

# Getting started with Phidgets in REALBasic

## 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](http://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:

- C API Manual
- Programming Manual
- The Product Manual for your device

The C 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

Applications using Phidget can be developed in REALbasic through the C API. This tutorial assumes the use of REALbasic 2009, other versions should work as well and would be set up in a similar manner. Although it's possible to import Phidgets as ActiveX controls, this is not recommended due to the way REALbasic handles them. To begin, launch REALbasic and create a new "Desktop Application". Then, place an EditField "EditField1" in the form designer for the purpose of capturing some simple output.

## Coding For Your Phidget

Before you can use the Phidget, you must declare each of the functions from the C library that the program will use. For example, you can declare at initialization (inside Windows1.Open) some C functions for a PhidgetInterfaceKit with the following syntax:

```
Declare Function CPhidget_open lib "phidget21" (phid as Ptr, serialNumber as Integer) as integer
```

```
Declare Function CPhidget_waitForAttachment lib "phidget21" (phid as Ptr, milliseconds as Integer) as Integer
```

```
Declare Function CPhidgetInterfaceKit_create lib "phidget21" (CPhidgetInterfaceKitHandle as Ptr) as Integer
```

```
Declare Function CPhidgetInterfaceKit_set_OnSensorChange_Handler lib "phidget21" (CPhidgetInterfaceKitHandle as Ptr, fptr as Ptr, UserPtr as Ptr) as Integer
```

The declaration closely follows the C API with a few minor changes. Generally, when any type

of PhidgetHandle is used, it's safe to change the type to Ptr. You will also need to copy the phidget21.dll file from the Phidget Framework directory into your project output directory for these declarations to work.

Afterwards, the Phidget object itself will need to be declared and then initialized. This can be done by declaring a property on Windows1 named "ifKitHandle" as a Ptr, calling create on an empty block of memory, and then setting the handle to the result.

```
Dim tmp As MemoryBlock
Dim result As Integer

tmp = NewMemoryBlock(1000)
Call CPhidgetInterfaceKit_create(tmp)
ifKitHandle = tmp.Ptr(0)
```

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.

```
result = CPhidget_open(ifKitHandle, -1)
if CPhidget_waitForAttachment(ifKitHandle, 3000) <> 0 Then
    MsgBox("Can't find a PhidgetInterfaceKit")
else
    'More code goes here
end if
```

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.

## Event Driven Programming

We recommend the use of event driven programming when working with Phidgets. In Real Basic, a few steps will need to be followed to properly hook C events. Event handler methods with matching definitions must be created inside their own module (Project | Add | Module).

```
Function ifKit_OnSensorChange(ByVal phid As Ptr, ByVal userPtr As Ptr, ByVal index
As Integer, ByVal sensorValue As Integer) As Integer
    #pragma X86CallingConvention StdCall
    Window1.EditField1.text = Str(index) + ":" + Str(sensorValue)
End Function
```

In this example, a new module "Module1" was created and given the `ifKit_OnSensorChange` method. Note that the handler must have `"#pragma X86CallingConvention StdCall"` as the first line. Inside `Window1.Open` we can then link to the method we wrote with a call to `CPhidgetInterfaceKit_set_OnSensorChange_Handler`.

```
Call CPhidgetInterfaceKit_set_OnSensorChange_Handler(ifKitHandle, AddressOf Module1.ifKit_OnSensorChange, nil)
```

With this, the code inside `ifKit_onSensorChange` will get executed every time the `InterfaceKit` reports a change on one of its analog inputs.

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 declare the C function before using calls such as `CPhidgetInterfaceKit_getSensorValue` or `CPhidgetInterfaceKit_setOutputState` for `InterfaceKits`.

```
Declare Function CPhidgetInterfaceKit_getSensorValue lib "phidget21" (phid as Ptr, index as Integer, sensorValue as Ptr) as Integer
```

```
result = CPhidgetInterfaceKit_getSensorValue(ifKitHandle, 0, tmp)
Dim sensorValue As Integer = tmp.Long(0)
Window1.EditField1.Text = Str(sensorValue)
```

## Working with multiple Phidgets

Multiple `Phidgets` of the same type can easily be run inside the same program. In our case, it requires another instance of a `PhidgetInterfaceKit` to be defined and initialized. The new instance can 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 `Interfacekit`, you would call `CPhidgetRFID_create` instead of `CPhidgetInterfaceKit_create`. The methods and events available would change but they can be used in a similar manner.