

Getting started with Phidgets in Adobe Director

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:

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

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

This tutorial was written for Adobe Director 11.5 in Lingo and assumes its use. Other versions of director should work as well, and each would be set up in a similar manner.

First launch Director and generate a new movie file with a descriptive name such as PhidgetTest. Next, insert an ActiveX control (Insert | Control | ActiveX...) and add the "PhidgetInterfaceKit Class". The Phidget object will be added to your cast, and you can then drag and drop it into the stage to create a sprite for it. Add a field control to your movie (Insert | Control | Field) for the purpose of capturing some simple output, as well as a button (Insert | Control | Button) labelled "Open" to get things started.



Coding For Your Phidget

In this tutorial, the Phidget ActiveX object is assumed to be sprite 1 and the text field was given the member name "OutputField" to help distinguish it from the other objects.

By double clicking an ActiveX object, Director will fully list its functionality. In Lingo, you can apply the functionality for an object's sprite through CallStrings. The object name for any type of Phidget is also listed in the API manual. Every type of Phidget also shares some functionality from the base Phidget class.

Connecting to the Phidget

The program can try to connect to the Phidget through a call to open. 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. Here, the Phidget is opened on clicking the Open button inside the mouseUp event.

```
on mouseUp me
    sprite(1).CallString("Open()")
    sprite(1).CallString("WaitForAttachment(3000)")
end
```

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 use CallString("Close") at any time outside of the Phidget's own event handlers to close the connection.

Event Driven Programming

We recommend the use of event driven programming when working with Phidgets. This allows the program to execute other tasks until the Phidget fires a new event. In Adobe director, you can hook an event handler inside a script for the Phidget object with the following code:

```
on OnSensorChange(Index, SensorValue) me
    member("OutputField").text = string(Index) & ": " & string(SensorValue)
end
```

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

Some events such as OnAttach and OnDetach belong to a base Phidget object and 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 CallString such as SensorValue(Index) or OutputState(Index, OutputState) for InterfaceKits.

```
sprite(1).CallString("OutputState(0,1)")
```

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 ActiveX object to be created and placed. The new object 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 Interfacekit, 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.