

Getting started with Phidgets in VB6

Environment and Libraries

First, we need to set up the proper environment and get the necessary files off the Phidgets website. Visit the drivers section at www.phidgets.com and get the latest:

- Phidget Framework

You will need the Phidget Framework to program with or use Phidgets. We also recommend that you download the following reference materials from the programming section:

- VB6 examples
- COM API Manual
- Programming Manual
- The Product Manual for your device

In the VB6 examples you can find more robust examples for each type of Phidget. The COM API manual contains calls and events for every type of Phidget and can be used as a reference. You can find a high level discussion about programming with Phidgets in general in the Programming Manual. The Product manual for your device also contains an API section that describes limitations, defaults, and implementation details specific to your Phidget. You may want to have the manuals and examples open while working through these instructions.

Setting up a Phidgets Project

The Phidget examples were written using Microsoft Visual Basic 6.0 and this tutorial assumes its use. To begin, launch VB6 and create a Standard EXE for our project. Then, place a TextBox (Text1) in the form designer for the purpose of capturing some simple output.

Coding For Your Phidget

Before you can use the Phidget, you must include the Phidget ActiveX objects in your project. This can be accomplished from the components controls screen (Project | Components...) by checking the box beside "Phidget Library 2.1", or by browsing to the location the framework was installed and choosing the Phidget21COM.dll.

Afterwards, the Phidget ActiveX object will need to be declared and then initialized. The simplest method is to place the Phidget ActiveX object from the palette directly on to your form.

Alternatively, you can dynamically create the ActiveX object in the code using Controls.Add(). You may need to uncheck 'Remove information about unused ActiveX Controls' in Project Options when using this if the form does not have the corresponding control placed.

```
Public WithEvents PhidgetInterfaceKit1 As PhidgetInterfaceKit

Set PhidgetInterfaceKit1 = Controls.Add("Phidget21COM.PhidgetInterfaceKit",
    "PhidgetInterfaceKit1")
```

The object name for any type of Phidget is listed in the API manual. Every type of Phidget also inherits functionality from the Phidget base class.

Connecting to the Phidget

The program can try to connect to the Phidget through an open call. Open will continuously try to connect to a Phidget, based on the parameters given, even trying to reconnect if it gets disconnected. This means that simply calling open does not guarantee you can use the Phidget immediately. We can account for a connection by using event driven programming and tracking the AttachEvents and DetachEvents, or by calling WaitForAttachment. WaitForAttachment will block indefinitely until a connection is made to the Phidget, or an optional timeout is exceeded.

```
Private Sub Form_Load()  
    PhidgetInterfaceKit1.Open  
    PhidgetInterfaceKit1.WaitForAttachment (3000)  
End Sub
```

The different parameters and open calls can be used to open the first Phidget of a type it can find, open based on a serial number, or even open across the network. The API manual lists all of the available modes that open provides. One important thing to remember is that when working with Phidgets, a local connection will reserve the device until closed. This prevents any other instances from retrieving data from the Phidget, including other programs. The one connection per device limit does not apply when exclusively using the Phidget Webservice.

You can call Close any time outside of the Phidget's own event handlers to end the connection.

Event Driven Programming

We recommend the use of event driven programming when working with Phidgets. In Visual Basic, we hook an event handler with the following code:

```
Private Sub PhidgetInterfaceKit1_OnSensorChange(ByVal Index As Long, ByVal SensorValue  
As Long)  
    Text1.Text = Index & ":" & SensorValue  
End Sub
```

With this method, the code inside onSensorChange will get executed every time the PhidgetInterfaceKit reports a change on one of its analog inputs. You can let the editor generate the procedure prototypes for you through the drop down menu at the top of the code window.

Some events such as Attach and Detach belong to the base Phidget object and thus are common to all types of Phidgets. Please refer to the API manual for a full list of events and their usage.

Working directly with the Phidget

Some values can be directly read and set on the Phidget, and inside polling loops used as an alternative to event driven programming. Simply use the instance properties such as SensorValue(Index as Long) or OutputState(Index as Long) for PhidgetInterfaceKits.

```
phid.OutputState(4) = True
```

Working with multiple Phidgets

Multiple Phidgets of the same type can easily be run inside the same program. In our case, it requires another PhidgetInterfaceKit instance to be defined and initialized. The new instance can then be set up, opened and used in the same process as the previous one.

If the application needs to distinguish between the devices, open can be called with the serial number of a specific Phidget.

Other Phidgets

The design given in this document can also be followed for almost all Phidgets. For example, if you were using a PhidgetRFID instead of an PhidgetInterfaceKit, you would place a PhidgetRFID ActiveX object instead of a PhidgetInterfaceKit. The methods and events available would change but they can be accessed in a similar manner.