面向 .NET的ThinkGear SDK :开发指南与API参考手册

2013/5/30

NeuroSky
Brain-Computer Interface Technologies

NeuroSky神念科技产品系列包括硬件和软件部分，可以将生物传感技术简单的集成到消费级及工业级终端应用上。所有的产品被设计和制造用于满足关于品质，价格等特性集的消费门槛。NeuroSky因提供了与之相关以及相互补的模块化协同技术解决方案而与众不同。

不保修：神念科技产品系列和相关的文档以“美国情报科学学会”被提供，关于任何形式的适销性警告、非侵害性的知识产权（专利，版权等等）或者有特定目的的适用性担保，我们将不提供任何提示。如果有以下任何情况（包括但不局限于）造成的损害（对利润、业务损失造成的赔偿，更换货物的成本或者损坏或者丢失信息的）而让产品或文档不能使用的，神念科技或其供应商将会承担必要的责任。上述限制可能有一部分对您不适用，那是因为一些司法管辖区会对意外赔偿责任和损害赔偿进行禁止排除或者限制。

神念科技拥有最终用户许可协议。

“为ipod设计”、“为iphone设计”、“为ipad设计”表示一个电子配件已经被设计成用来分别连接到ipod、iphone、ipad，并且已经由开发人员来满足苹果的性能标准认证。苹果对此设备或其遵守的安全操作或者监管标准将不负任何责任。请注意，使用此配件ipod、iphone、ipad可能会影响到无线性能。

第一章

介绍

该指南将教你如何在.NET平台上使用NeuroSky神念科技的ThinkGear SDK去编写Windows应用程序。这些应用程序能够听过NeuroSky的ThinkGear生物传感器系列（也包括Cardio Chip产品系列）获取生物电信号的数据。它将确保你的Windows 应用程序能够从NeuroSky的传感器硬件上接受和使用像心电，脑电这样的生物电数据。

该指南（就整个ThinkGear SDK for .NET而言）面向于已经熟悉在.NET平台上用微软的Visual Studio开发的程序员们。如果您还不是很熟悉在.NET平台上的开发，请访问网址http://www.microsoft.com/net. 去学习如何构建.NET的开发环境和创建典型的.NET应用程序。

如何您已经对熟悉了如何创建典型的.NET应用程序,那么下一步要做的就是下载NeuroSky的ThinkGear SDK for .NET.很有可能的是，如何你正在这个文档时，然后你就已经有了它了。

ThinkGear SDK for .NET 的内容

- ThinkGear SDK for .NET:开发指南以及API参考手册（本文档）
- libs/:
  - ThinkGear.dll 库
  - JayrockJson.dll 支持库
  - NLog.dll 支持库
  - NLog.xml 和 NLog.config 配置文件
- TG-HelloEEG.exe - 构建了一个HelloEEG的实例项目。
- HelloEEG 实例项目的源代码。

你将会在libs/ 文件下发现.dll文件，配置文件以及第三方授权的文件。然后将整个文件复制到你的项目中。

你将会在Sample Projects/HelloEEG文件夹下发现"HelloEEG Sample Project"的源代码。

支持的ThinkGear硬件

ThinkGear SDK for .NET必须使用一个与ThinkGear配套的硬件传感设备。
下面是几款目前与ThinkGear配套的硬件传感设备：

- MindWave Mobile
- MindWave (RF)
- MindBand
- MindSet
- inkCap
- CardioChip Starter Kit Unit
- TGAM module
- CardioChip BMD101 module
- TGAT ASIC
- BMD101 ASIC

注意：在运行TG-SDK for.NET的windows应用程序之前，确保每个ThinkGear硬件设备仔细安装用户手册的说明，将ThinkGear传感器硬件与你的Windows机器配对好。在你的电脑的设备管理器中，ThinkGear传感器必须出现Windows机器的COM端口列表中。

第二章

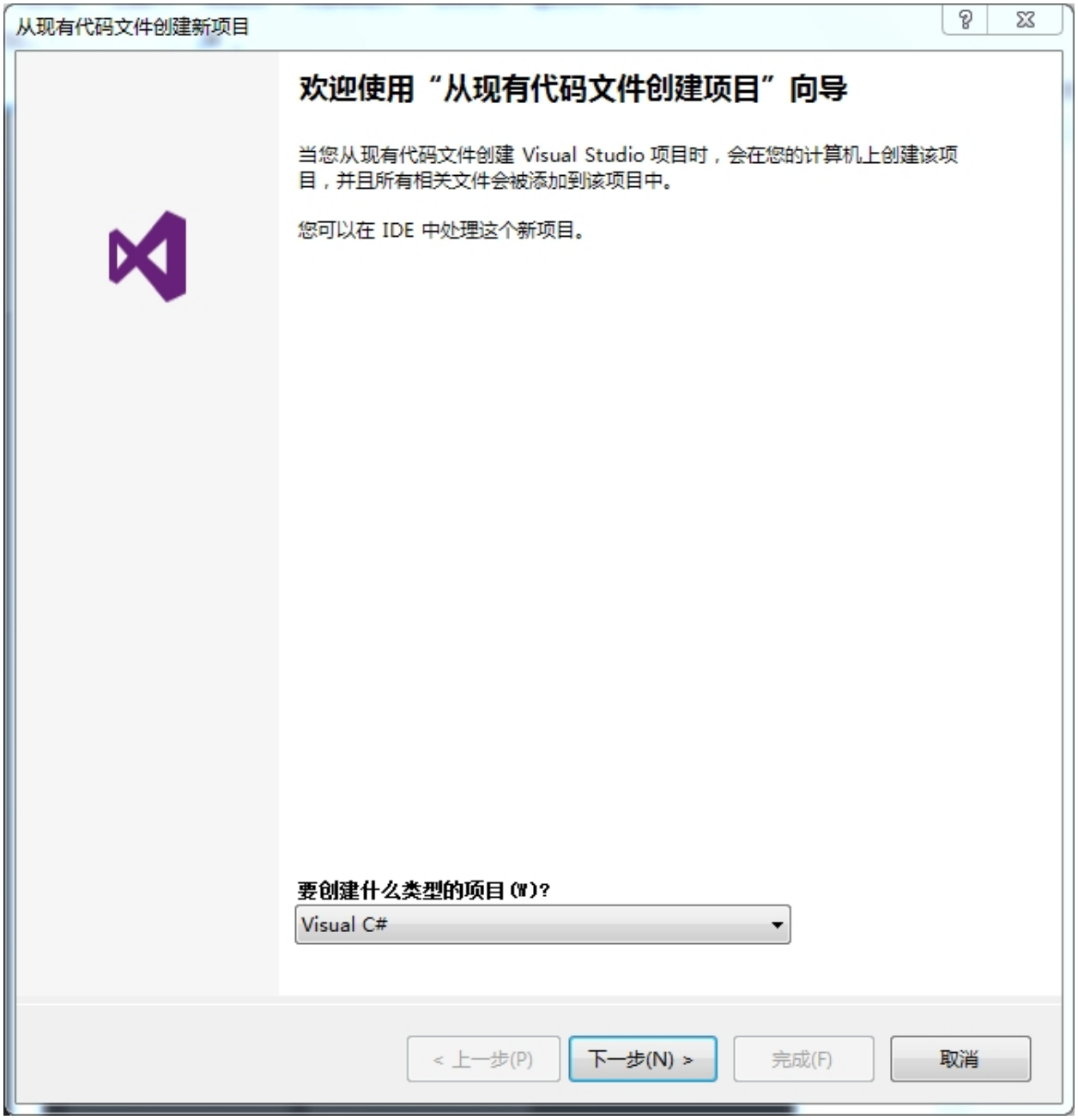
第一个项目：HelloEEG 控制台

HelloEEG是一个包含有ThinkGear SDK for.NET的实例项目，它向我们演示了如何去安装，连接和处理数据到一个ThinkGear设备上去。请将该项目按照以下的步骤添加到你的Visual Studio中：

1.打开Visual Studio,在Visual Studio 工具栏中，点击 文件——>新建——>从现有代码创建项目。

2.在从现有代码文件创建新项目向导中，选择“Visual C#”类型的项目。

3.点击“下一步”按钮。



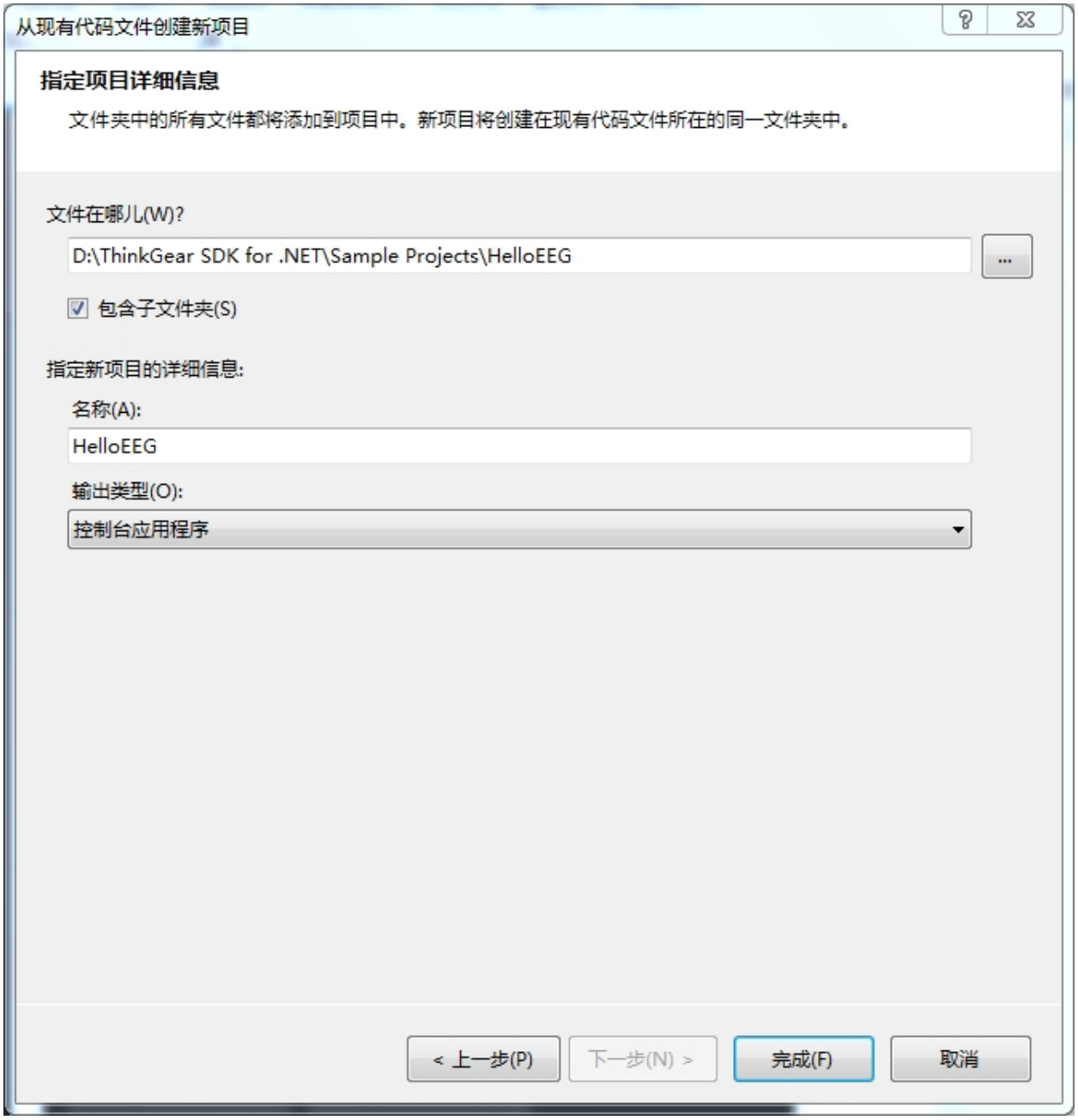
4.在“文件在哪儿?”栏位中，浏览到你已经扩展到的SDK文件的位置("ThinkGear SDK for .NET\SampleProjects\HelloEEG")。

5.选中“包含子文件夹”。

6.在“名称”栏位中，输入”HelloEEG“的名称。

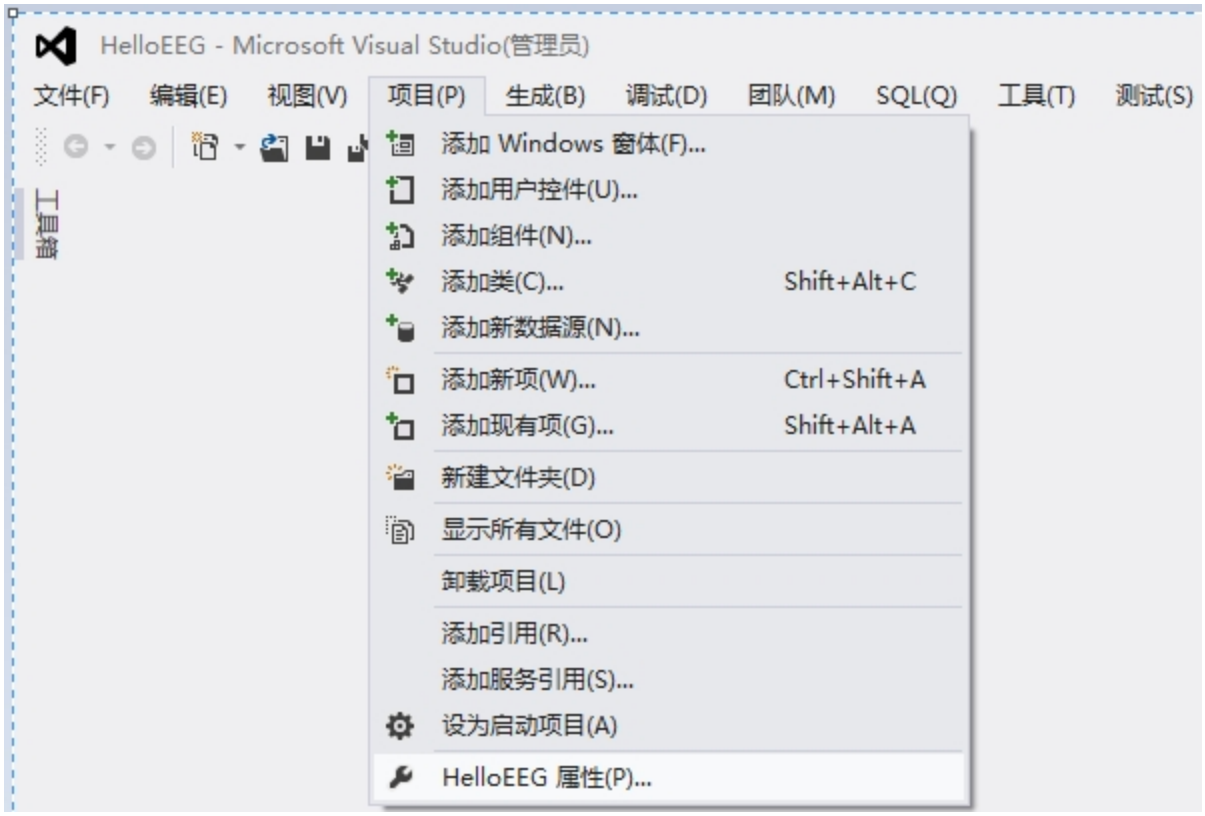
7.在”输出类型“栏位中，选择”控制台应用程序“。

8.点击”完成“按钮。



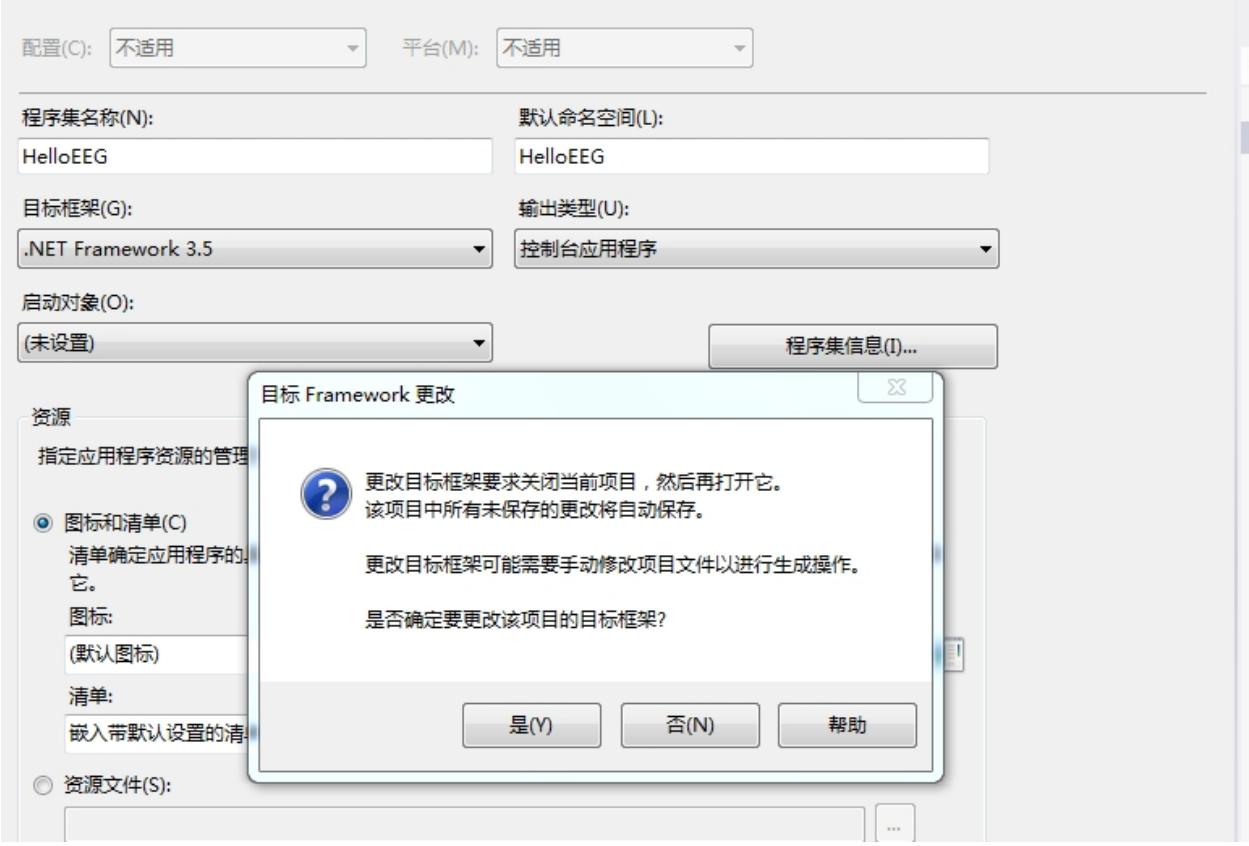
9.在Visual Studio 工具栏中，点击 项目——>HelloEEG属性。

10.在”程序集名称“栏位中，名称改为”HelloEEG“。

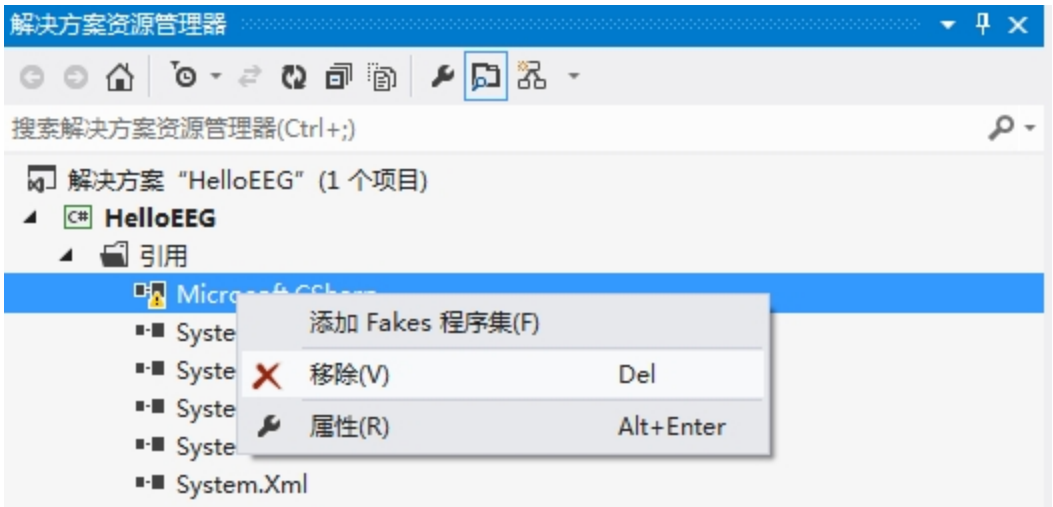


11.在”目标框架“栏位中，改为“.NET Framework3.5”。

12.如果出现确定要更该项目的目标框架选项，则点击“是”。

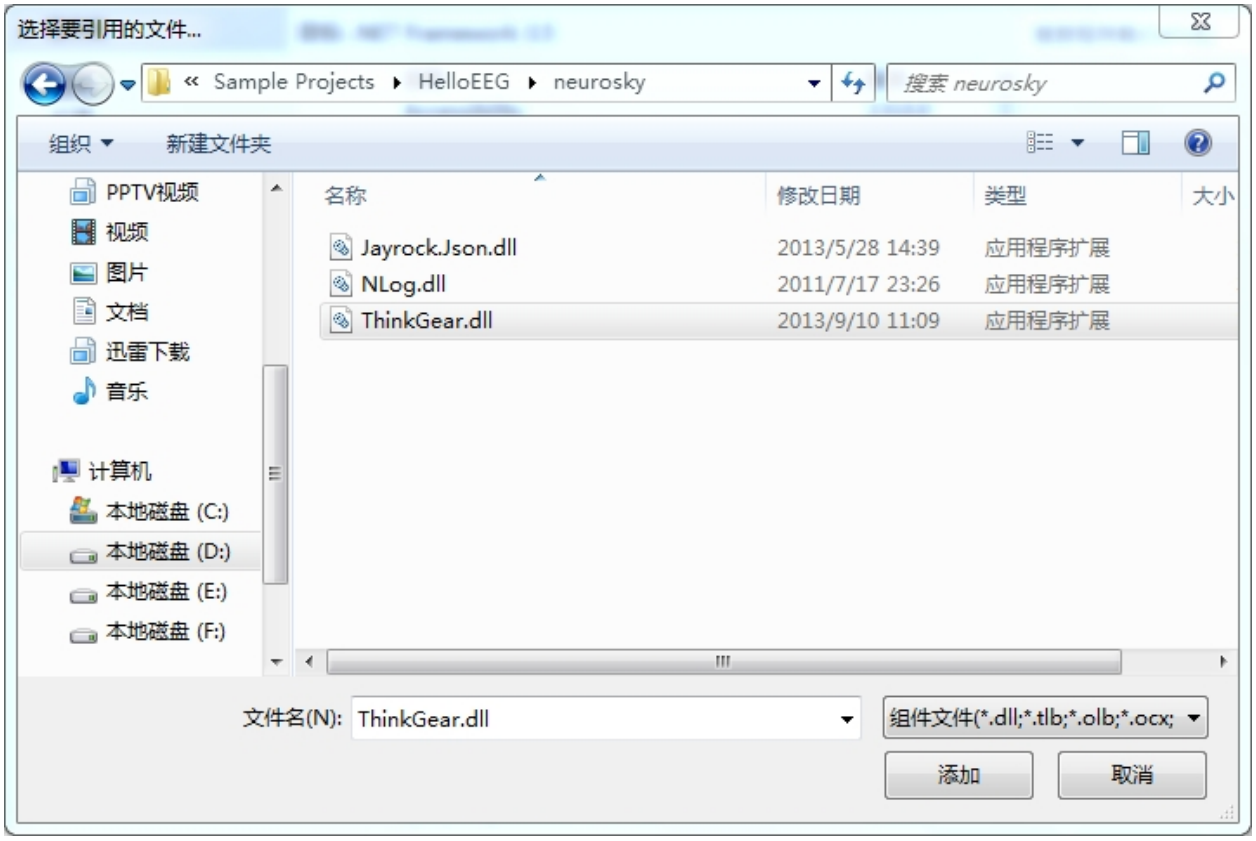


13.在Visual Studio 工具栏中，点击 视图——>解决方案资源管理器。
14.在解决方案资源管理器窗格中，选择并扩展“应用”项。
15.如果你看到你一个感叹号警告在“Microsoft.CSharp”上。
16.选择它并单击右键,移除引用“Microsoft.CSharp”。



17.选择“应用”项，右击，选择“添加应用”。

18.选择”浏览“选项卡,选择“neurosky”的文件夹”,然后选择“ThinkGear.dll”。



19.在Visual Studio 工具栏中,选择“生成”——>生成解决方案。

20.如果没有出现错误的话，你应该能够浏览代码，稍作修改，编辑，运行程序了。

注意：这些步骤都是在Visual Studio 2012 进行的，如果你的步骤有所不同，你可能得去按照这些指令去做。

注意：TG-HelloEEG.exe参考程序是有由这些相同的程序和进程生成的。它有所不同的在于Microsoft ILMerge 程序被用于合并来自“neurosky”的dll文件，生成.exe文件以致可以以更加独立的方式运行。

第三章

开发你自己的基于.NET平台启用ThinkGear的应用

准备你的.NET项目

ThinkGear.NET SDK 的API通过NeuroSky.ThinkGear 命名空间生成应用程序。ThinkGear.dll 文件确保了你的应用程序能够与NeuroSky.ThinkGear 命名空间顺利连接。

ThinkGear.dll

首先，将ThinkGear.dll文件添加到你的.NET应用程序的项目工作区里。ThinkGear.dll是一个C#.NET的类库，它仅仅是作为.NET项目的一部分（它不会在本地的项目中工作）。这个.dll文件包括NeuroSky.ThinkGear 命名空间.

NeuroSky.ThinkGear Namespace

ThinkGear.NET SDK 的API通过NeuroSky.ThinkGear namespace生成应用程序。一旦你已经将ThinkGear.dll文件添加到你的项目中，然后你就能将下面的代码放到你的程序的顶部去连接NeuroSky.ThinkGear namespace:

```
using NeuroSky.ThinkGear;
```

使用 NeuroSky.ThinkGear 命名空间

NeuroSky.ThinkGear 命名空间包括两个类：

- Connector - 连接到计算机的串行COM端口和读取端口的连续流的数据作为DataRowArrays。
- TGParser - 解析一个DataRowArrays去识别你的应用程序能够使用的ThinkGear 数据类型。

使用这些类，首先我们要声明一个Connector实例，然后初始化它。

```
 private Connector connector;
connector = new Connector();
```

接下来，创建EventHandlers 去处理每个Connector Event的类型，链接那些处理程序到Connector 事件。

```
connector.DeviceConnected += new EventHandler( OnDeviceConnected );
 connector.DeviceFound += new EventHandler( OnDeviceFound );
connector.DeviceNotFound += new EventHandler( OnDeviceNotFound );
connector.DeviceConnectFail += new EventHandler( OnDeviceNotFound );
connector.DeviceDisconnected += new EventHandler( OnDeviceDisconnected );
connector.DeviceValidating += new EventHandler( OnDeviceValidating );
```

在处理DeviceConnected 事件中，你应该创建另一个EventHander去处理来自Device的DataReceived 事件，就像这样：

```
void OnDeviceConnected( object sender, EventArgs e ) {

Connector.DeviceEventArgs deviceEventArgs = (Connector.DeviceEventArgs)e;
Console.WriteLine( "New Headset Created." + deviceEventArgs.Device.DevicePortName );
deviceEventArgs.Device.DataReceived += new EventHandler( OnDataReceived );
}
```

现在，每当从设备接收数据，DataReceived 处理程序将会处理这些数据。这儿是一个OnDeviceReceived()实例，显示了如何做到这一点，它使用了TGParser去解析DataRow[ ]:

```
void OnDataReceived( object sender, EventArgs e ){
    /* Cast the event sender as a Device object, and e as the Device's DataEventArgs */
    Device d = (Device)sender;
    Device.DataEventArgs de = (Device.DataEventArgs)e;
    /* Create a TGParser to parse the Device's DataRowArray[] */
    TGParser tgParser = new TGParser();
    tgParser.Read( de.DataRowArray );
    /* Loop through parsed data TGParser for its parsed data... */
    for( int i=0; i<tgParser.ParsedData.Length; i++ ) {
        // See the Data Types documentation for valid keys such
        // as "Raw", "PoorSignal", "Attention", etc.
        if( tgParser.ParsedData[i].ContainsKey("Raw") ){
        Console.WriteLine( "Raw Value:" + tgParser.ParsedData[i]["Raw"] );
        }
        if( tgParser.ParsedData[i].ContainsKey("PoorSignal") ){
```

```
            Console.WriteLine( "PQ Value:" + tgParser.ParsedData[i]["PoorSignal"] );
            }
            if( tgParser.ParsedData[i].ContainsKey("Attention") ) {
            Console.WriteLine( "Att Value:" + tgParser.ParsedData[i]["Attention"] );
            }
            if( tgParser.ParsedData[i].ContainsKey("Meditation") ) {
            Console.WriteLine( "Med Value:" + tgParser.ParsedData[i]["Meditation"] );
            }
            if( tgParser.ParsedData[i].ContainsKey("RespiratoryRate") ){
            Console.WriteLine( "Respiratory Rate:" + tgParser.ParsedData[i]["RespiratoryRate"]);
            }

            if( tgParser.ParsedData[i].ContainsKey("RelaxationLevel") ){
            Console.WriteLine( "Relaxation Level:" + tgParser.ParsedData[i]["RelaxationLevel"]);
            }
        }
    }
```

当你想开始Task Familiarity和Mental Effort 计算时，使用connector实例化对象：

```
connector.setTaskFamiliarityEnable(true);
connector.setMentalEffortEnable(true);
```

一旦你如上所述设置好处理程序,实际上你就可以通过下章节连接到设备所述的连接方法将Connector 连接到任意设备/耳机/端口。如果portName有效，而且连接成功,那么OnDataReceived()方法会在数据到达时耳机时，被自动被调用并执行。

退出之前，应用程序必须调用Connector的close() 方法关闭Connector'打开的连接。

```
connector.close();
```

如果close()没在打开的连接中被调用，那么连接的进程便一直存在。（就好像一个后台线程/进程只关闭了GUI窗口缺没有终止进程本身。）那么耳机仍将保持连接的状态,直到断开前其他进程便无法连接到耳机。

Events 事件

如果你选择通过特定的COM端口连接,需要以下步骤：
1. connector.Connect(portName);

2. connector.Connect 依次验证COM端口。所以会触发 DeviceValidating 事件。
3. 如果COM端口验证有效,它将会连接到设备。 DeviceFound事件不会触发。
4. 如果端口无效, 会触发DeviceNotFound事件。

如果您选择使用自动连接的方法,它将采取以下步骤：
1. connector.Find();
2. 如果能够找到一个有效的带ThinkGear包的COM端口, 便会触发DeviceFound事件，否则 触发DeviceNotFound事件。
3. OnDeviceFound 方法依次调用 connector.Connect(tempPortName); 当temp-PortName为有效COM端口时，会反过来依次调用 DeviceValidating.
4. 如果COM端口验证有效,它将会连接到设备。
5. 如果端口无效, 会触发DeviceNotFound事件。

ThinkGear.NET使用注意事项：

• 为了快速连接,应用程序始终应记住上一次成功连接的端口portName ，然后下次连接时便能优先尝试连接相同的portName。如果该portName 已失效或无法连接，你可以用 ConnectScan(string portName) 方法寻找其他有效的portName。
• 如果发生意外断开,您的应用程序应该自动重新连接，并提示用户按照下述检查他们的耳机设备：
–电池放置正确，电量充足（或更换新电池）
–耳机设备已打开
–耳机设备与蓝牙正确配对
–耳机设备在蓝牙接收器射程范围内（10m畅通）

API 参考

Connector 类

方法

连接到设备

void Connect(string portName) 企图通过portName 打开连接并识别端口名称。调用这个方法会导致下列两个事件之一被触发：
• DeviceConnected – 连接成功并识别portName
• DeviceConnectFail – 尝试连接失败

void ConnectScan() 尝试打开连接到Connector可见的第一个设备。调用后这个方法会导致下列两个事件之一被触发：
• DeviceConnected -连接成功并识别portName
• DeviceConnectFail -尝试连接失败

void ConnectScan(string portName) 和ConnectScan相同，但是要通过 portName识别扫描。调用后这个方法会导致下列两个事件之一被触发：
• DeviceConnected -连接成功并识别portName
• DeviceConnectFail -尝试连接失败

从一个设备断开连接

void Disconnect() 关闭所有连接。调用该方法会导致下面的事件被广播到每个开着的设备中：
• DeviceDisconnected - 设备被关闭。

void Disconnect(Connection connection) 通过connection 关闭特定连接。调用该方法会导致下面的事件被广播到特定的设备中：
• DeviceDisconnected - 设备被关闭。

void Disconnect(Device device) 通过device关闭特定连接。调用该方法会导致下面的事件被广播到特定的设备中：
• DeviceDisconnected - 设备被关闭。

向设备发送字节

void Send(string portName, byte[] bytesToSend) 向特定端口发送的字节数组
Configure Task Familiarity/Mental Effort

void enableTaskDifficulty() DEPRICATED 开始记录数据60秒。一旦记录完成，会计算好任务难度值。注意：Mental Effort 的初始计算值为0。

void enableTaskFamiliarity() DEPRICATED 开始记录数据60秒。一旦记录完成，会计算好任务熟悉度。注意：Task Familiarity的初始计算值为0。

Events

DeviceFound
当找到ThinkGear设备时发生。这就是应用程序选择是否连接到该端口的位置。

DeviceNotFound
当找不到ThinkGear设备时发生。这通常是应用程序显示错误【找不到任何设备】的位置。

DeviceValidating
在Connector尝试串行端口前发生。主要用于通知在试图连接端口的GUI。

DeviceConnected
当ThinkGear设备连接完成时发生。这就是应用程序为设备链接到OnDataReceived的位置。

DeviceConnectFail
当Connector与特定端口连接失败是发生。

DeviceDisconnected
当Connector和ThinkGear设备断开连接时发生。

DataReceived
当能从ThinkGear设备读取数据时发生。

TGParser 类

TGParser类相当于用数据字典将接收到的数值转换成可理解的数据。

Methods

Dictionary<string, double>[] Read(DataRow[] dataRow)

解析耳机的dataRow返回一个数据字典的可用数据。它也存储在ParsedData属性的字典中。当MindSet,MindWave或MindWave Mobile连接时，Read() 方法会如下返回该字典中的标准值。

键值	描述	数据类型
Time	接受到TimeStamps的包	double
Raw	原始EEG数据	short
EegPowerDelta	Delta值	uint
EegPowerTheta	Theta值	uint
EegPowerAlpha1	低Alpha值	uint
EegPowerAlpha2	高Alpha值	uint
EegPowerBeta1	低Beta值	uint
EegPowerBeta2	高Beta值	uint
EegPowerGamma1	低Gamma值	uint
EegPowerGamma2	高Gamma值	uint
Attention	专注度eSense值	double
Meditation	冥想度eSense值	double
Zone	性能区域	double
PoorSignal	弱信号值	double
BlinkStrength	检测眨眼的强度。Blink强度范围从1（小强度眨眼）到255（大强度眨眼）。只有当PoorSignal值少于51时，眨眼值才会被计算。	unit
Task Familiarity	可以用来比较测试对象学习新技能（机械）的熟练度。一分钟收集的数据构成一次实验,并会产生一个熟悉度索引值。分次实验得到的熟	double

	悉度索引值可以用来比较。	
Mental Effort	可以用来比较测试对象脑力劳动的使用值。一分钟收集的数据构成一次实验,并会产生一个脑力劳动索 引值。分次实验获取的脑力劳动值可以用来比较。	double

当连接到ThinkCap时，Read()方法会如下返回该字典中的值：

键值	描述	数据类型
Time	接受到TimeStamps的包	double
RawCh1	EEG 通道1	short
RawCh2	EEG 通道2	short
RawCh3	EEG 通道3	short
RawCh4	EEG 通道4	short
RawCh5	EEG 通道5	short
RawCh6	EEG 通道6	short
RawCh7	EEG 通道7	short
RawCh8	EEG 通道8	short

当连接到BMD10X时，Read()方法会如下返回该字典中的值：

键值	描述	数据类型
PoorSignal	弱信号/信号状态	short
Raw	AD转换器的最小处理样本	short
HeartRate	用户的瞬时心率 (BPM)	double
RrInt	心跳间隔的时间,以毫秒为单位	unit

RespiratoryRate	用户每分钟呼吸率	double
Relaxation	从用户的心电图获取其放松程度	unit

详细描述

ZONE 区域

这个值记录了当前目标的表现区域，取值范围从0到9，而且这个值只有当目标从一个区域过渡到另一个区域时才会传送。
该算法使用专注度和冥想度指导目标的最佳性能。
要进入精英Zone (9)，目标必须保持他们的注意力水平至少值82,同时持有冥想水平稳定或增加。
要进入中等 Zone (5), 目标必须保持他们的注意力水平至少值67,同时持有冥想水平稳定或增加。
要进入初级Zone(1), 目标必须保持他们的注意力水平至少值53,同时持有冥想水平稳定或增加。
还未就绪状态 Zone (0) 是指所有注意力水平低于53并且目标的冥想的初级Zone水平也在下降。.

如果传感器没能和人体有效接触，所有Zone的计算会暂停，其数值也会重置。

注意: 相比其他NeuroSky的产品，这是不同的性能实现Zone。
参考: Golf Putting Training Algorithm v 2.0, 2012年9月, Dr. KooHyoung Lee.

ThinkGear数据类型

ThinkGear的数据类型可以大致分成三个部分：只适用于EEG的传感器设备的数据类型，只适用于ECG/EKG（CardioChip）的传感器设备的数据类型，适用于所有启用ThinkGear的设备的数据类型，包括EEG和ECG/EKG。

通用型

这些数据类型可以适用于大部分类型的ThinkGear硬件设备。

POOR_SIGNAL/SENSOR_STATUS

这些整型值是用来反应传感器芯片上生物信号强弱的。该值通常由ThinkGear硬件设备每秒输出一次。

这个值非常重要，任何应用程序都需要它来读懂，理解，处理ThinkGear传感器的数据。根据你的应用和用户的使用情况，你的应用程序是否需要进行调整取决于当前的POOR_SIGNAL / SIGNAL_STATUS值。比如，假设这个值表明生物传感器还没接触到物体，那么这段时间接收到的RAW_DATA 或 EEG_POWER应该被算作浮点噪音，而不是从人体获取的数据，基于应用的需求这些数据应被舍弃。这个也可以用来让用户调节传感器或第一次使用设备时的基准。

重要：这个更新版本可以读取不同的硬件设备poorSignal值并转换成统一格式（区别于旧版本的SDK）。如果你有针对poorSignal值的软件,你应该评估软件是否需要适当改变。

Poor signal 会由很多因素引起. 按照其影响由强到弱排序，分别是：
• 传感器、地面或参考电极不在人体的头上或身体上(即当用户没有穿戴ThinkGear设备)。
• 传感器、地面或参考电极与人的皮肤接触不良。(如头发的遮挡,耳机设计与大脑不能贴合,耳机没有正确佩戴等)。.

• 佩戴者过多的行为 (如过多的摇晃大脑、身体，碰撞耳机或传感器).
• 过度的环境静电噪音 (一些环境有很强的电信号或是人穿戴传感器时有静电积聚).
• 过度的生物噪声(如不需要的EMG, EKG/ECG, EOG, EEG等)

对于EEG来说, ThinkGear传感器的正常使用下有一些噪音干扰是不可避免的，为此神念科技设计了过滤技术和算法来检测，纠正，补偿，解释，承担各种类型的信号噪声。大多数用户都是只对eSence的值有兴趣，比如专注度和冥想度的值。不用太担心POOR_SIGNAL的影响，只要注意POOR_SIGNAL的值为0时就能取到专注度和冥想度的值，并且没带耳机的POOR_SIGNAL是200。POOR_SIGNAL的值针对一些对噪声更敏感的应用（比如一些医学或研究的应用程序）或者需要很快检测出轻微噪音的应用更有帮助。

RAW_DATA

这个数据类型提供了生物传感器的原始样本值。采样率(同时产生的输出率)，大概的范围值和这些值的解释（从原始单位转换为伏特单位）依赖于ThinkGear硬件设备执行抽样的硬件特点。你必须参考ThinkGear硬件的各种类型的技术文档了解细节来支持你的应用程序。
例如，大多数ThinkGear设备样本频率值设为512Hz，具体范围可以从-32768到32767。
再比如，要把基于TGAT 的EEG传感器的值（如TGAT, TGAM, MindWave Mobile, MindWave, MindSet） 转换为伏特值，要这样做：

(rawValue * (1.8/4096) / 2000

注意：CardioChip或者基于BMD10X的设备的 ECG/EKG元数据必须使用不同的转换方式。

RAW_MULTI

这个数据类型不被目前任何正在支持的ThinkGear产品所使用。将其保存
这里的目的是兼容一些已淘汰的产品,也能看作是未来产品可能需要的一个占位符。

EEG

这些数据类型只适用于EEG传感器硬件设备，例如 MindWave Mobile， MindSet， MindBand和TGAM 芯片、模块。s

ATTENTION专注度

“eSense专注度指数”表明了使用者精神“集中度”水平或“注意度”水平的强烈程度，例如，当你能够进入高度专注状态并且可以稳定地控制你的心理活动，该指数的值就会很高。该指数值的范围是0到100。心烦意乱、精神恍惚、注意力不集中以及焦虑等精神状态都将降低专注度指数的数值。请参见eSense指数内容了解eSense指数的详细说明。

默认情况下会启用该数值的 输出,通常每秒输出一次。

MEDITATION冥想度

“eSense冥想度指数” 表明了使用者精神“平静度”水平或者“放松度”水平。该指数值的范围是0到100。需要注意的是，放松度指数反映的是使用者的精神状态，而不是其身体状态，所以，简单地进行全身肌肉放松并不能快速地提高放松度水平。然而，对大多数人来说，在正常的环境下，进行身体放松通常有助于精神状态的放松。放松度水平的提高与大脑活动的减少有明显的关联。长期观察结果表明：闭上眼睛可以使得大脑无需处理通过眼睛看到的景象从而降低大脑精神活动水平。所以，闭上眼睛通常是提高放松度值的有效方法。心烦意乱、精神恍惚、焦虑、激动不安等精神状态以及感官刺激等都

将降低放松度指数的数值。请参见eSense指数内容了解eSense指数的详细说明。

默认情况下会启用该数值的 输出,通常每秒输出一次。

eSense™ 指数

对所有不同类型的eSense(如专注度和冥想度)其指数以1到100之间的具体数值来指示。数值在40和60之间表示此刻该项指数的值处于一般范围，这一数值范围类似于常规脑电波测量技术中确定的“基线”。（但是ThinkGear的基线测定方法是自有的专利技术，与常规脑电波的基线测定办法不同）。 数值在60—80之间表示此刻该项指数的值处于“较高值区”，也就是说略高于正常水平 （即当前情况下你的专注度或者是放松度比正常情况下要高）。数值在80—100之间表示处于“高值区”。它表示你的专注度或放松度达到了非常高的水平，即处于非常专注的状态或者是非常放松的状态。

同理，如果数值在20—40之间则表示此时的eSense指数水平处于“较低值区”，数值 1—20则意味着处于“低值区”。与前述其它区值所代表的人的精神状态相反，eSense指数处于这2个区域则表示被试者的精神状态表现为不同程度的心烦意乱、焦躁不安、行为反常等。

ZONE 区域

这个值记录了当前目标的表现区域，取值范围从0到9，而且这个值只有当目标从一个区域过渡到另一个区域时才会传送。
该算法使用专注度和冥想度指导目标的最佳性能。
要进入精英Zone (9)，目标必须保持他们的注意力水平至少值82,同时持有冥想水平稳定或增加。
要进入中等 Zone (5), 目标必须保持他们的注意力水平至少值67,同时持有冥想水平稳定或增加。
要进入初级Zone(1), 目标必须保持他们的注意力水平至少值53,同时持有冥想水平稳定或增加。
还未就绪状态 Zone (0) 是指所有注意力水平低于53并且目标的冥想的初级Zone水平也在下降。.

如果传感器没能和人体有效接触，所有Zone的计算会暂停，其数值也会重置。

> **注意:** 相比其他NeuroSky的产品，这是不同的性能实现Zone。
> 参考: Golf Putting Training Algorithm v 2.0, 2012年9月, Dr. KooHyoung Lee.

BLINK眨眼
这个整型值记录了用户最近眨眼的强度。取值范围从1到255，每次检测到眨眼变会记录数据。这个值表示相对眨眼强度没有单位。

必须启用眨眼检测。

```
if (setBlinkDetectionEnabled(true)) {
```

```
        // return true, means success
        Console.WriteLine("HelloEEG: BlinkDetection is Enabled");
        }
        else {
        // return false, meaning not supported because:
        // + connected hardware doesn't support
        // + conflict with another option already set
        // + not support by this version of the SDK
        Console.WriteLine("HelloEEG: BlinkDetection can not be Enabled");
}
```

当前配置可以恢复。

```
if (getBlinkDetectionEnabled()) {
        // return true, means it is enabled
        Console.WriteLine("HelloEEG: BlinkDetection is configured");
        }
        else {
        // return false, meaning not currently configured
        Console.WriteLine("HelloEEG: BlinkDetection is NOT configured");
}
```

注意: 如果这些方法在MSG_MODEL_IDENTIFIED恢复之前被调用，则它被认为是一个鉴定连接设备时需要处理的请求。很有可能这个特性被启用后突然发现它不再启用了。一旦连接设备被鉴定，如果请求和软硬件不兼容，他就会被重写，并且发送MSG_ERR_CFG_OVERRIDE作为提醒。

EEG_POWER

这个数值代表当前普遍认可的8个量级的EEG频率带。
这8个EEG频率段是: delta (0.5 - 2.75Hz), theta (3.5 - 6.75Hz), low-alpha (7.5 - 9.25Hz), high-alpha (10 - 11.75Hz), low-beta (13 - 16.75Hz), high-beta (18 - 29.75Hz), low-gamma (31 -39.75Hz), 和 mid-gamma (41 - 49.75Hz). 这些值没有单位,只是用来和其他频率段的样品值作对比。

默认情况下,启用此数据值的输出,输出大约每秒一次。

THINKCAP_RAW

这个数据类型不被目前任何正在支持的ThinkGear产品所使用。将其保存这里的目的是兼容一些已淘汰的产品,也能看作是未来产品可能需要的一个占位符。

POSITIVITY积极性

取值范围从-100到+100，表明Values -100 to +100, 表明目标对某事的关注度,负值越多说明目标关注度越低，正值越高说明越关注。

注意: 目前这个功能还不可用。

FAMILIARITY 熟练度

该算法试图表达目标对某种运动技能的熟练程度,它可以和一些复杂算法结合，研究已掌握的运动技能的不同方面。但它也可以单独使用。
它可以用来比较一个测试对象学习一个新技能的熟练度。一分钟的收集的数据构成一次试验,并且会产生一个针对此人的熟练度索引值。运动技能单独试验的熟练度索引可以用来作为同一个人的比较。
熟练度索引值是浮点数，没有单位，只是用来对比来自同一个人的其他数据。有利于展示出从基线 (或过去的试验)到当前试验的变化百分比。注意不同值可以表示正面也可以表示负面的。
观察HelloEEG示例应用程序是如何应用这些信息的。
熟练度任务的计算必须已经被启用。

```
if (setTaskFamiliarityEnable(true)) {
        // return true, means success

        Console.WriteLine("HelloEEG: TaskFamiliarity is Enabled");
}
else {
        // return false, meaning not supported because:
        // + connected hardware doesn't support
        // + conflict with another option already set
        // + not support by this version of the SDK

        Console.WriteLine("HelloEEG: TaskFamiliarity can not be Enabled");
}
```

当前设备可以恢复。

```
if (getTaskFamiliarityEnable()) {
        // return true, means it is enabled
        Console.WriteLine("HelloEEG: TaskFamiliarity is configured");
        }
else {
        // return false, meaning not currently configured
        Console.WriteLine("HelloEEG: TaskFamiliarity is NOT configured");
}
```

启用后,该算法将执行一次。收集到60秒的可用数据便立即执行。发布结果后,算法便自动结束。

可以配置算法连续运行。但启用连续选项不会自动启用该算法，设置完RunContinuous后，你必须同时启用算法。

```
if (setTaskFamiliarityRunContinuous(true)) {
        // return true, means success

        Console.WriteLine("HelloEEG: TaskFamiliarity Continuous operation");
        }
else {
        // return false, meaning not supported because:
        // + connected hardware doesn't support
        // + conflict with another option already set
        // + not support by this version of the SDK

        Console.WriteLine("HelloEEG: TaskFamiliarity normal operation ");
}
```

当前配置可被恢复。

```
if (getTaskFamiliarityRunContinuous()) {
        // return true, means it is enabled

        Console.WriteLine("HelloEEG: TaskFamiliarity Continuous operation");
}
else {
        // return false, meaning not currently configured

        Console.WriteLine("HelloEEG: TaskFamiliarity normal operation");
}
```

注意: 如果这些方法在MSG_MODEL_IDENTIFIED恢复之前被调用，则它被认为是一个鉴定连接设备时需要处理的请求。很有可能这个特性被启用后突然发现它不再启用了。一旦确了连接设备，若请求和软硬件不兼容，它会被重写，再发送提醒消息MSG_ERR_CFG_OVERRIDE。

注意:该算法是资源和计算密集型的。如果你需要运行Android调试器,请注意这个计算可能需要好几分钟才能完成。然后它会完成计算并给出结果。不用调试器,这个计算将在几秒钟完成。

MENTAL EFFORT脑力劳动

该算法试图表达目标的脑力劳动值, 他可以和熟练度算法一起使用来检查学习技能的不同方面。但它也可以单独使用。

它可以用来比较一个测试对象掌握一个新技能的困难程度。一分钟的收集的数据构成一次试验, 并且会产生一个针对此人的脑力劳动困难度索引值。运动技能单独试验的困难度索引值可以用来作为同一个人的比较。

脑力劳动索引值是浮点数，没有单位，只是用来对比来自同一个人的其他数据。有利于展示出从基线 (或过去的试验)到当前试验的变化百分比。注意不同值可以表示正面也可以表示负面的。观察HelloEEG示例应用程序是如何应用这些信息的。

脑力劳动任务的计算必须已经被启用。

```
if (setMentalEffortEnable(true)) {
// return true, means success
Console.WriteLine("HelloEEG: MentalEffort is Enabled");
}
else {
// return false, meaning not supported because:
// + connected hardware doesn't support
// + conflict with another option already set
// + not support by this version of the SDK
Console.WriteLine("HelloEEG: MentalEffort can not be Enabled");
}
```

当前配置可被恢复。

```
if (getMentalEffortEnable()) {
// return true, means it is enabled
Console.WriteLine("HelloEEG: MentalEffort is configured");
}
else {
// return false, meaning not currently configured
Console.WriteLine("HelloEEG: MentalEffort is NOT configured");
```

}

启用后,该算法将执行一次。收集到60秒的可用数据便立即执行。发布结果后,算法便自动结束。

可以配置算法连续运行。但启用连续选项不会自动启用该算法，设置完RunContinuous后，你必须同时启用算法。

```
if (setMentalEffortRunContinuous(true)) {
// return true, means success
Console.WriteLine("HelloEEG: MentalEffort Continuous operation");
}
else {
// return false, meaning not supported because:
// + connected hardware doesn't support
// + conflict with another option already set
// + not support by this version of the SDK
Console.WriteLine("HelloEEG: MentalEffort normal operation ");
}
```

当前配置可被恢复。

```
if (getMentalEffortRunContinuous()) {
// return true, means it is enabled
Console.WriteLine("HelloEEG: MentalEffort Continuous operation");
}
else {
// return false, meaning not currently configured
Console.WriteLine("HelloEEG: MentalEffort normal operation");
}
```

注意: 如果这些方法在MSG_MODEL_IDENTIFIED恢复之前被调用，则它被认为是一个鉴定连接设备时需要处理的请求。很有可能这个特性被启用后突然发现它不再启用了。一旦确了连接设备，若请求和软硬件不兼容，它会被重写，再发送提醒消息MSG_ERR_CFG_OVERRIDE。

注意:该算法是资源和计算密集型的。如果你需要运行Android调试器,请注意这个计算可能需要好几分钟才能完成。然后它会完成计算并给出结果。不用调试器,这个计算将在几秒钟完成。

ECG/EKG 心电

这些数据类型只适用于ECG/EKG传感器(CardioChip)硬件设备，例如CardioChip Starter Kit Unit 和 BMD10X 芯片、模块。

HEART_RATE 心率

这个整型值记录了当前的用户的心率，以每分钟的跳动为单位(BPM)。与许多其他设备常见的心率报告不同，该值是根据每个用户实际心跳的R-peak（波峰）间隔时间精确实时计算的。这将得到一个非常精确并连续记录的随每个用户的实际的心跳而变化的心率值。
为了方便得到类似其他ECG/EKG 设备一样普遍的“平滑，平均”心率值，使用下面描述的平滑的心率的值作为输入。

Smoothed Heart Rate 平滑的心率

通常，在多数ECG/EKG设备上查看心率值时，显示“平滑的”值可以表明基于目标的自然HRV心率中不存在有节奏的波动。差不多相同 “平滑的”效果可以通过精确的HEART RATE值获得，利用SDK提供的HeartRateAcceleration 类的getAcceleration ()方法即可。

查看心跳加速章节中如何计算平滑心率的描述，接着可到API参考的HeartRateAcceleration类获取全部细节。

Heart Rate Acceleration 心跳加速

一个潜在的有用的心率指标就是加速率。正加速度值表示用户在一个特定的时间段内（如超过10秒）由一定数量的BPM得知心脏速率加快，而负加速度值表示用户的心脏速率在给定时间内一定数量的BPM减慢了。当开始运动,或在运动过后,这种加速度量可以作为一个人分别在参加运动时心跳的加快和运动后心跳多快恢复正常的指标。

为了计算心跳加速率（或者平缓心跳率），首先要初始化应用中的HeartRateAcceleration()对象：

```
HeartRateAcceleration heartRateAcceleration = new HeartRateAcceleration();
```

这将使用10秒的时间段初始化计算。（您也可以根据你应用的需要选择重载的构造函数用更长或更短的时间来初始化计算。）

接着，当新的Heart Rate可用时，可以像这样获取平缓心跳率和加速率：

```
int[] result = heartRateAcceleration.getAcceleration( heartRate, poorSignal );
if( result[0] != -1 ) {
        int smoothedHeartRate = result[0];
        int heartRateAcceleration = result[1];
}
```

利用API参考的HeartRateAcceleration类获取全部细节。

Target Heart Rate for Physical Training 体能训练的目标心率

给定用户的年龄和性别信息,可以确定一个他们的心率范围是否可以达到特定的体能训练“区域”。加上传感器的 HEART RATE信息，应用程序可以告诉用户他们的当前心率是否在目标训练区域中（例如运动后正常）。

为了确定人的目标心率范围, 首先要创建 TargetHeartRate 对象:

```
TargetHeartRate targetHeartRate = new TargetHeartRate();
```

接着，要让用户在任何时候可以确定目标心率范围(最小到最大值)是否达到特定的体能训练区域，使用getTargetHeartRate()方法：

```
int age = 25;
String gender = "Male";
String zone = "Aerobic";
int[] range = targetHeartRate.getTargetHeartRate( age, gender, zone );

int lowerBound = range[0];
int upperBound = range[1];
```

下限值和上限值可以用来比对用户的HEART_RATE 或 平滑心跳率来确定用户的目标范围是否适用于目标训练区域。

性别必须是“男”或者“女”。区域必须是下列之一：

- "Light Exercise"
- "Weight Loss"
- "Aerobic"
- "Conditioning"
- "Athletic"

如果任何参数有误，那么该方法会返回int[] of -1，-1 。

重要: 测量心率不应该在用户从事体力活动的时候,用户需要暂时停止活动,然后再测量他们的心率。

（参考文献）

1.http://www.heart.org/HEARTORG/GettingHealthy/PhysicalActivity/Target-Heart-Rates_UCM_434341_

2. http://www.cdc.gov/physicalactivity/everyone/measuring/heartrate.html

3. http://www.heart.com/heart-rate-chart.html

4. http://www.thewalkingsite.com/thr.html

Heart Fitness Level 健康心脏水平

给定一个人的年龄,性别,和当前的静息心率,可以大致得到该人目前的心脏健康值，并能标记为 “欠佳”,“低于平均水平”,“一般”,“高于平均水平”,“好”,“优秀”或“运动员”。

要确定一个人的心脏的健康水平,首先创建一个HeartFitnessLevel对象:

```
HeartFitnessLevel heartFitnessLevel = new HeartFitnessLevel();
```

然后,一旦获取了一个人的年龄,性别,和当前心率,既使用getHeartFitnessLevel()方法:

```
int age = 25;
String gender = "Male";              // "Male" or "Female"
int restingHR = 60;
String heartFitnessLevel = heartFitnessLevel.getHeartFitnessLevel( age, gender, restingHR );
```

性别必须是“男”或者“女”，否则该方法会返回空字符串（“”）。

heartFitnessLevel的值会返回为下列之一： “欠佳”,“低于平均水平”,“一般”,“高于平均水平”,“好”,“优秀”或“运动员”。

（参考文献）

1. http://www.topendsports.com/testing/heart-rate-resting-chart.htm

RELAXATION 放松度

放松度数值可以基于用户的心率可变性(HRV)特征，提示用户其心跳是否属于放松,或者兴奋，压力，疲劳。取值范围从1到100。高放松值趋向于表示一种放松的状态,而低放松值则趋向于表示兴奋,压力,或疲劳。

要通过MSG_RELAXATION接收值给Handler,只需将TGDevice连接到ThinkGear ECG/EKG传感器(CardioChip)，即用户能至少持续一分钟获取良好，干净的ThinkGear ECG/EKG传感器(CardioChip)的信号(SENSOR_STATUS = = 200每分钟)。如果信号中断,SENSOR_STATUS超过200,那么这个计算将重置并重新开始,再接收一分钟纯数据返回MSG_RELAXATION值。

为达到最佳效果,用户在数据收集时应平静端坐。

（参考文献）

1. Neurosci Biobehav Rev. 2009 Feb; 33(2): 71-80. Epub 2008 Jul 30. Heart rate variability explored in the frequency domain: a tool to investigate the link between heart and behavior. Montano N, Porta A, Cogliati C, Costantino G, Tobaldini E, Casali KR, Iellamo F.

2. Int J Cardiol. 2002 Jul; 84(1): 1-14. Functional assessment of heart rate variability: physiological basis and practical applications. Pumprla J, Howorka K, Groves D, Chester M, Nolan J.

3. International Conference on Computer and Automation Engineering. A Review of Measurement and Analysis of Heart Rate Variability. Dipali Bansal, Munna Khan, A. K. Salhan.

4. Neurosci Biobehav Rev. 2009 Feb; 33(2): 81-8. Epub 2008 Aug 13. Claude Bernard and the heart-brain connection: further elaboration of a model of neurovisceral integration. ayer JF, Lane RD.

Respiratory Rate 呼吸率

呼吸率数据值记录了用户大概每分钟的呼吸率。它是根据用户的ECG/EKG和心率变异性(HRV)特征计算的。

呼吸率通过应用程序的RespiratoryRate 键值接受，示例代码如下：

```
if( tgParser.ParsedData[i].ContainsKey("RespiratoryRate") ){
        Console.WriteLine( "Respiratory Rate:" + tgParser.ParsedData[i]["RespiratoryRate"]);
}
```

为达到最佳效果,用户在数据收集时应平静端坐。

(参考文献)

1. Rosenthal, Talma, Ariela Alter, Edna Peleg, and Benjamin Gavish. "Device-guided breathing exercises reduce blood pressure: ambulatory and home measurements." American Journal of Hypertension. 14. (2001): 74–76.

呼吸率 算法必须启用。一旦启用后便会持续运行直到被禁用。

```
if (setRespirationRateEnable(true)) {
        // return true, means success
        Console.WriteLine( "HelloEKG: RespirationRate is Enabled");
}
else {
        // return false, meaning not supported because:
        // + connected hardware doesn't support
        // + conflict with another option already set
        // + not support by this version of the SDK
        Console.WriteLine( "HelloEKG: RespirationRate can not be Enabled");
}
```

当前配置可以被恢复。

```
if (getRespirationRateEnable()) {
// return true, means it is enabled
Console.WriteLine( "HelloEKG: RespirationRate is configured");
}
else {
// return false, meaning not currently configured
Console.WriteLine( "HelloEKG: RespirationRate is NOT configured");
}
```

注意:如果这些方法在MSG_MODEL_IDENTIFIED接收之前被调用, 则它被认为是一个鉴定连接设备时需要处理的请求。很有可能这个特性被启用后突然发现它不再启用了。一旦确定了连接设备，若请求和软硬件不兼容，它会被重写，再发送提醒消息MSG_ERR_CFG_OVERRIDE。

注意:该算法是资源和计算密集型的。如果你需要运行Android调试器,请注意这个计算可能需要好几分钟(估计5分钟)才能完成。然后它会完成计算并给出结果。不用调试器,这个计算将在几秒钟完成。

Heart Risk Awareness 心脏病风险意识

心脏风险意识数值的目的是为了提高用户的健康意识，比如发现HRV非常低时。低HRV已被证实会增加心脏病发作的风险和死亡率。

要确定一个人的心脏风险意识,先创建NeuroSkyHeartMeters对象：

```
NeuroSkyHeartMeters neuroSkyHeartMeters = new NeuroSkyHeartMeters();
```

然后用下面两个方法之一来计算：

l 使用R-R间隔集

当应用Handler接收到MSG_EKG_RRINT消息,保存R-R间隔值到一个缓冲区。一旦缓冲区有至少60个R-R间隔，使用NeuroSkyHeartMeters 类的calculateHear-tRiskAware(Integer[] rrIntervalInMS)方法：

```
private final Handler handler = new Handler() {
ArrayList<Integer> temp_rrintBuffer = new ArrayList<Integer>();
Integer[] rrinterBuffer = new Integer[60];
@Override
public void handleMessage( Message msg ) {
switch( msg.what ) {
//...
case MSG_EKG_RRINT:
temp_rrintBuffer.add( msg.arg1 );
if( temp_rrintBuffer.size()==60 ) {
for( int i = 0; i<60; i++ ) {
rrintBuffer[i] = temp_rrintBuffer.get(i);
}
temp_rrintBuffer.clear();
int heartRiskAwareness = neuroSkyHeartMeters.calculateHeartRiskAware(
rrintBuffer );
}
break;
//...
} /* end switch on message type */
} /* end handleMessage() */
}; /* end Handler */
Using Storage Data
Simply use the calculateHeartRiskAware( String fileName ) method in the NeuroSkyHeart-
Meters class:
int heartRiskAwarness = neuroSkyHeartMeters.calculateHeartRiskAware( "john" );
```

注意：方法里的“fileName“参数是储存了已计算好的心脏年龄的文件名。

Results 结果

返回值将是一个“心脏病风险意识”指数,其值为“0”,“1”,“2”或“3”。可以根据用户的“心脏风险意识”由应用程序将以下信息反馈给用户：

HeartRiskAwareness = 0

你的HRV此时并不低。低HRV已被证实会增加心脏病发作的风险和死亡率。现在你的HRV值表明你目前承担的风险有限或没有风险。

HeartRiskAwareness = 1

你的HRV相对较低。低HRV已被证明会增加心脏病发作的风险和死亡率。 建议你多运动，注意饮食。你可以吃些可预防心脏病的食物,如坚果、鱼类、粗粮、蔬菜，多喝绿茶。

HeartRiskAwareness = 2

你的HRV很低。低HRV已被证实会增加心脏病发作的风险和死亡率。会影响心脏的健康的坏习惯有：总吃高脂高糖的食物,抽烟,酗酒,缺乏锻炼,心理压力高,长时间的睡眠不足。建议你改变这些坏习惯，适量饮酒,健康饮食,适当锻炼,控制体重,养成良好的睡觉习惯,保持和平的心态。

HeartRiskAwareness = 3

你的HRV非常低。低HRV已被证明会增加心脏病发作的风险和死亡率。建议你改变你的一些习惯。你要考虑戒烟,停止饮酒，还要确保一定的运动量,控制你的体重,养成良好的睡眠习惯,饮食多吃纤维少盐,保持和平的心态。心脏病的症状包括胸痛、肩痛、呼吸困难,消化不良,严重疲劳。如果出现这些症状，请去医院求诊。

(参考内容,详见 心脏年龄)

HEART_AGE 心脏年龄

心脏年龄数值表示了目标心脏的相对年龄,该值是根据他们的心率变异性(HRV)与普遍人群特征对比得出的。低HRV会增加死亡风险,也代表可能高于该用户的生理年龄的心脏年龄(如一个35岁用户的HRV特征表明其心脏是45岁)。该计算将考虑用户的生理年龄。要用这个数据建议目标对象生理年龄至少10岁。

要通过MSG_HEART_AGE接收数据给应用的Handler,首先通过TGDevice对象设定用户的生理年龄tgDevice.inputAge = 25（理所应当用25代替用户的实际年龄）。接着，只需将TGDevice连接到ThinkGear ECG/EKG传感器(CardioChip)，即用户能至少持续60次心跳在良好,干净的ThinkGear ECG/EKG传感器(CardioChip)获取信号(SENSOR_STATUS>= 200每60次心跳)。如果信号中断,SENSOR_STATUS会少于200,然后这个计算将重置并重新开始,再接收60次心跳的纯数据返回MSG_HEART_AGE值。

为达到最佳效果,用户在数据收集时应平静端坐。

举一个你的应用和用户可能会用到这个信息的例子：你的应用可以根据心脏年龄值反馈下述信息给客户：

青少年心脏: < 25 岁

你的心脏年龄是xx岁,比你的实际年龄大/少xx岁。你年轻的心脏年龄让你精力充沛,积极思考,它有助于你处理高要求的工作和运动。年轻的心脏同时也需要照顾。建议您避免熬夜深夜,做适量的运动,保持一个和平积极的心态。你还应多吃新鲜水果和蔬菜,减少高脂肪食物的摄入来保持你的状态。

年轻人心脏: 26 - 39 岁

你的心脏年龄是xx岁,比你的实际年龄大/少xx岁。你有一颗成熟的心脏。在紧张的工作环境里,请不要忘了保证足够的睡眠和锻炼,吃好,照顾好自己。

中年人心脏: 40 - 55 岁

你的心脏年龄是xx岁,比你的实际年龄大/少xx岁。请密切关注您的心脏健康并合理安排你的工作和生活以此来减轻心脏的负担。建议你多吃对心脏有益的食物,如鱼类、全麦、豆类、坚果、蔬菜、红酒和绿茶。你也应当保证一定的运动量来强化你的心脏。

有活力的老年人心脏: 56 - 70 岁

你的心脏年龄是xx岁,比你的实际年龄大/少xx岁。你的心脏功能正在向老年前进。建议您讲究生活纪律,避免变得过于兴奋或紧张。你也应该定期体检和进食更多对你的

心脏有益的食物,如鱼类、粗粮、豆类、坚果、蔬菜、红酒和绿茶。同样重要的是你也应当保证适当的运动量让你的心脏有效工作。

老年人心脏: >70 岁

你的心里年龄是xx岁,比你的实际年龄大/少xx岁。建议您定期去医院体检查,仔细按照医生的指示来预防或治疗心脏病。你也需要适量的运动,保持和平的心态。讲究生活纪律和健康饮食可以改善心血管系统的功能,防止心脏病。

（参考文献）

1. Res Sports Med. 2010 Oct; 18(4):263-9. Age and heart rate variability after soccer games. YuS, Katoh T, Makino H, Mimuno S, Sato S.

2. J Am Coll Cardiol. 1998 Mar 1; 31(3): 593-601. Twenty four hour time domain heart rate variability and heart rate: relations to age and gender over nine decades. Umetani K, Singer DH, McCraty R, Atkinson M.

3. Am J Cardiol. 2010 Apr 15; 105(8): 1181-5. Epub 2010. Relation of high heart rate variability to healthy longevity. Zulëqar U, Jurivich DA, Gao W, Singer DH.

4. Cardiovasc Electrophysiol. 2003 Aug; 14(8): 791-9. Circadian proële of cardiac autonomic nervous modulation in healthy subjects: differing effects of aging and gender on heart rate variability. Bonnemeier H, Richardt G, Potratz J, Wiegand UK, Brandes A, Kluge N, Katus HA.

5. Pacing Clin Electrophysiol. 1996 Nov; 19(11 Pt 2): 1863-6. Changes in heart rate variability with age. Reardon M, Malik M.

Personalization 个性化心电识别

该算法允许TGDevice尝试去识别已连接到ECG/EKG数据的用户。要使用它, 用户需要“培训”他们到TGDevice的ECG/EKG数据。然后,当TGDevice从用户读取心电图数据时,它可以试图确定具体读取的哪个培训用户(如果有的话)的数据。

使用个性化算法有两个步骤:

培训

记录用户ECG/EKG数据的第一部分是使用TGDevice类的EKGstartLongTraining(String userName)方法。如果用户能和ECG/EKG传感器设备保持良好干净的接触,消息MSG_EKG_TRAIN_STEP将会发送到你应用的Handler，告诉你目前在哪一步。完成

这两个步骤后, MSG_EKG_TRAINED会发到应用的Handler表明记录结束,接着Handler会使用EKGstopTraining()方法来停止。

connector.EKGstartLongTraining("NeuroSky");

检测

这部分是根据第一部分保存的数据来识别用户的。要识别用户,调用EKGstartDetection()方法。接着,如果用户能与ECG/EKG传感器硬件保持良好干净的连接,TGDevice会将消息MSG_EKG_IDENTIFIED发送到应用程序的Handler。返回值将是一个已注册过的用户名,或“佚名”。

connector.EKGstartDetection();

EKGPersonalizationEvent的Event Handler

EKGPersonalizationEvent 的事件处理器event handler 表示个性化算法是目前的状态。

当前有四种可能的状态:
1. MSG_EKG_IDENTIFIED 表明该算法确定的用户是谁。用户名会返回到dataMessage中。
2. MSG_EKG_TRAINED表明培训完成了最后步骤。 通常下一步会调用EKGstartLongTraining.
3. MSG_EKG_TRAIN_STEP 表明培训完成了一个步骤。培训完成的数量会返回到edataMessage中。
4. MSG_EKG_TRAIN_TOUCH 表明用户现在应该和ECG/EKG sensor 保持良好、干净的接触。

在 EKGPersonalizationEvent的事件处理器中做如下操作：

```
static void OnEKGPersonalizationEvent(object sender, EventArgs e) {
EKGPersonalizationEventArgs ekgArgs = (EKGPersonalizationEventArgs)(e);
int status = ekgArgs.statusMessage;

switch(status) {
case 268:
        string data = (string)(ekgArgs.dataMessage);
        Console.WriteLine("status = MSG_EKG_IDENTIFIED " + " and username = " + data);
        break;

case 269:
        Console.WriteLine("status = MSG_EKG_TRAINED");
        break;

case 270:
        int trainStep = (int)ekgArgs.dataMessage;
        Console.WriteLine("status = MSG_EKG_TRAIN_STEP " + " and training step = " + trainStep
+ ". Please remove fingers from sensors");
```

```
                break;


        case 271:
                Console.WriteLine("status = MSG_EKG_TRAIN_TOUCH. Please place fingers on sensors");
                break;
        }
}
```

要使用该event handler, 在你的主函数中添加如下代码：

```
Connector.EKGPersonalizationEvent += new EventHandler(OnEKGPersonalizationEvent);
```

重要：注意EKGPersonalizationEvent是一个静态事件处理程序。

RrInt 心电间隔

每当沿着用户的心电轴检测到R-peak（波峰）时，消息MSG_EKG_RRINT会被发送到Handler，用毫秒表示距离上次R-peak的R-R间隔。

恰当的应用设计

重要：要在应用程序正式发布之前,确保你的应用已考虑或处理了以下几点：

- 如果你的应用Handler收到了MSG_STATE_CHANGE消息或者除了STATE_CONNECTING和STATE_CONNECTED以外的任意消息, 就应该仔细把每一个可能发生的错误的处理信息通过UI 发送给客户。不在UI处理好这些错误，会导致用户体验很差。下面是一些例子说明：

——如果收到STATE_ERR_BT_OFF消息,应该提示用户打开蓝牙适配器,然后再重新尝试连接。

——如果收到STATE_ERR_NO_DEVICE消息, 应该提醒用户先根据他们ThinkGear硬件设备收到的指令，把ThinkGear硬件设备和Android设备的蓝牙进行配对。

——如果收到STATE_NOT_FOUND消息,应该提醒用户检查ThinkGear硬件设备是否和Android设备已成功配对 (和STATE_ERR_NO_DEVICE的情况相同)，比如查看他们ThinkGear硬件设备是否打开，是否在距离范围内，电池电量是否充足。

——查看 TGDevice 状态 获取更多信息。

- 确保你的应用程序总是在处理POOR_SIGNAL/SENSOR_STATUS数据类型。它是由ThinkGear设备输出,并且提供了传感器是否正常与用户接触的重要信息。如果它正在提示可能出了一些问题(即EEG设备不等于0或ECG/EKG设备不等于200)，那么应用程序应该通知用户应该正确穿戴好ThinkGear硬件设备，或者当POOR_SIGNAL/SENSOR_STATUS持续显示问题期间忽视其输出的数据值，可根据实际需求做出最符合你应用程序的判断。

- 为了做出让用户体验保持一致性、熟悉性、容易掌握和使用的跨平台设备,您的应用程序设计应该遵循NeuroSky的App Standards的准则和惯例。

故障排除

注意： 目前还没有已知问题，如果你遇到任何bug或问题，请访问 http://support.neurosky.com，或联系邮箱 support@neurosky.com.

- 如果你需要进一步的帮助，可以访问链接http://developer.neurosky.com查看是否有最新信息。

- 要联系NeuroSky技术支持，请访问http://support.neurosky.com，或者发送邮件给support@neurosky.com 。

- 关于开发者的论坛支持,请访问我们的社区论坛：

  http://www.linkedin.com/groups/NeuroSky-Brain-Computer-Interface-Technology-3572341

重要声明

SDK里的算法是专门用来提高个人健康养生意识的,并不能代替医疗护理。这些算法不用于诊断,治疗,治愈或预防任何疾病,开出任何药物,亦或是代替医疗设备的治疗。在某些情况下,该算法可能也会产生错误或不准确的结果。SDK文档中的算法和显示的数据只是针对特定用户使用的。NeuroSky算法公开可用，不承担其最终使用和显示内部代码的责任。

如果用户拥有心脏起搏器，该算法功能可能不好也可能显示准确的数据。为了收集最佳效果的EEG数据，需要用户平静端坐，规律呼吸，尽可能小幅动作。

警告和免责声明：

这些算法禁止任意形式的非法使用, 禁止应用于生命保障，安全设备或系统的组件，禁止应用于军事或核程序,禁止任意程序中的算法故障会导致人员伤亡。你使用软件开发工具包,算法和其他NEUROSKY的产品或服务是基于现状的，神念科技公司不做任何明示或 暗示的保证，授权或许可。包括有关适销性，知识产权(包括专利，版权或其他) 或特殊目的的适用性的保证。

神念电子科技概不负责任何意外，偶然，间接的伤害。包括但不限于利润或收入的损失，无论神念科技公司是否已被告知，此类损害均有可能发生。