

Getting started with Phidgets in LiveCode

Environment and Libraries

Phidgets supports development in LiveCode 4.5 for InterfaceKit 8/8/8 devices on local connections in Windows XP SP3 and Mac OS X 10.6.4 environments. First, we need to set up the proper environment and get the necessary files off the Phidgets website. Visit the drivers and programming section at <http://www.phidgets.com> and get the latest:

- Phidget21 Installer
- Externals.txt and phidgets_livecode.dll(for Windows)
- Externals.txt and phidgets_livecode.bundle(for Mac OS X)

You will need to install the latest Phidget21 libraries for your platform. We also recommend that you download the following reference materials from the programming section:

- LiveCode API Manual
- Programming Manual
- The Product Manual for your device
- LiveCode examples

The LiveCode API manual documents calls for the InterfaceKit 8/8/8 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 LiveCode. This tutorial assumes the use of the LiveCode 4.5 IDE in a Windows XP SP3 or a Mac OS 10.6.4 environment, other recent versions should work as well and would be set up in a similar manner. To begin,

- The first step is to integrate the phidgets_livecode.dll(Windows) or phidgets_livecode.bundle(Mac OS X) and the Externals.txt into the LiveCode IDE. Locate the "Externals" directory under the LiveCode User Extensions folder. On a Windows machine, you can verify the location of the Users Extensions folder by checking the User Extensions field (from Edit | Preferences | Files & Memory | User Extensions in the LiveCode 4.5 IDE). In Mac OS X, the path is: LiveCode | Preferences | Files & Memory | User Extensions. If the folders do not exist, please create it. Place the phidgets_livecode.dll(or phidgets_livecode.bundle) and the Externals.txt inside the "Externals" folder. For more information on the User Extensions Folder, please consult <http://www.runrev.com>.
- Launch the LiveCode IDE, and create a "New MainStack".
- Insert Text Entry Fields on the card for the purpose of capturing output.

The project now has access to Phidgets and we are ready to begin coding.

Coding For Your Phidget

The LiveCode libraries does not implement the full Phidget API - some function calls and Phidget classes are not be supported. Please refer to the LiveCode API manual for a list of supported classes and functions. The Phidget object will need to be declared and initialized. For example, we can declare and initialize an instance of a PhidgetInterfaceKit inside our script with:

```
//Declare a variable to store the InterfaceKit object
global ifKit

//Create the InterfaceKit object and store it in the variable, "ifKit"
get phidgetsInterfaceKit_create("ifKit")
```

The PhidgetInterfaceKit also inherits functionality from the Phidget base class.

Connecting to the Phidget

The program needs to try and connect to the Phidget through an open call. The open will tell the program to 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 handle this 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.

```
get phidgets_open(ifKit, -1) // -1 to open any Phidget
get phidgets_waitForAttachment(ifKit, 1000)
```

The open call can specify a serial number to open a specific Phidget, or specify -1 to open any Phidget. The API manual lists all of the available modes that open provides.

At the end of your program, don't forget to call close and delete to free any locks on the Phidget.

```
get phidgets_close(ifKit)
get phidgets_delete(ifKit)
```

Event Driven Programming

We recommend the use of event driven programming when working with Phidgets. We can setup an event handler with the following code:

```
get phidgetsInterfaceKit_set_OnSensorChange_Handler(ifKit, "SensorChangeHandler")
```

The SensorChangeHandler command is defined as follows:

```
command SensorChangeHandler ifKit pIndex pValue
    put "Sensor#: " & pIndex & " , Value: " & pValue & return after field
        "currSensorValues"
end SensorChangeHandler
```

In this function, the code inside SensorChangeHandler will get executed every time the PhidgetInterfaceKit reports a change on one of its analog inputs. Please refer to the LiveCode API

manual for a full list of events and their usage.

Working directly with the Phidget

Most values can be read directly as an alternative to event driven programming. Simply use the LiveCode API functions, such as `phidgetsInterfaceKit_getSensorValue` for `PhidgetInterfaceKits`.

```
repeat with currSensor=0 to 7
    local sensorValue
    get phidgetsInterfaceKit_getSensorValue(ifKit, currSensor, "sensorValue")
    put "Sensor: " & currSensor & ", Value: " & sensorValue into
        sensorValueList[currSensor]
end repeat
combine sensorValueList using return
put sensorValueList into field "sensorValueField"
```

Above, `phidgetsInterfaceKit_getSensorValue` copies the sensor value into the variable "sensorValue", and places the value in the field "sensorValueField".

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

At this time, only the `PhidgetInterfaceKit 8/8/8` is supported. For more information on how you can leverage LiveCode with other Phidget devices, please refer to the "Extending LiveCode to other Phidgets" section.

Creating Standalone Applications

Packaging standalone applications require the `phidgets_livecode.dll`(Windows) or `phidgets_livecode.bundle`(Mac OS X) to be attached with the executable. Please ensure that the `phidgets_livecode.dll`(or `phidgets_livecode.bundle`) and `Externals.txt` are in the "Runtime" directory under the LiveCode User Extensions folder. If the directories do not exist, please create it. In addition, include the `phidgets_livecode` external in the Standalone Application Settings of the LiveCode IDE. For more information on the exact procedure for packaging externals with standalones, please consult <http://www.runrev.com>.

Extending LiveCode to other Phidgets

A C++ external wrapper was created which allows the InterfaceKit 8/8/8 to be programmable in LiveCode. It is possible to extend this external to other Phidgets as well. The goal of this section is to equip the reader with knowledge so that they can understand the relationship between LiveCode and the Phidgets framework. It is intended to provide guidance for readers who wish to extend LiveCode to other Phidgets. This guide uses C++, Microsoft Visual Studio 2005(for Windows XP SP3), XCode 3.2.4(for Mac OS X 10.6.4), LiveCode 4.5 IDE and it is assumed that the reader has a solid understanding of these technologies. For your reference, there is also Visual Studio and XCode project examples in the LiveCode examples under the programming section at <http://www.phidgets.com>.

The first step is to download LiveCode Externals Creator SDK. The SDK can be found in the Externals tutorial section at <http://www.runrev.com>. Run the SDK, select C++ as the language, enter the project name, select the platform and specify the LiveCode installation path. Hit generate to create a skeleton project for the external.

The next step is to setup the environment to allow for access to the Phidget21 function calls.

In Visual Studio, you will need the latest Phidget21 Windows library and header - these are installed into "Program Files/Phidgets" by the Phidgets installer. Place the phidget21.lib and phidget21.h in the Visual Studio project directory. Be sure to open the properties window, navigate to Configuration Properties | Linker | Input and add "phidget21.lib" to the additional dependencies section. Also, include a reference to the library header in the source code.

In XCode, open the newly created .xcodeproj project into XCode. Add the Phidget21 framework to the project and include the a reference to the library header in the source code.

Now, we are ready to create external functions. LiveCode will have to acknowledge the existence of any external function. Function declarations are placed inside the USER DECLARATIONS section of the the code. Here, a call to the function phidgetsInterfaceKit_create in LiveCode will be mapped to a call to the C++ function phidgetsInterfaceKit_create.

```
EXTERNAL_DECLARE_FUNCTION("phidgetsInterfaceKit_create", phidgetsInterfaceKit_create)
```

For example, LiveCode will use the following lines of code to call the external function phidgetsInterfaceKit_create,

```
local ifKit
get phidgetsInterfaceKit_create("ifKit")  //"ifKit" is the name of the variable phid
```

The result of get phidgetsInterfaceKit_create("ifKit") will be 0 or the error code depending on whether the external function succeeded or failed, respectively.

Now, we implement the `phidgetsInterfaceKit_create` C++ function.

```
void phidgetsInterfaceKit_create(char *p_arguments[], int p_argument_count, char **r_
result, Bool *r_pass, Bool *r_err)
{
    char outcome_str[8];
    CPhidgetInterfaceKitHandle ifKit = (CPhidgetInterfaceKitHandle) strtoul(p_
arguments[0], NULL, 16);

    int outcome=CPhidgetInterfaceKit_create(&ifKit);
    if(outcome==0) {
        char str[90];
        int t_success;

        /* Converts to Hexadecimal format */
        sprintf(str,"%08x",ifKit);

        /* Passes back to LiveCode a reference of the InterfaceKit handle */
        SetVariable(p_arguments[0], str, &t_success);
    }

    sprintf(outcome_str, "%d", outcome);
    *r_result = strdup(outcome_str);
    *r_err = False;
    *r_pass = False;
    return;
}
```

As you can see, this function follows the same format as a typical external function. `p_arguments[]` stores the arguments that are passed from the LiveCode function. First, we allocate memory to store a `CPhidgetInterfaceKitHandle` and store its address in `ifKit`. Next, we call the C function `CPhidgetInterfaceKit_create`, create the `PhidgetInterfaceKit` handle and return the error code to `outcome`. If `outcome` is 0, the address of `ifKit` is converted into hexadecimal form. The hexadecimal form is then stored in the LiveCode variable "ifKit", so that LiveCode has a reference to the newly created `InterfaceKit` handle. Lastly, the result of this external function is stored in `*r_result`, which is passed back as the return value for the LiveCode function.

It is also possible to implement Phidgets events with LiveCode. We will implement the `phidgetsInterfaceKit_set_OnSensorChange_Handler` function. Again, we declare the existence of the function:

```
EXTERNAL_DECLARE_FUNCTION("phidgetsInterfaceKit_set_OnSensorChange_Handler",
phidgetsInterfaceKit_set_OnSensorChange_Handler)
```

In LiveCode, we use the following code to call the external function:

```
get phidgetsInterfaceKit_set_OnSensorChange_Handler(ifKit, "sensorChange")
```

and "sensorChange" is the name of the LiveCode command to be called whenever a digital input changes. This command must be in the script of the card that called phidgetsInterfaceKit_set_OnSensorChange_Handler or its stack. The form of the command is as follow:

```
command sensorChange phid pIndex pValue
    ...
end sensorChange
```

Now, we implement the phidgetsInterfaceKit_set_OnSensorChange_Handler C++ function.

```
void phidgetsInterfaceKit_set_OnSensorChange_Handler(char *p_arguments[], int p_argument_
count, char **r_result, Bool *r_pass, Bool *r_err)
{
    CPhidgetInterfaceKitHandle ifKit = (CPhidgetInterfaceKitHandle)strtoul(p_
arguments[0], NULL, 16);

    char* str=(char *)malloc(90);
    str=strdup(p_arguments[1]);

    int outcome=CPhidgetInterfaceKit_set_OnSensorChange_Handler((CPhidgetInterfaceKitHa
ndle)ifKit, &SensorChangeHandler, str);

    returnOutcome(outcome, r_result, r_pass, r_err);
}
```

and the callback function

```
int __stdcall SensorChangeHandler(CPhidgetInterfaceKitHandle ifKit, void *usrptr, int
Index, int Value)
{
    int t_success;
    char * scmMsg;
    scmMsg=NULL;
    scmMsg = (char *)malloc(512);

    sprintf(scmMsg, "send \"%s %08x, %d, %d\" to current card", (char *)usrptr, ifKit,
Index, Value);
    SendCardMessage(scmMsg, &t_success);

    if (scmMsg != NULL)
        free(scmMsg);

    return 0;
}
```

Note: For Mac OS X, __stdcall is not needed.

Here, we use `SendCardMessage` to run the LiveCode command "sensorChange". The message will be passed to the card that called `phidgetsInterfaceKit_set_OnSensorChange_Handler` before passing through the stack in the message hierarchy.

Other Phidgets and functions can be extended in a similar manner. The C API manual contains calls and events for every type of Phidget.