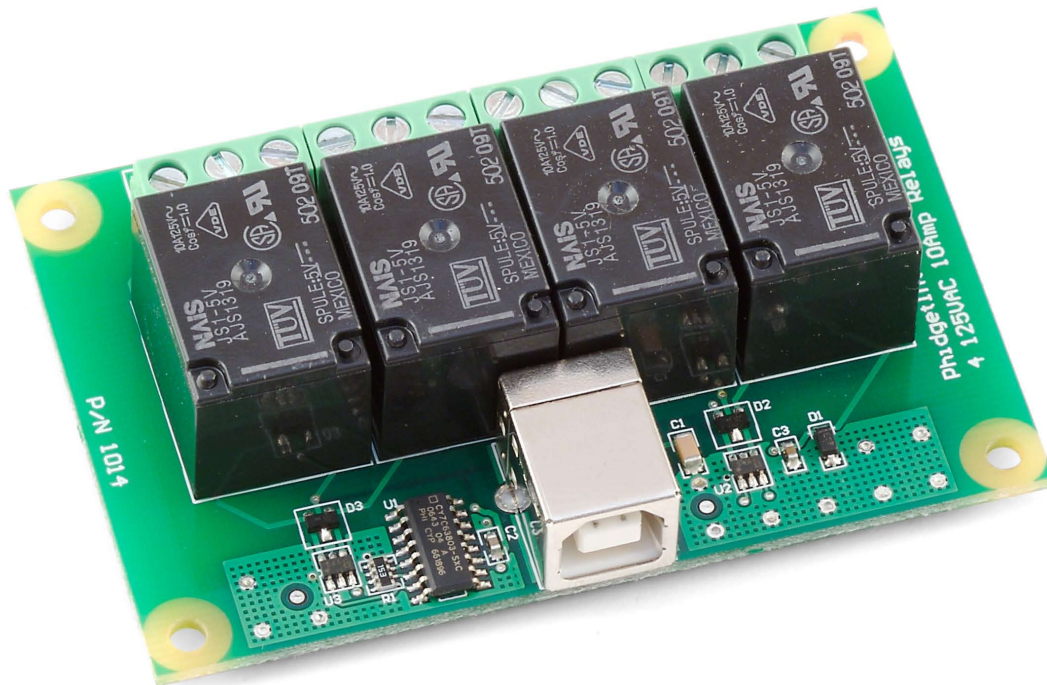


# 1014 - PhidgetInterfaceKit 0/0/4

## for Board Revision 0



## 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](http://www.phidgets.com).

# What Can the PhidgetInterfaceKit 0/0/4 Do?

The PhidgetInterfaceKit 0/0/4 allows you to digitally control 4 SPDT (Single Pole Double Throw) relay outputs. It provides a generic, convenient way to interface your PC with various higher-voltage devices.

## Getting Started

### Installing the hardware

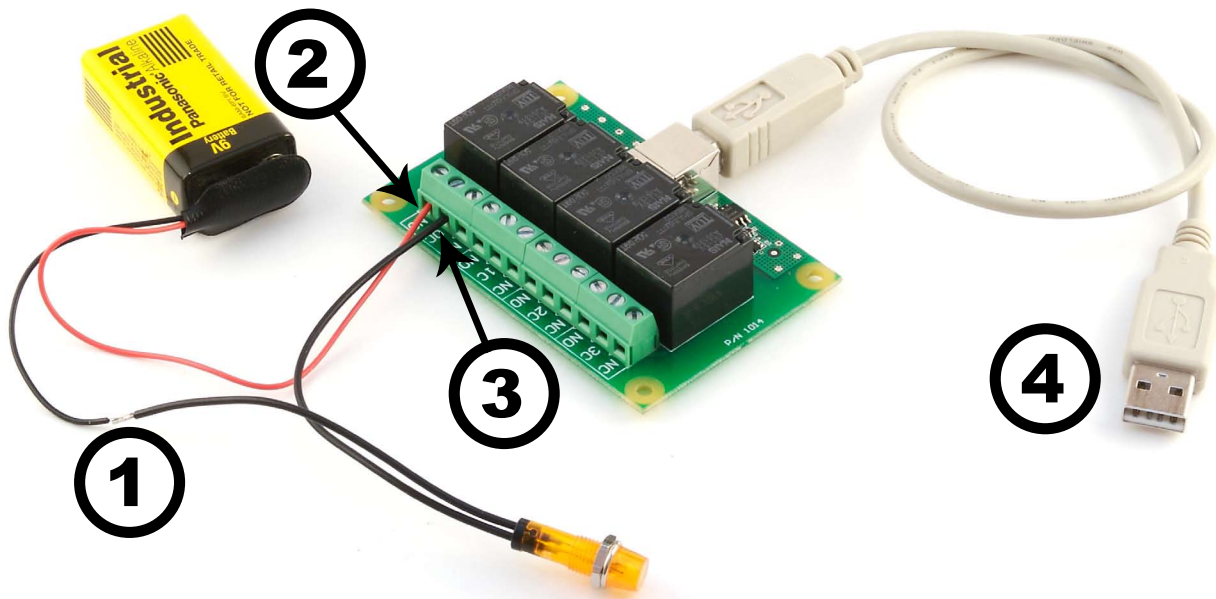
The kit contains:

- A PhidgetInterfaceKit 0/0/4
- A USB Cable

You will also need:

- A 9V battery
- A battery connector
- A 9V incandescent bulb with wires

### Connecting all the pieces




1. Connect the black/negative [-] wire from the battery connector to one of the bulb wires.
2. Connect the red/positive [+] wire from the battery connector to the NO (Normally Open) connector on the InterfaceKit.
3. Connect the other bulb wire to the OC (Common) connector on the InterfaceKit.
4. Connect the InterfaceKit to your PC using the USB cable.

# Downloading and Installing the software

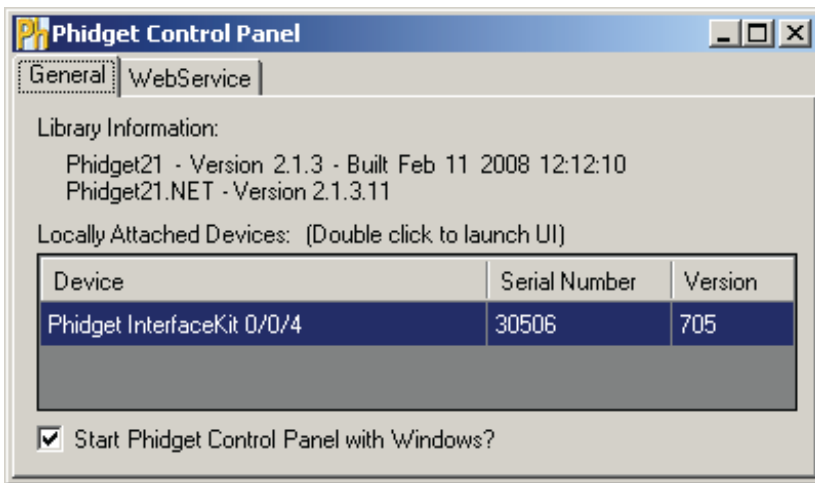
## If you are using Windows 2000/XP/Vista

Go to [www.phidgets.com](http://www.phidgets.com) >> Downloads >> Windows

Download and run Phidget21.MSI

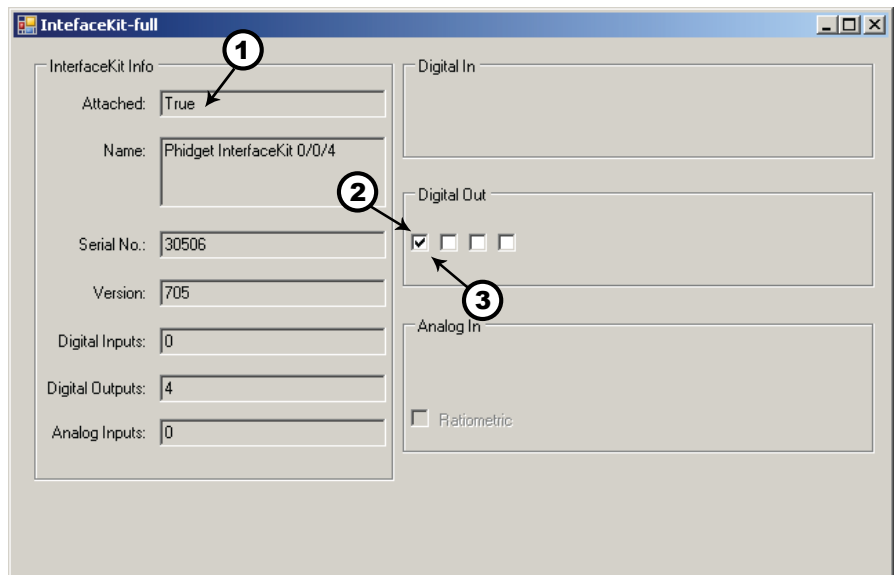
You should see the  icon on the right hand corner of the Task Bar.

## Testing the PhidgetInterfaceKit 0/0/4 Functionality



Double Click on the  icon to activate the Phidget Control Panel and make sure that **PhidgetInterfaceKit 0/0/4** is properly attached to your PC.

1. Double Click on **PhidgetInterfaceKit 0/0/4** in the Phidget Control Panel to bring up InterfaceKit-full and check that the box labelled Attached contains the word True.
2. Click on the first Digital Out box. A tick mark appears in the box and the bulb lights up. Click on the box again. The tick mark goes away and the light goes out. If you unplug the USB cable while the light is on, it will go off.
3. Move the positive (+) wire from NO to NC (normally closed). The light is now on when the Digital box has no tick mark. Clicking on the box turns the light off. If you unplug the USB cable when the light is on, it will stay on.

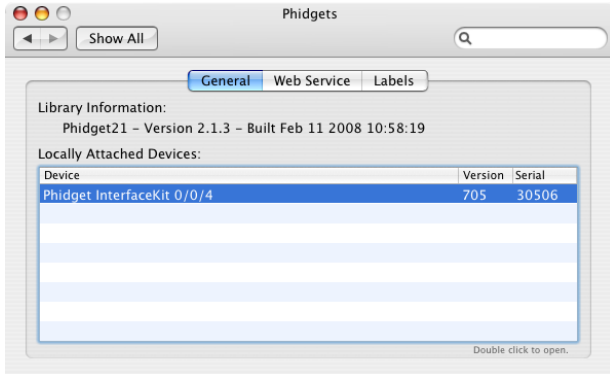


# If you are using Mac OS X

Go to [www.phidgets.com](http://www.phidgets.com) >> downloads >> Mac

Download Mac OS X Framework

## Testing the PhidgetInterfaceKit 0/0/4 functionality

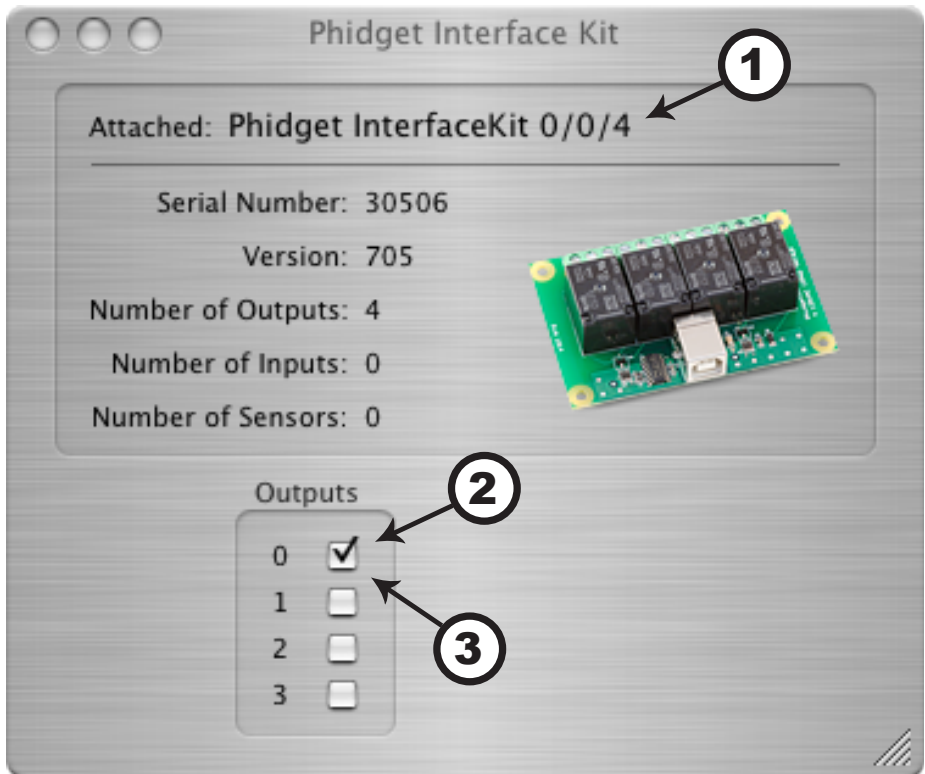


Click on System Preferences >> Phidgets (under Other) to activate the Phidgets Preference Pane. Make sure that the **PhidgetInterfaceKit 0/0/4** is properly attached.

1. Double Click on **PhidgetInterfaceKit 0/0/4** in the Phidget Preference Pane to bring up Phidget Interface Kit Example and check that the PhidgetInterfaceKit 0/0/4 is attached.

2. Click on Outputs Box O. A tick mark appears in the box and the bulb lights up. Click on the box again. The tick mark goes away and light goes out. If you unplug the USB cable while the light is on, it will go off.

3. Move the positive (+) wire from NO to NC (normally closed). The light is now on when the Outputs box has no tick mark. Clicking on the box turns the light off. If you unplug the USB cable when the light is on, it will stay on.



# Programming a Phidget

## Where to get information

- Go to [www.phidgets.com](http://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 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 InterfaceKit 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 basic 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
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 value to
    //the console
    static void ifKit_OutputChange(object sender, OutputChangeEventArgs e)
    {
        Console.WriteLine("Output index {0} value {0}", e.Index, e.Value.ToString());
    }

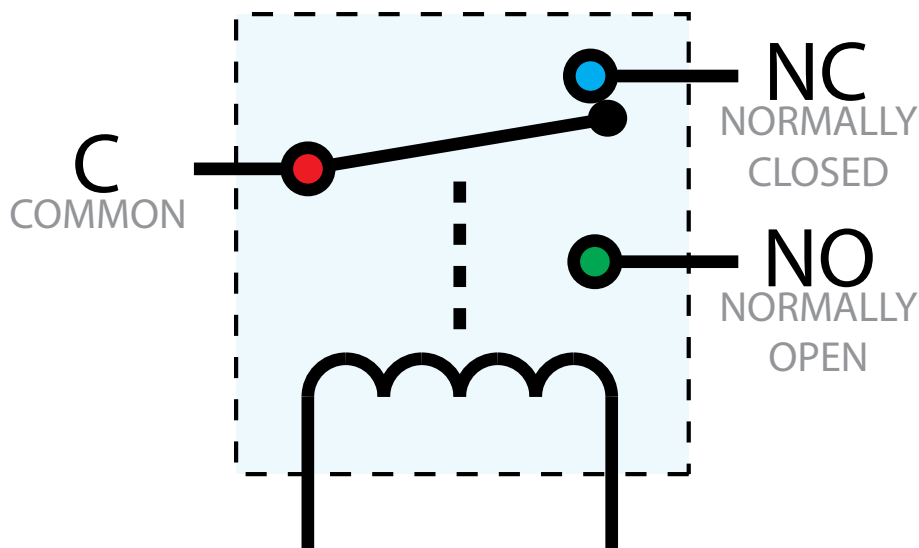
    //Sensor Change event handler...Display the sensor index and it's new value to
    //the console
    static void ifKit_SensorChange(object sender, SensorChangeEventArgs e)
    {
        Console.WriteLine("Sensor index {0} value {1}", e.Index, e.Value);
    }
}
}

```

## Learning more ...

- Check out the forums
- Check out the Phidgets projects

## Technical Section



### Relays

A relay is an electrically-controlled switch. Although many types of electrical switches exist, a relay's mechanical nature gives it the advantage of reliability and current-switching capacity. The main disadvantage to using mechanical relays is their limited life-span, as opposed to solid state relays who do not suffer from this drawback.

### Using a Digital Output Relay

Relays have a connection scheme determined by the arrangement of contacts within the relay. Because relays are a type of switch, they are defined in the same way other electromechanical switches are defined.

In switch schemes, the number of poles represents the number of common terminals a switch has, and the number of throws represents the number of switchable terminals that exist for each pole. The relays used in the InterfaceKit O/O/4 are SPDT relays: single pole, double throw. The internal construction of this type of relay is depicted in the diagram above. Many other types of relays exist: SPST, DPDT, and DPST, to name a few.

In an SPDT relay, one of the throw terminals is labelled Normally Closed (NC), and the other is labelled Normally Open (NO). As the name indicates, the normally closed terminal is the terminal connected to common when the relay coil is not powered. When the relay coil is energized by the relay control circuit, the electromagnetic field of the coil forces the switch element inside the relay to break its contact with the normally closed terminal and make contact with the normally open terminal. The switch element would then connect the normally open terminal and the common terminal.

### **Wetting Current**

When a relay is in one switch position for a period of time, oxidation of the open contact(s) can occur. Depending upon the internal coating material of the contacts, oxide films of varying density will be displaced upon the surface of open contacts; this film acts as an insulator to current flow. When the relay is switched, a certain amount of current flowing through the contacts, known as the wetting current, is required to remove the film of oxides and ensure proper conduction. Because of this requirement, these relays are not reliable for signal switching. See the device specification on page 10 for detailed requirements.

### **Load Noise**

If highly inductive loads are used with the InterfaceKit, it is recommended that a noise limiting component be used to prevent damage to the device. An MOV, TVS diode, or kickback diode (for DC applications) shunted across the load will assist in dissipating voltage transients.

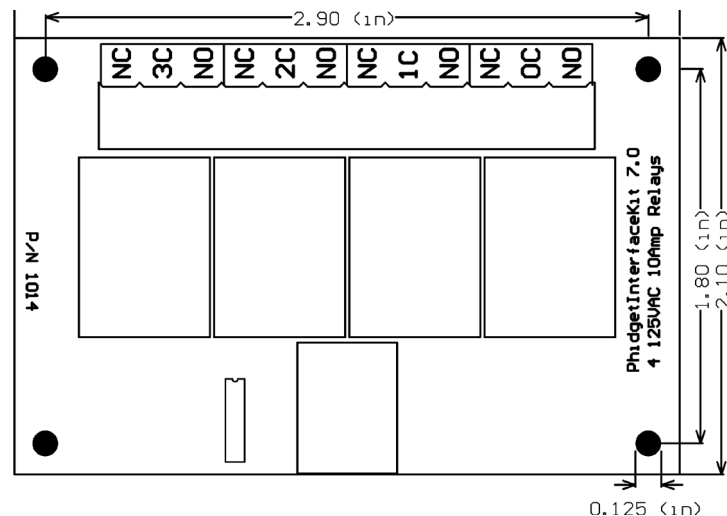


## Device Specifications

Contact Resistance (max)	120 mohms
Minimum Switching Current (Wetting Current)	100mA @ 5VDC
Maximum DC Switching Voltage	100 VDC
Maximum DC Switching Current	5 A
Maximum AC Switching Voltage	250 VAC
Maximum AC Switching Current	10 A
Operate Time	10 ms
Switching Speed (Contacts Per Minute)	20 cpm
Recommended Terminal Wire Size	14 - 26 AWG
Terminal Wire Strip Length	5 - 6mm (0.196" - 0.236")
USB-Power Current Specification	500mA max
Device Quiescent Current Consumption	14mA
Device Active Current Consumption	320mA max

## Mechanical Drawing

1:1 scale



## Product History

Date	Product Revision	Comment
August 2002	DeviceVersion 700	Product Release
January 2004	DeviceVersion 704	Added State Echoing