Developer Tools 2.5 Development Guide

What's New

Version 2.5 of the Mind Developer Tools (MDT) adds new interfaces to access data from NeuroSky devices, specifically: ThinkGear SDK for Mac OSX and ThinkGear SDK for .NET. There are also minor updates to clarify documentation and improve consistency for future releases of new algorithms. Previews of these new algorithm in advance of a Developer Tools release can be found in separate Beta previews that will be available through http://developer.neurosky.com.

Version History

Version 2.5:
Addition of ThinkGear SDK for Mac OSX and .NET
Inclusion of Application Standards and related images. Documentation update

Version 2.1:
Eye-blink Detection in software API

Version 2:
Addition of ThinkGear Connector

Introduction

This Development Guide will direct you to the files and code samples you need to develop BCI-enabled applications using NeuroSky ThinkGear-enabled headsets (such as the MindWave and MindWave Mobile) for computers and microcontrollers (ie. Arduino). The MDT for PC/Mac offers 4 levels of interfaces for communicating with these headsets. From highest to lowest levels, they are:

1. ThinkGear SDK for Mac (Mac OS X) and ThinkGear SDK for .Net (Windows)
2. ThinkGear Connector (TGC) (Windows and Mac OS X executable)
3. ThinkGear Communications Driver (TGCD) (Windows, Windows Mobile, Mac OS X, and J2ME (Symbian) libraries)
4. Communication Stream (source code for any C platform)

The highest level interface provides an SDK to develop applications for Windows and Mac OS X. The included sample code provides a detailed way to receive and use bio-signal data. These two higher-level interfaces supply executables and binary libraries for some of the most common computing platforms, like Windows and Mac OS X. The lowest-level Serial Stream SDK interface provides source code and low level communication stream specs that allow headset development on virtually any platform that can receive a serial data stream.

Please review the appropriate chapter for the interface and platform you are interested in developing for. This Development Guide provides a broad summary of the different interfaces.

About NeuroSky's ThinkGear

ThinkGear is the technology inside every NeuroSky brainwave product or partner product that enables a device to interface with the user's brainwaves. It includes the sensor that touches the forehead, the contact and reference points located on the ear pad, and the onboard chip that processes all of the data. Both the raw brainwaves and the eSense Meters (Attention and Meditation) are calculated on the ThinkGear chip.

ThinkGear technology in the MindWave and MindWave Mobile power the development tools, drivers, and APIs provided in the MDT. Please be certain to review the documentation with your corresponding hardware.

Headsets

MindWave (MW001)	**MindWave Mobile (MW003)**

The MindWave and MindWave Mobile are two of the headsets compatible with the Developer Tools. The best and most up to date source of data for both the MindWave and MindWave Mobile can be found on the following webpages. These pages include full installer files to setup drivers/applications specific to your headset.

MindWave: http://MindWave.NeuroSky.com

MindWave Mobile: http://MindWaveMobile.NeuroSky.com

eSense(tm) Meters

One of the first questions on development is: What does the data mean?

For the different types of eSenses (i.e. Attention, Meditation), the meter value is reported on a relative scale of 1 to 100. On this scale, a value between 40 to 60 at any given moment in time is considered “neutral”, similar in notion to the “baselines” that are established in conventional EEG measurement techniques (though the method for determining a ThinkGear baseline is proprietary and may differ from conventional EEG). A value from 60 to 80 is considered “slightly elevated”, and may be interpreted as levels possibly higher than normal (levels of Attention or Meditation that may be higher than normal for a given person). Values from 80 to 100 are considered “elevated”, meaning they are strongly indicative of heightened levels of that eSense.

Similarly, on the other end of the scale, a value between 20 to 40 indicates “reduced” levels of the eSense, while a value between 1 to 20 indicates “strongly lowered” levels of the eSense. These levels may indicate states of distraction, agitation, or abnormality, according to the opposite of each eSense.

An eSense meter value of 0 is a special value indicating the ThinkGear is unable to calculate an eSense level with a reasonable amount of reliability. This may be (and usually is) due to excessive noise as described in the POOR_SIGNAL Quality section above.

The reason for the somewhat wide ranges for each interpretation is that some parts of the eSense algorithm are dynamically learning, and at times employ some “slow-adaptive” algorithms to adjust to natural fluctuations and trends of each user, accounting for and compensating for the fact that EEG in the human brain is subject to normal ranges of variance and fluctuation. This is part of the reason why ThinkGear sensors are able to operate on a wide range of individuals under an extremely wide range of personal and environmental conditions while still giving good accuracy and reliability. Developers are encouraged to further interpret and adapt these guideline ranges to be fine-tuned for their application (as one example, an application could disregard values below 60 and only react to values between 60-100, interpreting them as the onset of heightened attention levels).

Attention eSense

The Attention value which indicates the intensity of a user's level of mental “focus” or “attention”, such as that which occurs during intense concentration and directed (but stable) mental activity. Its value ranges from 0 to 100. Distractions, wandering thoughts, lack of focus, or anxiety may lower the Attention meter levels. See eSense™ Meters above for details about interpreting eSense levels in general.

Meditation eSense

The Meditation value indicates the level of a user's mental “calmness” or “relaxation”. Its value ranges from 0 to 100. Note that Meditation is a measure of a person's mental levels, not physical levels, so simply relaxing all the muscles of the body may not immediately result in a heightened Meditation level. However, for most people in most normal circumstances, relaxing the body often helps the mind to also relax. Meditation is related to reduced activity by the active mental processes in the brain, and it has long been an observed effect that closing one's eyes turns off the mental activities which process images from the eyes, so closing the eyes is often an effective method for increasing the Meditation meter level. Distractions, wandering thoughts, anxiety, agitation, and sensory stimuli may lower the Meditation meter levels. See “eSense Meters” above for details about interpreting eSense levels in general.

Additional Data

Headsets output additional data, including Raw Data, Brainwave Bands, and Signal Quality measurements. In addition, there are additional algorithms in some of the SDK's, which enable additional calculations including Eyeblinks. Please review the in-depth documentation SDK for a complete review of the additional outputs.

ThinkGear SDK for Mac OSX

The ThinkGear SDK for Mac OSX provides documentation and sample code to show how to write Mac applications that can utilize data from NeuroSky's ThinkGear family of bio-sensors.

Folder Contents

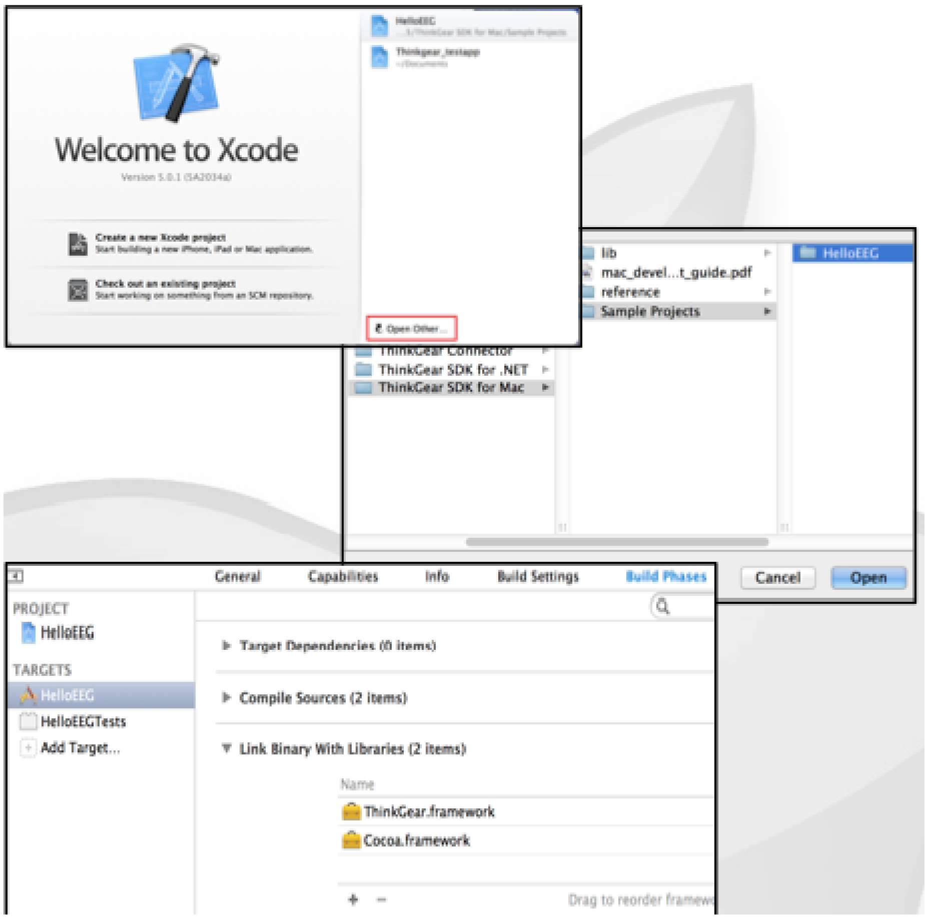
* ThinkGear SDK for Mac: Development Guide (mac_development_guide.pdf)

* ThinkGear.framework library

* ThinkGearTouch example project for Mac

You will find the “ThinkGear.framework” in the `lib/` folder, and the “HelloEEG” in the `Sample Project/` folder within the ThinkGear SDK for Mac OSX folder.

Quick Start for Mac Developers



1. From Xcode, choose “Open Other…”.
2. Browse through *ThinkGear SDK for Mac\Sample Projects\HelloEEG* and Open it.
3. Import ThinkGear.framework from *ThinkGear SDK for Mac/lib* into the project.

For further details, please refer to *mac_development_guide.pdf* in /ThinkGear SDK for Mac/

API Reference Table

void Connect(string portName)	Attempts to open a connection with the port name specified by portName.
void ConnectScan()	Attempts to open a connection to the first Device seen by the Connector.
void ConnectScan(string portName)	Same as ConnectScan but scans the port specified by portName first.
void Discount()	Closes all open connections.
void Disconnect(Connection connection)	Close a specific connection specified by Connection.
void Disconnect(Device device)	Close a specific device specified by Device.
void Send(string portName, byte[byteToSend)	Send an array of bytes to a specific port.
void enableMentalEffort() DEPRECATED	Starts recording data for 60 sec. Once the recording is complete, the Mental Effort will be calculated.

void enableFamiliarity() DEPRECATED	Starts recording data for 60 sec. Once the recording is complete, the Familiarity will be calculated.
Event	
DeviceFound	Occurs when a ThinkGear device is found.
DeviceNotFound	Occurs when a ThinkGear device could not be found.
DeviceValidating	Occurs right before the connector attempts a special port.
DeviceConnected	Occurs when a ThinkGear device is connected.
DeviceConnectFail	Occurs when the Connector fails to connect to that port specified.
DeviceDisconnected	Occurs when the Connector disconnects from a ThinkGear device.
DataReceived	Occurs when data is available from a ThinkGear Device.
TGParser Class	
Dictionary<string, double>[] Read(DataRow[] dataRow)	Parses the raw headset data in dataRow and returns a dictionary of usable data.

ThinkGear SDK for .NET

The ThinkGear SDK for .NET provides documentation and sample code to show how to write .NET applications that can utilize data from NeuroSky's ThinkGear family of bio-sensors.

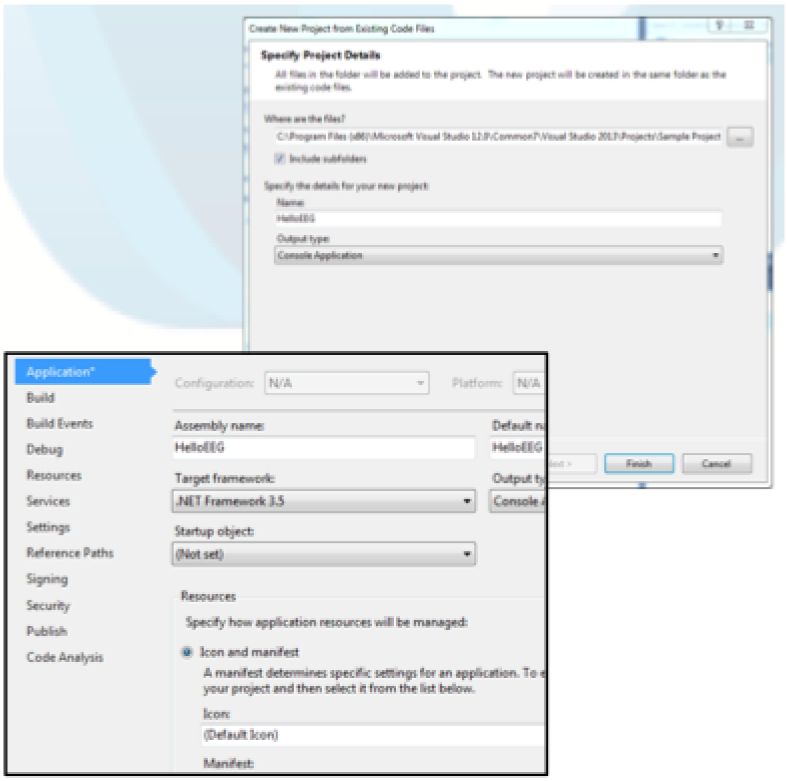
Folder Contents

* NeuroSky SDK for .NET: Development Guide and API Reference

- libs/:
  - **ThinkGear.dll** library
  - **JayrockJson.dll** supporting library
  - **NLog.dll** supporting library
  - **NLog.xml** and **NLog.config** configuration files
  - **NLog.LICENSE.txt** and **Jayrock.LICENCE.txt** licence files
- **TG-HelloEEG.exe** - a reference build of the HelloEEG sample project
- **HelloEEG Sample Project** source code

You'll find the .dll, configuration files and 3rd party license documents in the `libs/` folder. Copy this entire folder into your project.

Quick Start for .Net Developers



Specify Project Details

All files in the folder will be added to the project. The new project will be created in the same folder as the existing code files.

Where are the files?
C:\Program Files (x86)\Microsoft Visual Studio 12.0\Common7\Visual Studio 2013\Projects\Sample Project

Include subfolders

Specify the details for your new project:
Name:
HelloEEG
Output type:
Console Application

Configuration: N/A Platform: N/A

Assembly name:
HelloEEG
Default n
HelloEEG

Target framework:
.NET Framework 3.5
Output ty
Console

Startup object:
(Not set)

Resources
Specify how application resources will be managed:
Icon and manifest
A manifest determines specific settings for an application. To
your project and then select it from the list below.
Icon:
(Default Icon)
Manifest:

1. *From Visual Studio, choose File → New → Project From Existing Code…
2. Project type: **Visual C#**
3. Project path: *ThinkGear SDK for .NET\Sample Project\HelloEEG*
4. Project name: **HelloEEG**
5. Output type: **Console Application**
6. From toolbar Project → HelloEEG Properties and set Target framework to **.NET Framework 3.5**
7. Right click on **Reference** from Solution Explorer and choose **Add Reference**
8. Choose “Browse” Tab and add **ThinkGear.dll** from *HelloEEG/neurosky* into the project
9. Remove **Microsoft.CSharp** reference if there is a warning

*For VS Express users, unzip **HelloEEG_Ex.zip** from the project directory. Choose “Open project” and open HelloEEG.csproj.

For further details, please refer to *thinkgear.net_sdk_dev_guide_and_api_reference.pdf* in /ThinkGear SDK for .NET/

API Reference Table

void Connect(string portName)	Attempts to open a connection with the port name specified by portName.
void ConnectScan()	Attempts to open a connection to the first Device seen by the Connector.
void ConnectScan(string portName)	Same as ConnectScan but scans the port specified by portName first.
void Discount()	Closes all open connections.
void Disconnect(Connection connection)	Close a specific connection specified by Connection.
void Disconnect(Device device)	Close a specific device specified by Device.

void Send(string portName, byte[byteToSend)	Send an array of bytes to a specific port.
void enableMentalEffort() DEPRECATED	Starts recording data for 60 sec. Once the recording is complete, the Mental Effort will be calculated.
void enableFamiliarity() DEPRECATED	Starts recording data for 60 sec. Once the recording is complete, the Familiarity will be calculated.
Event	
DeviceFound	Occurs when a ThinkGear device is found.
DeviceNotFound	Occurs when a ThinkGear device could not be found.
DeviceValidating	Occurs right before the connector attempts a special port.
DeviceConnected	Occurs when a ThinkGear device is connected.
DeviceConnectFail	Occurs when the Connector fails to connect to that port specified.
DeviceDisconnected	Occurs when the Connector disconnects from a ThinkGear device.
DataReceived	Occurs when data is available from a ThinkGear Device.
TGParser Class	
Dictionary<string, double>[] Read(DataRow[] dataRow)	Parses the raw headset data in dataRow and returns a dictionary of usable data.

ThinkGear Connector (TGC)

The ThinkGear Connector (TGC) is an executable that provides a daemon-like service that manages communications with ThinkGear devices, such as the MindWave and MindWave Mobile, that are connected to the computer. The TGC runs continuously in the background and keeps an open socket on the local user's computer, allowing applications to connect to it and receive information from the connected ThinkGear devices. This means that any application in any language that can open and read from sockets (such as Flash's ActionScript3 and other scripting languages) can connect and receive data from NeuroSky headsets.

The TGC is provided as an executable for the following platforms:

- Windows
- Mac OSX

For more information on the TGC, including how to run it, and how to write applications to communicate with it, please browse to the MDT's `ThinkGear Connector/` directory, and read the corresponding documentation. These documents review the ThinkGear Connector application in addition to the Socket API used to pass data between the headset and your developed application.

ThinkGear Communications Driver (TGCD)

The ThinkGear Communications Driver (TGCD) is a device driver with a simple API that allows communication between an Application on a computer and a ThinkGear chip/module/headset. It is available as a .dll (for x86 or ARMV4I platforms), as a .bundle (for Mac OS X platforms), or as a .java library (for J2ME/Symbian platforms).

The ThinkGear Communication Driver is deprecated as of Version 2.5 of the MDT.

Windows PC Development

One method to begin Windows PC development is by using the provided ThinkGear Communications Driver (TGCD), compatible with both Win32 and Windows Mobile.

Files

Listed below are the development file(s) you will need to add to your development environment/project. Only add the file(s) that correspond to the language you intend to use:

Language	**Project File(s) (under `ThinkGear Communications Driver/win32/`)**
C/C++	`thinkgear.h` and `thinkgear.lib`
C#	`ThinkGear.cs`
Java (via JNI)	`ThinkGear.java`

API Documentation

Look for API documentation in the directories and files below:

Reference	**Language**	**File (under `ThinkGear Communications Driver/docs/`)**
TGCD API (doxygen)	C/C++	`html/index.html`
TGCD API (javadoc)	Java (via JNI)	`java/index.html`

The primary documentation for the TGCD API's for the C# and Java(JNI) languages in the `.cs` and `.java` wrapper files respectively, and are almost identical to the C/C++ API from which they are derived, with only possibly some cosmetic naming changes. It is recommended that you refer directly to the contents of the `.cs` or `.java` files themselves for API documentation for those languages. Otherwise, you may refer to the doxygen-generated C/C++ html documentation mentioned in the table above.

In Windows, COM port names should have a `\\.\` prepended to them. It is a required prefix for addressing serial ports above `COM9`, but is optional otherwise. Read this MSDN document for more details (particularly the section on “Win32 Device Namespaces”).

Examples

Here is an example C program that uses the most important parts of the TGCD API, along with instructions for setting up and compiling the program in Visual Studio:

Example/tutorial/guide	File (under `ThinkGear Communication Driver/win32/`)
Sample C/C++ App	`thinkgear_testapp.c`
Visual Studio 2005 Guide	`README.txt`

The function calls relating to the TGCD API in that example are almost identical to the function calls that would need to be made in C# or Java/JNI, so please use the C example as a guide towards building your first MindWave or MindWave Mobile application in those languages.

Execution

For all languages, once you have built an executable, you'll need to place the TGCD DLL (called `thinkgear.dll`) in the same directory as the executable (or in an appropriate place on your system PATH) in order to run the executable:

Driver	File (under `ThinkGear Communications Driver/win32/`)
TGCD DLL	`thinkgear.dll`

Once you are comfortable with the API and the example program, you'll find additional low level details, full descriptions of ThinkGear Data Types, Command Byte tables, and other advanced information in the mindset_communications_protocol document (found in the MDT's `Communication Stream/` directory).

Windows Mobile Development

Using the TGCD to interface with a MindWave or MindWave Mobile on devices running Windows Mobile (WinMo) is virtually identical to Windows PC Development, except that you'll need to use a different set of files.

Files

Here are the development file(s) you'll need to add to your development environment/project. You only need to add file(s) for the language you intend to use:

Language	Project File(s) (under `ThinkGear Communications Driver/winmobile/`)
C/C++	`thinkgear.h` and `thinkgear_ARMV4I.lib`
C#	`ThinkGear_ARMV4I.cs`

Language	Project File(s) (under `ThinkGear Communications Driver/winmobile/`)
Java (via JNI)	*Not supported*

API Documentation

The API for WinMo is the same as the one described in Windows PC Development. C#-specific API is documented within the `.cs` wrapper file itself, and is almost identical to the C/C++ API, with only possibly some cosmetic naming changes. Please refer directly to the `ThinkGear_ARMV4I.cs` file for C# API documentation.

Serial COM ports on WinMo devices have a colon appended to their names, e.g. `COM1:`, `COM2:`, etc..

Also, please refer to Windows PC Development for code samples and starter projects that can be adapted to WinMo, as they are similar for WinMo (since the APIs are the same). WinMo-specific examples and tutorials are not available at this time.

Execution

Once your program is compiled and deployed, you'll also need to use this DLL, compiled for mobile ARMV4I processors. Place it in the same directory as your deployed executable, or wherever else DLLs can be found by the WinMo system PATH:

Driver	File (under `ThinkGear Communications Driver/winmobile/`)
TGCD DLL	`thinkgear_ARMV4I.dll`

Mac OS X Development

The ThinkGear Communication Driver is deprecated as of Version 2.5 of the MDT. It is advised to use the ThinkGear SDK for Mac OSX.

Here are the development file(s) you'll need to add to your development environment/project. You only need to add file(s) for the language you intend to use:

Language	Project File(s) (under `ThinkGear Communications Driver/macosx/`)
C/C++	`ThinkGear.bundle`
C#	*Not supported*
Java (via JNI)	*Not supported*

API Documentation

The API for Mac OS X is the same as the one described in Windows PC Development. In addition, refer to the How To Use The ThinkGear API In Xcode (Mac OS X) document for additional information about using the API on Mac OS X (found in `ThinkGear Communications Driver/macosx/`).

An example Cocoa/Carbon application using the TGCD API is described in the document ThinkGear API MacOSX Example (also found in `ThinkGear Communications Driver/macosx/`).

J2ME (Symbian) Mobile Development

Symbian devices supporting J2ME can use the Java libraries in the MDT's `ThinkGear Communications Driver/j2me/` directory to develop applications:

- See `ThinkGear Communications Driver/j2me/javadoc/index.html` for documentation.

ThinkGear Serial Stream SDK

General Development (All Other Platforms, Including Microcontrollers)

For all other platforms not covered by the TGC nor TGCD, it is up to the application to open the serial I/O communication channel (COM port or direct serial UART). Please refer to the platform's documentation for “serial I/O” or “UART” on how to open and read from such a channel since serial I/O APIs tend to be platform-specific.

Once the communication channel is open and bytes can be read, we provide a packet parsing library for parsing and decoding the incoming data bytes. The library is in the form of ANSI C source code (for general compatibility), and can be easily ported to other languages as necessary:

(You'll find these files in the MDT's `Serial Stream SDK/` directory)

- `ThinkGearStreamParser.h`
- `ThinkGearStreamParser.c`

Refer to the mindset_communications_protocol (also found in the MDT's `Serial Stream SDK/` directory) for low-level specs, instructions, and sample code.

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.

IN NO EVENT SHALL NEUROSKY BE LIABLE FOR ANY SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES, INCLUDING BUT NOT LIMITED TO LOSS OF PROFITS OR INCOME, WHETHER OR NOT NEUROSKY HAD KNOWLEDGE, THAT SUCH DAMAGES MIGHT BE INCURRED.