# ThinkGear API Mac OS X example

```
/**
 * This program serves as a simple example of how one can use
ThinkGear.bundle inside their Core Foundation
 * (e.g. Cocoa and Carbon-based) apps. For more details on OS X bundles,
read:
 *
http://developer.apple.com/DOCUMENTATION/CoreFoundation/Conceptual/CFBundles
/CFBundles.html
 *
 * Or check the "How to use the ThinkGear API in Xcode (Mac OS X)" document
in the ThinkGear documentation.
 *
 * Note: When executing the program, make sure ThinkGear.bundle is in the
same current directory.
 */

#include <CoreFoundation/CoreFoundation.h>
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <unistd.h>

/**
 * Baud rate for use with TG_Connect() and TG_SetBaudrate().
 */
#define TG_BAUD_1200         1200
#define TG_BAUD_2400         2400
#define TG_BAUD_4800         4800
#define TG_BAUD_9600         9600
#define TG_BAUD_57600       57600
#define TG_BAUD_115200     115200

/**
 * Data format for use with TG_Connect() and TG_SetDataFormat().
 */
#define TG_STREAM_PACKETS      0
#define TG_STREAM_5VRAW        1
#define TG_STREAM_FILE_PACKETS 2

/**
 * Data type that can be requested from TG_GetValue().
 */
#define TG_DATA_BATTERY      0
#define TG_DATA_POOR_SIGNAL  1
#define TG_DATA_ATTENTION    2
#define TG_DATA_MEDITATION   3
#define TG_DATA_RAW          4
```

```
#define TG_DATA_DELTA        5
#define TG_DATA_THETA        6
#define TG_DATA_ALPHA1       7
#define TG_DATA_ALPHA2       8
#define TG_DATA_BETA1        9
#define TG_DATA_BETA2       10
#define TG_DATA_GAMMA1      11
#define TG_DATA_GAMMA2      12

CFURLRef bundleURL;          // path reference to bundle
CFBundleRef thinkGearBundle; // bundle reference

int connectionID = -1;       // ThinkGear connection handle

/*
 * ThinkGear function pointers
 */

int (*TG_GetDriverVersion)() = NULL;
int (*TG_GetNewConnectionId)() = NULL;
int (*TG_Connect)(int, const char *, int, int) = NULL;
int (*TG_ReadPackets)(int, int) = NULL;
float (*TG_GetValue)(int, int) = NULL;
int (*TG_Disconnect)(int) = NULL;
void (*TG_FreeConnection)(int) = NULL;

/**
 * This function handles signal interrupts.
 *
 * Basically perform cleanup on the objects and then exit the program.
 */
void siginthandler(int sig){
   fprintf(stderr, "\nDisconnecting...\n");

   // close the connection
   if(connectionID != -1){
      TG_Disconnect(connectionID);
      TG_FreeConnection(connectionID);
   }

   // release the bundle references
   if(bundleURL)
      CFRelease(bundleURL);

   if(thinkGearBundle)
      CFRelease(thinkGearBundle);

   exit(1);
}
```

```
/**
 * The main driver for this program.
 *
 * Handle command-line arguments, initialize the ThinkGear connection,
 * and handle output.
 */
int main (int argc, const char * argv[]) {
   // register the signal interrupt handler
   signal(SIGINT, siginthandler);

   // cmd line argument checking
   if(argc < 2){
      fprintf(stderr, "Usage: %s portname\n", argv[0]);
      exit(1);
   }

   const char * portname = argv[1];       // port name
   int retVal = -1;                       // return values from TG functions

   int numPackets = 0;                    // number of packets returned from
ReadPackets
   float signalQuality = 0.0;             // poor signal status
   float attention = 0.0;                 // eSense attention
   float meditation = 0.0;                // eSense meditation

   // create the path reference to the bundle
   bundleURL = CFURLCreateWithFileSystemPath(kCFAllocatorDefault,
                                             CFSTR("ThinkGear.bundle"),
                                             kCFURLPOSIXPathStyle,
                                             true);

   // create the bundle reference
   thinkGearBundle = CFBundleCreate(kCFAllocatorDefault, bundleURL);

   // make sure the bundle actually exists
   if(!thinkGearBundle){
      fprintf(stderr, "Error: Could not find ThinkGear.bundle. Does it exist
in the current directory?\n");
      exit(1);
   }

   // now start setting the function pointers
   TG_GetDriverVersion =   (void *)CFBundleGetFunctionPointerForName(
thinkGearBundle, CFSTR("TG_GetDriverVersion"));
   TG_GetNewConnectionId = (void *)CFBundleGetFunctionPointerForName(
thinkGearBundle, CFSTR("TG_GetNewConnectionId"));
   TG_Connect =            (void *)CFBundleGetFunctionPointerForName(
thinkGearBundle, CFSTR("TG_Connect"));
   TG_ReadPackets =        (void *)CFBundleGetFunctionPointerForName(
thinkGearBundle, CFSTR("TG_ReadPackets"));
   TG_GetValue =           (void *)CFBundleGetFunctionPointerForName(
thinkGearBundle, CFSTR("TG_GetValue"));
   TG_Disconnect =         (void *)CFBundleGetFunctionPointerForName(
thinkGearBundle, CFSTR("TG_Disconnect"));
   TG_FreeConnection =     (void *)CFBundleGetFunctionPointerForName(
thinkGearBundle, CFSTR("TG_FreeConnection"));

   // check for any invalid function pointers
   if(!TG_GetDriverVersion || !TG_GetNewConnectionId || !TG_Connect || !
TG_ReadPackets ||
      !TG_GetValue || !TG_Disconnect || !TG_FreeConnection){
      fprintf(stderr, "Error: Expected functions in ThinkGear.bundle were
not found. Are you using the right version?\n");
      exit(1);
   }

   // get the connection ID
   connectionID = TG_GetNewConnectionId();

   fprintf(stderr, "Connecting to %s ... ", portname);

   // attempt to connect
   retVal = TG_Connect(connectionID, portname, TG_BAUD_9600,
TG_STREAM_PACKETS);

   // check whether the connection attempt was successful
   if(!retVal){
      fprintf(stderr, "connected.\n");

      // loop until we get the interrupt signal from the console. control
      // then gets passed onto the signal handler function
      while(1){
         // sleep for half a second
         usleep(500000);

         // read the packets from the output stream
         numPackets = TG_ReadPackets(connectionID, -1);

         // check whether we've received any new packets
         if(numPackets > 0){
            // if so, parse them
            signalQuality = TG_GetValue(connectionID, TG_DATA_POOR_SIGNAL);
            attention = TG_GetValue(connectionID, TG_DATA_ATTENTION);
            meditation = TG_GetValue(connectionID, TG_DATA_MEDITATION);

            // then output everything
            fprintf(stdout, "\rPoorSig: %3.0f, Att: %3.0f, Med: %3.0f",
signalQuality, attention, meditation);
            fflush(stdout);
         }
      }
```

```
    }
    else {
        fprintf(stderr, "unable to connect. (%d)\n", retVal);
        exit(1);
    }

    return 0;
}
```