Stream SDK for PC: Development Guide

July 6, 2016

NeuroSky®
Brain-Computer Interface Technologies

The NeuroSky® product families consist of hardware and software components for simple integration of this biosensor technology into consumer and industrial end-applications. All products are designed and manufactured to meet consumer thresholds 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

This guide will teach you how to use **Stream SDK for PC** to write applications that can connect your PC and get bio-signal data from NeuroSky's bio-sensors.

Stream SDK contents

- Stream SDK for PC Development Guide (this document)
- libs
  - win32
    - thinkgear.dll
    - thinkgear.lib
    - thinkgear.h
    - csharp
      - thinkgear.dll
      - NativeThinkgear.cs
  - x64
    - thinkgear64.dll
    - thinkgear64.lib
    - thinkgear.h
    - csharp
      - thinkgear64.dll
      - NativeThinkgear64.cs
- Sample Project
  - win32
    - thinkgear_testapp
    - thinkgear_testapp_csharp
  - x64
    - thinkgear_testapp
    - thinkgear_testapp_csharp_64

- ChangLog.txt

Supported NeuroSky Hardware

- MindWave Mobile
- MindWave Mobile 1.5

Run the Sample project

The sample project is created with Microsoft Visual Studio 2012.

1. Make sure your ThinkGear headset is connected to the computer and find out which COM port is taken.

2. Open the .sln file with Visual Studio

3. Open the thinkgear_testapp.c file, find the variable **comPortName**, change the value to the COM port name that is take by ThinkGear headset.

4. Build the project (The "thinkgear_testapp.exe" file should appear in Windows Explorer in the "SOLUTION\Debug\" folder)

5. Use Windows Explorer to copy the "thinkgear.dll" file we provided into the same folder as the "thinkgear_testapp.exe" ("SOLUTION\Debug\"), or into any folder on your system PATH.

6. Run the "thinkgear_testapp.exe" file and watch the EEG brainwave values.

Note:

Create a 32 bit project is different from a 64 bit one, please visit https://msdn.microsoft.com/en-us/library/h2k70f3s.aspx to find out how to config a 64 bit program.

Develop with Stream SDK

1. Get connection id

```
int   connectionId = 0;
/* Get a connection ID handle to ThinkGear */
connectionId = TG_GetNewConnectionId();
if( connectionId < 0 ) {
    fprintf( stderr, "ERROR: TG_GetNewConnectionId() returned %d.\n",connectionId );
    wait();
    exit( EXIT_FAILURE );
}
```

2. connect

```
int   errCode      = 0;
...
errCode = TG_Connect( connectionId,
                     comPortName,
                     TG_BAUD_57600,
                     TG_STREAM_PACKETS );
if( errCode < 0 ) {
    fprintf( stderr, "ERROR: TG_Connect() returned %d.\n", errCode );
    wait();
    exit( EXIT_FAILURE );
}
```

3. Read packet

```
/* Read a single Packet from the connection */
int   packetsRead  = 0;
int   count = 1;
...
packetsRead = TG_ReadPackets( connectionId, count );
```

4. Get value status and get value

```
int data_type = TG_DATA_RAW;
...
if( TG_GetValueStatus(connectionId, data_type ) != 0 ) {

   /* Get the current time as a string */
   currTime = time( NULL );
    currTimeStr = ctime( &currTime );

   /* Get and print out the new raw value */
   fprintf( stdout, "%s: raw: %d\n", currTimeStr,
          (int)TG_GetValue(connectionId, data_type) );
          fflush( stdout );

 }
```

Develop with Stream SDK (c#)

1. Create c# project

2. Config the project as x86 (or x64)

3. Add NativeThinkgear.cs (or NativeThinkgear64.cs) to your project(right click your project - Add - Existing item… then select the file)

4. Put thinkgear.dll(or thinkgear64.dll) into these folders: Debug, Release, x86 or x64 which the exe file would be generated here

5. Add the namespace

```
using libStreamSDK;
```

6. For more information, please refer to the demo project thinkgear_testapp_csharp

API Reference

Constant

Maximum number of Connections

Macro	value
TG_MAX_CONNECTION_HANDLES	128

Baud rate for use with TG_Connect() and TG_SetBaudrate()

Macro	value
TG_BAUD_1200	1200
TG_BAUD_2400	2400
TG_BAUD_4800	4800
TG_BAUD_9600	9600
TG_BAUD_57600	57600
TG_BAUD_115200	115200

Data format for use with TG_Connect() and TG_SetDataFormat()

Macro	value
TG_STREAM_PACKETS	0
TG_STREAM_FILE_PACKETS	2

Data types that can be requested from TG_GetValue().
Only certain data types are output by certain ThinkGear chips and headsets.
Please refer to the Communications Protocol document for your chip/headset to determine which data types are available for your hardware.

Macro	value
TG_DATA_POOR_SIGNAL	1
TG_DATA_ATTENTION	2
TG_DATA_MEDITATION	3
TG_DATA_RAW	4
TG_DATA_DELTA	5
TG_DATA_THETA	6
TG_DATA_ALPHA1	7
TG_DATA_ALPHA2	8
TG_DATA_BETA1	9
TG_DATA_BETA2	10
TG_DATA_GAMMA1	11
TG_DATA_GAMMA2	12
MWM15_DATA_FILTER_TYPE	49

Filter type for MWM15_setFilterType()

Macro	value
MWM15_FILTER_TYPE_50HZ	4
MWM15_FILTER_TYPE_60HZ	5

TG_GetVersion

Returns a number indicating the version of the Stream SDK for PC library accessed by this API. Useful for debugging version-related issues.

```
THINKGEAR_API int TG_GetVersion();
```

Return

The Stream SDK's version number.

TG_GetNewConnectionId

Returns an ID handle (an int) to a newly-allocated ThinkGear Connection object. The Connection is used to perform all other operations of this API, so the ID handle is passed as the first argument to all functions of this API.

When the ThinkGear Connection is no longer needed, be sure to call TG_FreeConnection() on the ID handle to free its resources. No more than TG_MAX_CONNECTION_HANDLES Connection handles may exist simultaneously without being freed.

```
THINKGEAR_API int TG_GetNewConnectionId();
```

Return

-1 if too many Connections have been created without being freed by TG_FreeConnection().
-2 if there is not enough free memory to allocate to a new ThinkGear Connection.
The ID handle of a newly-allocated ThinkGear Connection.

TG_SetStreamLog

As a ThinkGear Connection reads bytes from its serial stream, it may automatically log those bytes into a log file. This is useful primarily for debugging purposes. Calling this function with a valid @c filename will turn this feature on. Calling this function with an invalid @c filename, or with @c filename set to NULL, will turn this feature off. This function may be called at any time for either purpose on a ThinkGear Connection.

```
THINKGEAR_API int TG_SetStreamLog( int connectionId, const char *filename );
```

Parameter

connectionId The ID of the ThinkGear Connection to enable stream logging for, as obtained from TG_GetNewConnectionId().
filename The name of the file to use for stream logging. Any existing contents of the file will be erased. Set to NULL or an empty string to disable stream logging by the ThinkGear Connection.

Return

-1 if @c connectionId does not refer to a valid ThinkGear Connection ID handle.
-2 if @c filename could not be opened for writing. You may check errno for the reason.
0 on success.

TG_SetDataLog

As a ThinkGear Connection reads and parses Packets of data from its serial stream, it may log the parsed data into a log file. This is useful primarily for debugging purposes. Calling this function with a valid @c filename will turn this feature on. Calling this function with an invalid @c filename, or with @c filename set to NULL, will turn this feature off. This function may be called at any time for either purpose on a ThinkGear Connection.

```
THINKGEAR_API int TG_SetDataLog( int connectionId, const char *filename );
```

Parameter

connectionId The ID of the ThinkGear Connection to enable data logging for, as obtained from TG_GetNewConnectionId().
filename The name of the file to use for data logging. Any existing contents of the file will be erased. Set to NULL or an empty string to disable stream logging by the ThinkGear Connection.

Return

-1 if @c connectionId does not refer to a valid ThinkGear Connection ID handle.
-2 if @c filename could not be opened for writing. You may check errno for the reason.
0 on success.

TG_WriteStreamLog

Writes a message given by @c msg into the Connection's Stream Log. Optionally the message can be written onto a new line preceded by a timestamp. The Connection's Stream Log must be already opened by TG_SetStreamLog(), otherwise this function returns an error.

```
THINKGEAR_API int TG_WriteStreamLog( int connectionId, int insertTimestamp,
                                     const char *msg );
```

Parameter

connectionId The ID of the ThinkGear Connection to write into the Stream Log for, as obtained from TG_GetNewConnectionId().
insertTimestamp If set to any non-zero number, a newline and timestamp are automatically prepended to the @c msg in the Stream Log file. Pass a zero for this parameter to disable the insertion of timestamp before @c msg.
msg The message to write into the Stream Log File. For Stream Log parsers to ignore the message as a human-readable comment instead of hex bytes, prepend a '#' sign to indicate it is a comment.

Return

-1 if @c connectionId does not refer to a valid ThinkGear Connection ID handle.
-2 if Stream Log for the @c connectionId has not been opened for writing via TG_SetStreamLog().
0 on success.

TG_WriteDataLog

Writes a message given by @c msg into the ThinkGear Connection's Data Log. Optionally the message can be written onto a new line preceded by a timestamp. The Connection's Data Log must be already opened by TG_SetDataLog(), otherwise this function returns an error.

```
THINKGEAR_API int TG_WriteDataLog( int connectionId, int insertTimestamp,
                                   const char *msg );
```

Parameter

connectionId The ID of the ThinkGear Connection to write into the Data Log for, as obtained from TG_GetNewConnectionId().
insertTimestamp If set to any non-zero number, a newline and timestamp are automatically prepended to the @c msg in the Stream Log file. Pass a zero for this parameter to disable the insertion of timestamp before @c msg.
msg The message to write into the Data Log File. For Data Log parsers to ignore the message as a human-readable comment instead of hex bytes, prepend a '#' sign to indicate it is a comment.

Return

-1 if @c connectionId does not refer to a valid ThinkGear Connection ID handle.
-2 if Data Log for the @c connectionId has not been opened for writing via TG_SetDataLog().
0 on success.

TG_Connect

Connects a ThinkGear Connection, given by @c connectionId, to a serial communication (COM) port in order to communicate with a ThinkGear module. It is important to check the return value of this function before attempting to use the Connection further for other functions in this API.

```
THINKGEAR_API int TG_Connect( int connectionId, const char *serialPortName,
                   int serialBaudrate, int serialDataFormat );
```

Parameter

connectionId The ID of the ThinkGear Connection to connect, as obtained from TG_GetNewConnectionId().
serialPortName The name of the serial communication (COM) stream port. COM ports on PC Windows systems are named like '\\.\COM4' (remember that backslashes in strings in most programming languages need to be escaped), while COM ports on Windows Mobile systems are named like 'COM4:' (note the colon at the end). Linux COM ports may be named like '/dev/ttys0'. Refer to the documentation for your particular platform to determine the available COM port names on your system.
serialBaudrate The baudrate to use to attempt to communicate on the serial communication port. Select from one of the TG_BAUD_* constants defined above, such as TG_BAUD_9600 or TG_BAUD_57600.
serialDataFormat The type of ThinkGear data stream. Select from one of the TG_STREAM_* constants defined above. Most applications should use TG_STREAM_PACKETS (the data format for Embedded ThinkGear).

Return

-1 if @c connectionId does not refer to a valid ThinkGear Connection ID handle.
-2 if @c serialPortName could not be opened as a serial communication port for any reason. Check that the name is a valid COM port on your system.
-3 if @c serialBaudrate is not a valid TG_BAUD_* value.
-4 if @c serialDataFormat is not a valid TG_STREAM_* type.
0 on success.

TG_ReadPackets

Attempts to use the ThinkGear Connection, given by @c connectionId, to read @c numPackets of data from the serial stream. The Connection will (internally) "remember" the most recent value it has seen of each possible ThinkGear data type, so that any subsequent call to @c TG_GetValue() will return the most recently seen values.
Set @c numPackets to -1 to attempt to read all Packets of data that may be currently available on the serial stream.
Note that different models of ThinkGear hardware and headsets may output different types of data values at different rates. Refer to the "Communications Protocol" document for your particular headset to determine the rate at which you need to call this function.

```
THINKGEAR_API int TG_ReadPackets( int connectionId, int numPackets );
```

Parameter

connectionId The ID of the ThinkGear Connection which should read packets from its serial communication stream, as obtained from TG_GetNewConnectionId().
numPackets The number of data Packets to attempt to read from the ThinkGear Connection. Only the most recently read value of each data type will be "remembered" by the ThinkGear Connection. Setting this parameter to -1 will attempt to read all currently available Packets that are on the data stream.

Return

-1 if @c connectionId does not refer to a valid ThinkGear Connection ID handle.
-2 if there were not even any bytes available to be read from the Connection's serial communication stream.
-3 if an I/O error occurs attempting to read from the Connection's serial communication stream.
The number of Packets that were successfully read and parsed from the Connection.

TG_GetValueStatus

Returns Non-zero if the @c dataType was updated by the most recent call to TG_ReadPackets(). Returns 0 otherwise.

```
THINKGEAR_API int TG_GetValueStatus( int connectionId, int dataType );
```

Parameter

connectionId The ID of the ThinkGear Connection to get a data value from, as obtained from TG_GetNewConnectionId().
dataType The type of data value desired. Select from one of the TG_DATA_* constants defined above.

Return

Non-zero if the @c dataType was updated by the most recent call to TG_GetValue(). Returns 0 otherwise.

NOTE: This function will terminate the program with a message printed to stderr if @c connectionId is not a valid ThinkGear Connection, or if @c dataType is not a valid TG_DATA_* constant.

TG_GetValue

Returns the most recently read value of the given @c dataType, which is one of the TG_DATA_* constants defined above. Use @c TG_ReadPackets() to read more Packets in order to obtain updated values. Afterwards, use @c TG_GetValueStatus() to check if a call to @c TG_ReadPackets() actually updated a particular @c dataType.

```
THINKGEAR_API int TG_GetValue( int connectionId, int dataType );
```

Parameter

connectionId The ID of the ThinkGear Connection to get a data value from, as obtained from TG_GetNewConnectionId().
dataType The type of data value desired. Select from one of the TG_DATA_* constants defined above. Although many types of TG_DATA_* constants are available, each model of ThinkGear hardware and headset will only output a certain subset of these data types. Refer to the Communication Protocol document for your particular ThinkGear hardware or headset to determine which data types are actually output by that hardware or headset. Data types that are not output by the headset will always return their default value of 0.0 when this function is called.

Return

The most recent value of the requested @c dataType.

NOTE: This function will terminate the program with a message printed to stderr if @c connectionId is not a valid ThinkGear Connection, or if @c dataType is not a valid TG_DATA_* constant.

TG_SendByte

Sends a byte through the ThinkGear Connection (presumably to a ThinkGear module). This function is intended for advanced ThinkGear Command Byte operations.

WARNING:

Always make sure at least one valid Packet has been read (i.e. through the @c TG_ReadPackets() function) at some point **BEFORE** calling this function. This is to ENSURE the Connection is communicating at the right baud rate. Sending Command Byte at the wrong baud rate may put a ThinkGear module into an indeterminate and inoperable state until it is reset by power cycling (turning it off and then on again).

```
THINKGEAR_API int TG_SendByte( int connectionId, int b );
```

Parameter

connectionId The ID of the ThinkGear Connection to send a byte through, as obtained from TG_GetNewConnectionId().
b The byte to send through. Note that only the lowest 8-bits of the value will actually be sent through.

Return

-1 if @c connectionId does not refer to a valid ThinkGear Connection ID handle.
-2 if @c connectionId is connected to an input file stream instead of an actual ThinkGear COM stream (i.e. nowhere to send the byte to).
-3 if an I/O error occurs attempting to send the byte (i.e. broken stream connection).
0 on success.

NOTE: After sending a Command Byte that changes a ThinkGear baud rate, you will need to call @c TG_SetBaudrate() to change the baud rate of your @c connectionId as well. After such a baud rate change, it is important to check for a valid Packet to be received by @c TG_ReadPacket() before attempting to send any other Command Bytes, for the same reasons as describe in the WARNING above.

TG_SetBaudrate

Attempts to change the baud rate of the ThinkGear Connection, given by @c connectionId, to @c serialBaudrate. This function does not typically need to be called, except after calling @c TG_SendByte() to send a Command Byte that changes the ThinkGear module's baud rate. See TG_SendByte() for details and NOTE.

```
THINKGEAR_API int TG_SetBaudrate( int connectionId, int serialBaudrate );
```

Parameter

connectionId The ID of the ThinkGear Connection to send a byte through, as obtained from TG_GetNewConnectionId().
serialBaudrate The baudrate to use to attempt to communicate on the serial communication port. Select from one of the TG_BAUD_* constants defined above, such as TG_BAUD_9600 or TG_BAUD_57600. TG_BAUD_57600 is the typical default baud rate for most ThinkGear models.

Return

-1 if @c connectionId does not refer to a valid ThinkGear Connection ID handle.
-2 if @c serialBaudrate is not a valid TG_BAUD_* value.
-3 if an error occurs attempting to set the baud rate.
-4 if @c connectionId is connected to an input file stream instead of an actual ThinkGear COM stream.
0 on success.

TG_EnableAutoRead

Enables or disables background auto-reading of the connection. This has the following implications:
- Setting @c enabled to anything other than 0 will enable background auto-reading on the specified connection. Setting @c enabled to 0 will disable it.
- Enabling causes a background thread to be spawned for the connection (only if one was not already previously spawned), which continuously calls TG_ReadPacket(connectionId, -1) at 1ms intervals.
- Disabling will kill the background thread for the connection.
- While background auto-reading is enabled, the calling program can use TG_GetValue() at any time to get the most-recently-received value of any data type. The calling program will have no way of knowing when a value has been updated. For most data types other than raw wave value, this is not much of a problem if the program simply polls TG_GetValue() once a second or so.
- The current implementation of this function will not include proper data synchronization. This means it is possible for a value to be read (by TG_GetValue()) at the same time it is being written to by the background thread, resulting in a corrupted value being read. However, this is extremely unlikely for most data types other than raw wave value.
- While background auto-reading is enabled, the TG_GetValueStatus() function is pretty much useless. Also, the TG_ReadPackets() function should probably not be called.

```
THINKGEAR_API int TG_EnableAutoRead( int connectionId, int enable );
```

Parameter

connectionId The connection to enable/disable background auto-reading on.
enable Zero (0) to disable background auto-reading, any other value to enable.

Return

-1 if @c connectionId does not refer to a valid ThinkGear Connection ID handle.
-2 if unable to enable background auto-reading.
-3 if an error occurs while attempting to disable background auto-reading.
0 on success.

TG_Disconnect

Disconnects the ThinkGear Connection, given by @c connectionId, from its serial communication (COM) port. Note that after this call, the Connection will not be valid to use with any of the API functions that require a valid ThinkGear Connection, except TG_SetStreamLog(), TG_SetDataLog(), TG_Connect(), and TG_FreeConnection().

Note that TG_FreeConnection() will automatically disconnect a Connection as well, so it is not necessary to call this function unless you wish to reuse the @c connectionId to call TG_Connect() again.

```
THINKGEAR_API void TG_Disconnect( int connectionId );
```

Parameter

connectionId The ID of the ThinkGear Connection to disconnect, as obtained from TG_GetNewConnectionId().

TG_FreeConnection

Frees all memory associated with the given ThinkGear Connection.

Note that this function will automatically call TG_Disconnect() to disconnect the Connection first, if appropriate, so that it is not necessary to explicitly call TG_Disconnect() at all, unless you wish to reuse the @c connectionId without freeing it first for whatever reason.

```
THINKGEAR_API void TG_FreeConnection( int connectionId );
```

Parameter

connectionId The ID of the ThinkGear Connection to free, as obtained from TG_GetNewConnectionId().

MWM15_getFilterType

Get the working filter type information. Only support MindWave Mobile 1.5. This is an asynchronous call, it will return immediately, and the filter type will return by data type MWM15_DATA_FILTER_TYPE. confirm the state is connected before calling this funciton.

```
THINKGEAR_API int MWM15_getFilterType( int connectionId);
```

Parameters

connectionId connectionId The ID of the ThinkGear Connection , obtained from TG_GetNewConnectionId().

Return

0 on success, <0 on fail.

MWM15_setFilterType

Set the filter type with MWM15_FILTER_TYPE_50HZ or MWM15_FILTER_TYPE_60HZ. The value depends on the alternating current frequency(or AC Frequency) of your country or territory, for examle, USA 60HZ, China 50HZ. If you are not sure about it, please google "alternating current frequency YOUR LOCATION". This function only support MindWave Mobile1.5. This is an asynchronous call, it will return immediately. Please confirm the state is connected before calling this funciton.

If call this function success, MindWave Mobile 1.5 will stop send out data for 0.8 second. If you want to check the result, please call MWM15_getFilterType 1 second later.

Please note that if you have called this function, and you want call it again, you have to wait more than 1 secend.

```
THINKGEAR_API int MWM15_setFilterType( int connectionId, int filterType);
```

Parameters

connectionId connectionId The ID of the ThinkGear Connection , obtained from TG_GetNewConnectionId().
filterType MWM15_FILTER_TYPE_50HZ or MWM15_FILTER_TYPE_60HZ

Return

0 on success, <0 on fail.

Warnings and Disclaimer of Liability

THE ALGORITHMS MUST NOT BE USED FOR ANY ILLEGAL USE, OR AS COMPONENTS IN LIFE SUPPORT OR SAFETY DEVICES OR SYSTEMS, OR MILITARY OR NUCLEAR APPLICATIONS, OR FOR ANY OTHER APPLICATION IN WHICH THE FAILURE OF THE ALGORITHMS COULD CREATE A SITUATION WHERE PERSONAL INJURY OR DEATH MAY OCCUR. YOUR USE OF THE SOFTWARE DEVELOPMENT KIT, THE ALGORITHMS AND ANY OTHER NEUROSKY PRODUCTS OR SERVICES IS "AS-IS," AND NEUROSKY DOES NOT MAKE, AND HEREBY DISCLAIMS, ANY AND ALL OTHER EXPRESS AND IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE, AND ANY WARRANTIES ARISING FROM A COURSE OF DEALING, USAGE, OR TRADE PRACTICE.