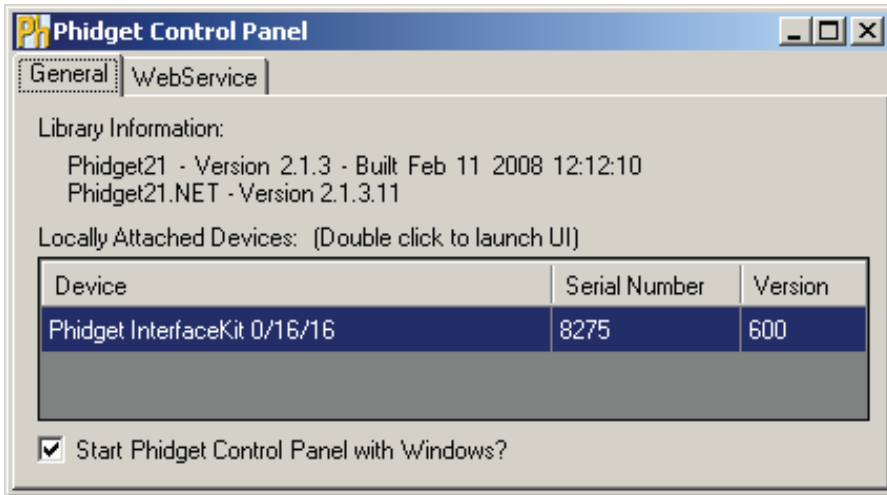
If you are using Windows 2000/XP/Vista

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

Download and run Phidget21.MSI

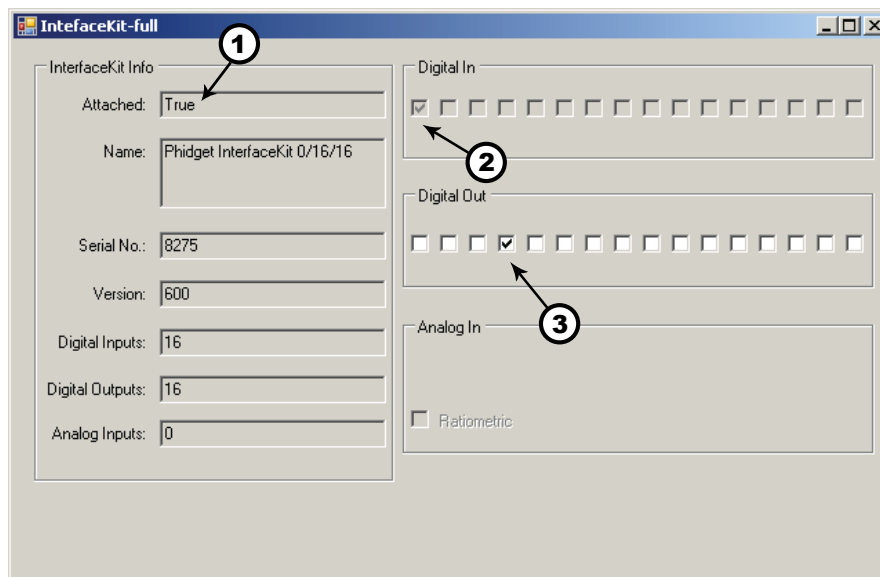
You should see the  icon on the right hand corner of the Task Bar.

Testing the PhidgetInterfaceKit 0/16/16 Functionality



Double Click on the  icon to activate the Phidget Control Panel and make sure that **PhidgetInterfaceKit 0/16/16** is properly attached to your PC.

1. Double Click on PhidgetInterfaceKit 0/16/16 in the Phidget Control Panel to bring up interfaceKit-full and check that the box labelled Attached contains the word True.



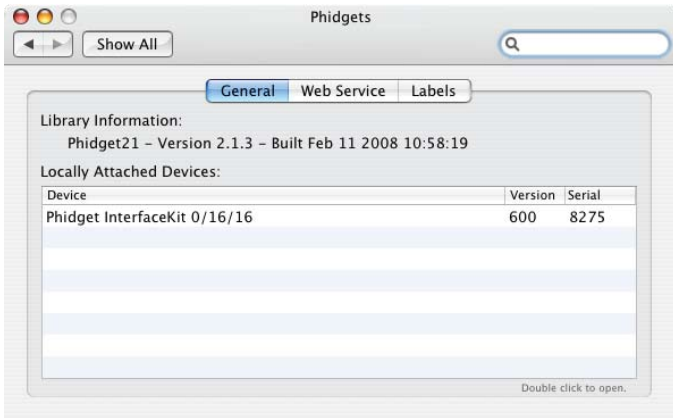
2. To test the digital input, toggle the switch on and off. When on, a tick mark will appear in the Digital In box and The on-board LED will also turn on; the tick mark will disappear when the switch is off and the on-board LED light will go off.
3. To test the digital output, put a tick mark in the digital out box and both the on-board LED and the incandescent bulb will turn on. If you click on the box again the tick mark will go away and both lights will turn off.

If you are using Mac OS X

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

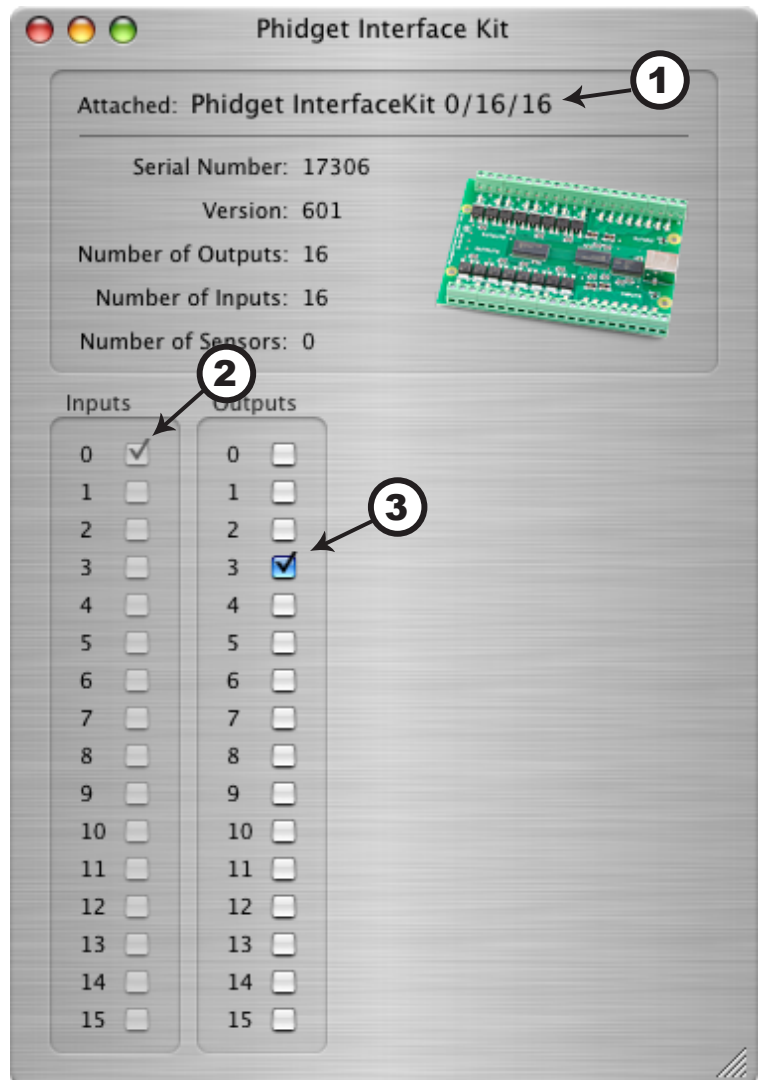
Download Mac OS X Framework

Testing the PhidgetInterfaceKit 0/16/16 functionality



Click on System Preferences >> Phidgets (under Other) to activate the Phidgets Preference Pane. Make sure that the PhidgetInterfaceKit 0/16/16 is properly attached.

1. Double Click on PhidgetInterfaceKit 0/16/16 in the Phidget Preference Pane to bring up the Phidget Interface Kit Example and check that the PhidgetInterfaceKit 0/16/16 is attached.
2. To test the digital input, toggle the switch on and off. When on, a tick mark will appear in the Inputs box and The on-board LED will also turn on; the tick mark will disappear when the switch is off and the on-board LED light will go off.
3. To test the digital output, put a tick mark in the Outputs box and both the on-board LED and the incandescent bulb will turn on. If you click on the box again the tick mark will go away and both lights will turn off.



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 besides 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#

```
/* - 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 ErrorHandler(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
static void ifKit_InputChange(object sender, InputChangeEventArgs e)
{
    Console.WriteLine("Input index {0} value (1)", e.Index, e.Value.ToString());
}

//Output change event handler...Display the output index and the new valu to
//the console
static void ifKit_OutputChange(object sender, OutputChangeEventArgs e)
{
    Console.WriteLine("Output index {0} value {0}", e.Index, e.Value.ToString());
}

//Sensor Change event handler...Display the sensor index and it's new value to
//the console

```

```

static void ifKit_SensorChange(object sender, SensorChangeEventArgs e)
{
    Console.WriteLine("Sensor index {0} value {1}", e.Index, e.Value);
}
}
}

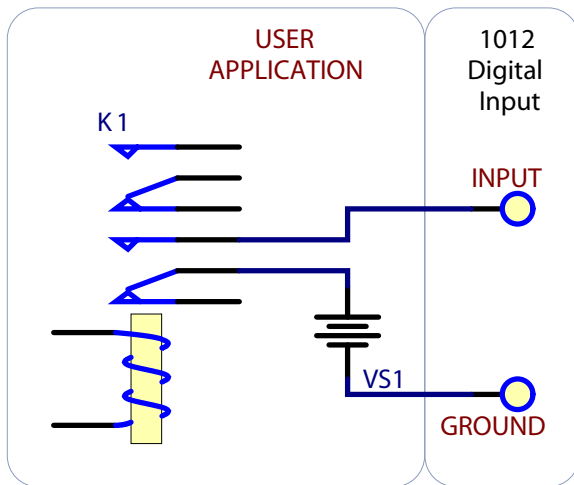
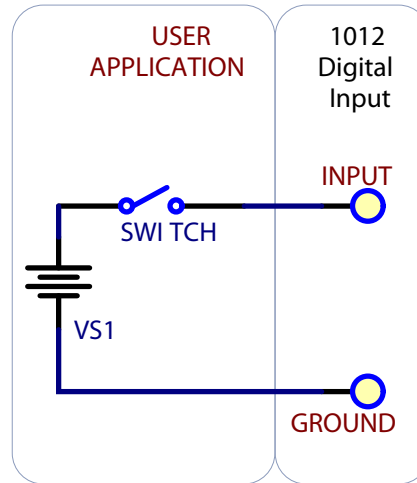
```

Digital Inputs

Using the Digital Inputs

Wiring a switch to a Digital Input

VSI should be 4-30V.

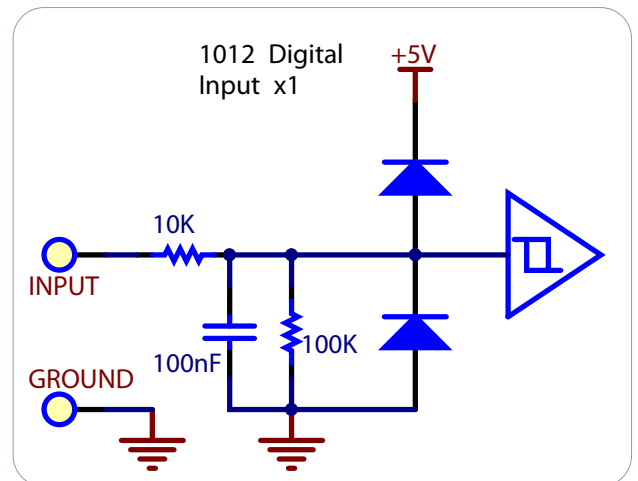


Monitoring the Position of a Relay

The relay contact causes the Digital Input to report TRUE. VSI should be 4-30V.

Functional Block Diagram

The Digital Inputs are capable of sensing up to 30VDC. A voltage of 4VDC to 30VDC will be read as True or logical 1; below 1VDC will be read as a False or logical 0. The input is high impedance, which means current flowing into the Phidget device will be limited. Ground terminals are provided in multiple locations along the input terminal strip; it is recommended that the ground terminal located nearest the input terminal be used.



Digital Input Hardware Filter

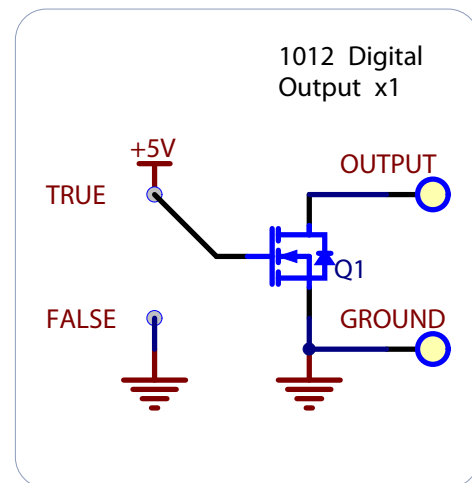
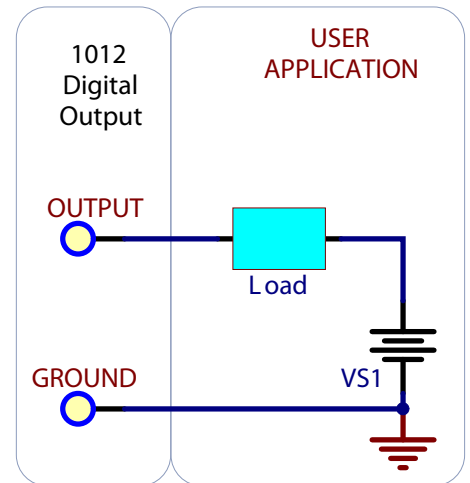
There is built-in filtering on the digital input, to eliminate false triggering from electrical noise. The digital input is first RC filtered by a 15K/100nF node, which will reject noise of higher frequency than 1KHz. This filter generally eliminates the need to shield the digital input from inductive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

Digital Outputs

Using the Digital Outputs

Using Digital Output as a Lowside switch for a load

Maximum Continuous current sink is 2 Amps. Load can be Relays, Solenoids, current limited LEDs, Bulbs, Motors, etc.



Functional Block Diagram

The Digital Outputs require an external voltage source to supply power, and act as a switch to ground. The outputs can sink up to 2A at 30V, and are designed to operate with DC voltage only. This type of switching output is often referred to as a low-side switch. It is common to use the Digital Outputs with electrical devices such as motors, lamps, relays, and solenoids. When using ground terminals, it is recommended that the ground terminal located nearest the output terminal be used. If highly inductive loads are used with the Digital Outputs, the use of a clamping diode is recommended to reduce noise.

Ground Protection

Ground terminals on the InterfaceKit share a common ground with USB ground. Because they are not internally isolated, these terminals will expose the USB ground potential of the PC

to which they are connected. Be sure you are completely familiar with any circuit you intend to connect to the InterfaceKit before it is connected. If a reverse voltage or dangerously high potential is applied to the input or output terminals, damage to the Phidget or the PC may result. Limit input and output voltages to 30VDC, and always observe correct polarity.

API Section

We document API Calls specific to the 1012. Functions common to all Phidgets, and functions not applicable to the 1012 are not covered here. This section is deliberately generic - for calling conventions in a specific language, refer to that language's API manual.

Functions

int InputCount() [get] : Constant = 16

Returns the number of digital inputs supported by this PhidgetInterfaceKit.

bool InputState(int InputIndex) [get]

Returns the state of a particular digital input. Digital inputs read True where they are activated and false when they are in their default state.

int OutputCount() [get] : Constant = 16

Returns the number of digital outputs supported by this PhidgetInterfaceKit.

bool OutputState (int OutputIndex) [get,set]

Sets/returns the state of a digital output. Setting this to true will activate the output, False is the default state. Reading the OutputState immediately after setting it will not return the value set - it will return the last state reported by the Phidget.

Events

OnInputChange(int InputIndex, bool State) [event]

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

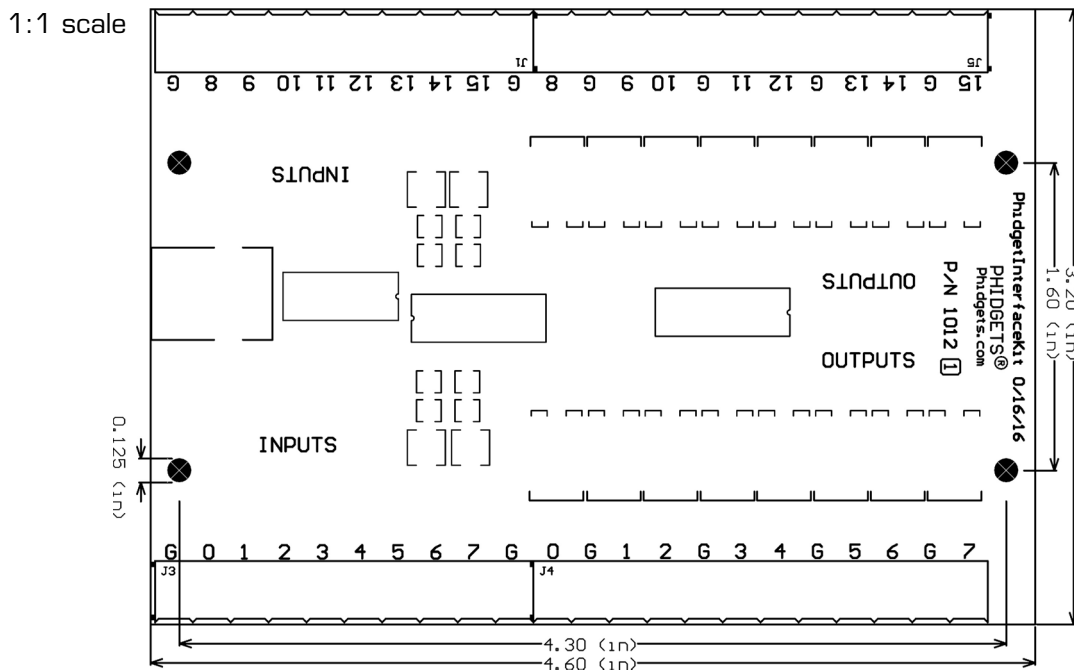
OnOutputChange(int OutputIndex, bool State), [event]

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

Device Specifications

| | |
|--|--------------------|
| Digital Input Impedance (-0.5V to +5.5V) | 110kΩ |
| Digital Input Impedance (>5.5V) | 10kΩ |
| Digital Output Impedance (on) | 0.2Ω |
| | |
| Digital Input Update Rate | 125 Updates/second |
| Digital Input Minimum Event Detection Time | 3mS |
| | |
| Digital Output Update Rate | 125 Updates/second |
| Digital Output Current Sinking (30V) | 2A max |
| | |
| USB Power Current Specification | 500mA max |
| Device Quiescent Current Consumption | 18mA |
| Device Active Current Consumption | 120mA max |

Mechanical Drawing



Product History

| Date | Product Revision | Comment |
|----------------|-------------------|-------------------------------|
| January 2003 | DeviceVersion 600 | Product Release |
| January 2004 | DeviceVersion 601 | Added State Echoing |
| September 2008 | Board Rev 1 | Added Digital Input Filtering |