How to Use the ThinkGear API in Xcode (Mac OS X)

June 28, 2009

NeuroSky®
Brain-Computer Interface Technologies

The NeuroSky product families consist of hardware and software components for simple integration of this bio-sensor technology into consumer and industrial end-applications. All products are designed and manufactured to meet exacting consumer specifications for quality, pricing, and feature sets. NeuroSky sets itself apart by providing building-block component solutions that offer friendly synergies with related and complementary technological solutions.

Contents

Introduction

Loadable modules (containing dynamic libraries or plugins) on Mac OS X are often packaged as bundles. These are generally analogous to `.dll` files in Windows or `.so` files in Linux and other *NIX platforms, though bundles also provide a richer set of functionality, e.g. facilities for loading non-executable assets such as localization strings or images.

Developers that want to integrate ThinkGear functionality into their OS X applications should utilize the `CFBundle` API in the Core Foundation framework to hook into `ThinkGear.bundle`. This document will describe the process of getting your Xcode project up and running with ThinkGear.

Note: The `NSBundle` API in the Cocoa framework applies strictly to bundles containing Objective-C classes. Since ThinkGear is a C-only API, discussions of `NSBundle` are inappropriate in this context.

Setting up Xcode

The only requirement for loading `ThinkGear.bundle` is that the Core Foundation framework be included in your project's list of external frameworks and libraries. This can be done by right-clicking on the **External Frameworks and Libraries** folder in your Xcode project window.

Then, choose **Add**, then **Existing Frameworks…**. Look for the `CoreFoundation.framework` folder in the directory browser, and click **Add**. The image below shows what your project window should look like once the Core Foundation framework has been added.

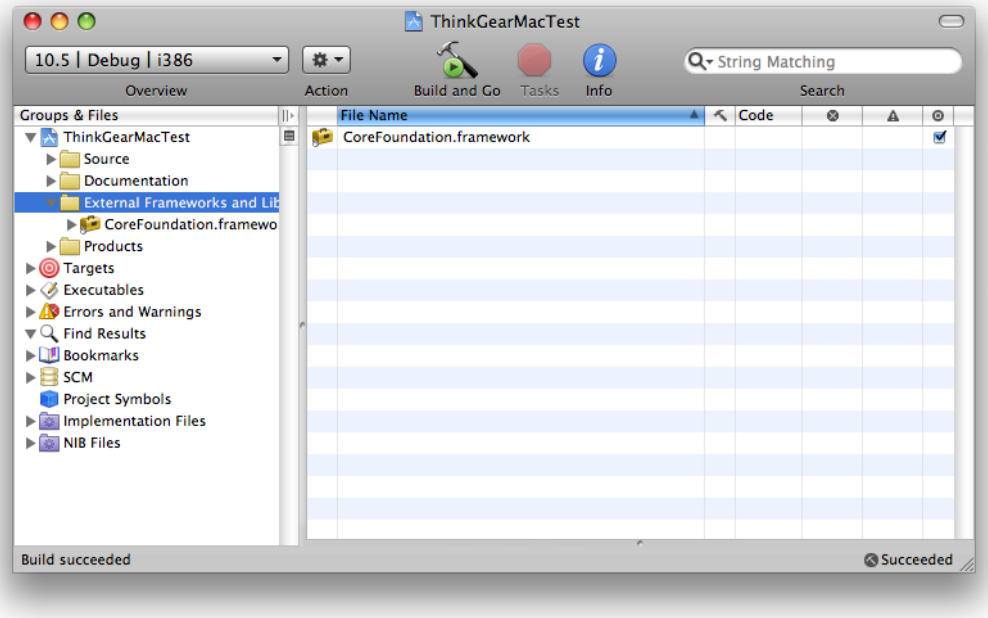

Figure 2.1: Xcode project window

Importing ThinkGear Functions

At the top of your header or implementation, you should include the Core Foundation library:

```
#include <CoreFoundation/CoreFoundation.h>
```

Before importing the functions, a bundle reference (`CFBundleRef`) must first be created for the bundle. This is constructed from a path describing the location of the bundle, which is encapsulated in a `CFURLRef` object. Let's first declare these objects.

```
CFURLRef bundleURL;
CFBundleRef thinkGearBundle;
```

And now, to instantiate them:

```
bundleURL = CFURLCreateWithFileSystemPath(kCFAllocatorDefault,
                                          CFSTR("ThinkGear.bundle"),
                                          kCFURLPOSIXPathStyle,
                                          true);

thinkGearBundle = CFBundleCreate(kCFAllocatorDefault, bundleURL);
```

`CFBundleCreate` returns `NULL` if the `bundleURL` points to an invalid bundle, so it's a good idea to check that for validity before continuing. Note that the path above is a relative path, so the executable will need the bundle to be located in the same directory. Apple provides documentation on different ways of locating bundles.

We then need to declare some function pointers that reference the functions inside `ThinkGear.bundle`. It is recommended to use the same naming scheme for the functions as is used in the API. A few examples are provided below for clarity. Refer to `ThinkGear.h` (provided in the ThinkGear SDK) for the function prototypes.

```
int (*TG_GetDV)()    = NULL;     // TG_GetDriverVersion
int (*TG_GetNCId)() = NULL;     // TG_GetNewConnectionId
int (*TG_Connect)(int, const char *, int, int) = NULL;
```

Finally, we'll want to create the references to the ThinkGear functions. This is done using the `CFBundleGetFunctionPointerForName` function, which takes the bundle reference as one of its parameters. This should be done for any ThinkGear functions that you plan on using in your application.

```
TG_GetDV   = (void*)CFBundleGetFunctionPointerForName(thinkGearBundle,
                                                      CFSTR("TG_GetDriverVersion"));
TG_GetNCId = (void*)CFBundleGetFunctionPointerForName(thinkGearBundle,
                                                      CFSTR("TG_GetNewConnectionId"));
TG_Connect = (void*)CFBundleGetFunctionPointerForName(thinkGearBundle,
                                                      CFSTR("TG_Connect"));
```

Before using these imported functions, it is prudent to check that they were successfully imported.

```
if(!TG_Connect)
   return -1;
```

Before your application quits (or when you're done using the functions), you'll need to release the allocated Core Foundation objects; namely, the `CFURLRef` and `CFBundleRef` objects. This is effectively equivalent to an object destructor.

```
CFRelease(bundleURL);
CFRelease(thinkGearBundle);
```

Using Imported ThinkGear Functions

The imported functions can be used as if they were normally declared and implemented in your code, e.g.

```
int retVal = TG_Connect(connectionID, "/dev/tty.MindsetMSEM1-DevB-1", 9600, 0);
printf("TG_Connect returned: %d\n", retVal);
```

Conclusion

By reading this document, you have familiarized yourself on how to integrate the ThinkGear library into your OS X application. A sample Xcode project, implementing a simple command-line data streamer for the headset, is included in the MindKit SDK.

References

- http://developer.apple.com/DOCUMENTATION/CoreFoundation/Conceptual/CFBundles/CFBundles.htm
- ThinkGear API and Reference Manual
- ThinkGear API MacOSX Example