EEG algorithm SDK for Windows: Development Guide

February 2, 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

About the Windows SDK

This document will guide you through the process of generating algorithm outputs from different NeuroSky Proprietary Mind Algorithms *using* **NeuroSky EEG Algorithm SDK for Windows** *with* EEG data collected by NeuroSky Biosensor System (e.g. TGAM module or MindWave Mobile Headset).

This development guide is intended for *Windows application developers* who are already familiar with standard Windows programming. If you are not already familiar with developing for Windows application.

Important: .

- Tested under Windows 7

EEG Algorithm SDK for Windows Contents

- EEG Algorithm SDK for Windows: Development Guide (this document)
- EEG Algorithm SDK Dynamic Linked Library (DLL): AlgoSdkDll.dll (for 32-bit) and AlgoSdkDll64.dll (for 64-bit)
- Readme: Readme file
- Algo SDK Sample project

Application development

Introduction

We recommend developers to use our COMM SDK for Windows in their application. Comm SDK reduces the complexity of managing EEG Algorithm SDK connections and handles data stream parsing. With the help of our SDKs, the application could be as simple as passing the data received and parsed by the Comm SDK to specific function call(s) at Algo SDK. Specific EEG algorithm index would then be returned accordingly.

Important: .

- NeuroSky Comm SDK can only communicate with one paired device at a time.

EEG Algorithm Sample Project

For collecting EEG data from NeuroSky Biosensor System (e.g. MindWave Mobile headset) with Windows, please refer to our COMM SDK for Windows document.

Algo SDK Sample is an sample Windows application using Communication (Comm) SDK to connect to NeuroSky Biosensor System (e.g. MindWave Mobile headset) and Algorithm (Algo) SDK for algorithmic computation for specific NeuroSky Mind algorithm, including Attention, Meditation, Appreciation, Mental Effort and Familiarity.

1. Connecting NeuroSky MindWave Mobile headset with Windows PC
2. On the Windows app development machine, double click the Algo SDK Sample Visual Studio Solution (**"Algo SDK Sample.sln"**) to launch the project with Visual Studio Express
3. Select **Build** —> **Rebuild Solution** to rebuild the "Algo SDK Sample" application
4. Run the sample app, selected algorithm indices will be output to the Text Field.

Important: .

- The sample project requires Microsoft Visual Studio Express 2015.
- The sample project only demonstrates how to iterate with the EEG Algo SDK.
- **Comm SDK** enclosed in the sample project is **version 1.1**. Please make sure you are using the latest stable Comm SDK version and make proper changes on sample project if needed.

API Documentation

The **EEG Algorithm SDK API Reference** in this section contains descriptions of the EEG Algorithm available APIs.

Return Codes

```
typedef enum _eNSK_ALGO_RET {
        NSK_ALGO_RET_SUCCESS,           /* Operation performed successfully */
        NSK_ALGO_RET_FAIL,              /* Operation performed with error */
        NSK_ALGO_RET_ALREADY_INITED,    /* Already inited */
        NSK_ALGO_RET_ALREADY_STARTED,   /* SDK has already started */
        NSK_ALGO_RET_ALREADY_STOPPED,   /* SDK has already been stopped */
        NSK_ALGO_RET_NOT_INITED,        /* SDK hasn't been inited */
        NSK_ALGO_RET_NO_MEM,            /* Not enough of memory */
        NSK_ALGO_RET_INVALID_PARAM,     /* Invalid input parameter */
        NSK_ALGO_RET_NOT_STARTED,       /* SDK hasn't been started yet */
        NSK_ALGO_RET_NOT_SUPPORTED,     /* Unsupported algorithm */
        NSK_ALGO_RET_NOT_SELECTED,      /* Unselected algorithm */
        NSK_ALGO_RET_NO_CALLBACK,       /* No callback registered */
} eNSK_ALGO_RET;
```

Data Type Declarations

EEG Data Types

```
typedef enum _eNSK_ALGO_DATA_TYPE {
    NSK_ALGO_DATA_TYPE_EEG = 0x01,   /* Raw EEG data */
    NSK_ALGO_DATA_TYPE_ATT,          /* Attention data */
    NSK_ALGO_DATA_TYPE_MED,          /* Meditation data */
    NSK_ALGO_DATA_TYPE_PQ,           /* Poor signal quality data */
    NSK_ALGO_DATA_TYPE_BULK_EEG,     /* Bulk of EEG data */
    NSK_ALGO_DATA_TYPE_MAX
} eNSK_ALGO_DATA_TYPE;
```

EEG Algorithm Data Types

```
typedef enum _eNSK_ALGO_TYPE {
    NSK_ALGO_TYPE_ATT   = 0x00000100,  /* Attention */
    NSK_ALGO_TYPE_MED   = 0x00000200,  /* Meditation */
    NSK_ALGO_TYPE_BLINK = 0x00000400,  /* Eye blink detection */
    NSK_ALGO_TYPE_BP    = 0x00004000   /* EEG Bandpower */
} eNSK_ALGO_TYPE;
```

EEG Algorithm SDK Callback Parameter Types

```
typedef enum _eNSK_ALGO_CB_TYPE {
    /* Register callback for Algo SDK state change */
    NSK_ALGO_CB_TYPE_STATE         = 0x01000000,
    /* Register callback for Algo signal quality level */
    NSK_ALGO_CB_TYPE_SIGNAL_LEVEL  = 0x02000000,
    /* Register callback for Algo analysis result */
    NSK_ALGO_CB_TYPE_ALGO          = 0x04000000
} eNSK_ALGO_CB_TYPE;
```

EEG Algorithm SDK States

```
typedef enum _eNSK_ALGO_STATE {
/* SDK state */
/*
 * Algo SDK is initialized (Reason code is omitted),
 * host application should never receive this state
 */
    NSK_ALGO_STATE_INITED                           = 0x0100,
/*
 * Algo SDK is performing analysis (i.e. NSK_ALGO_Start() is invoked
 */
    NSK_ALGO_STATE_RUNNING                          = 0x0200,
/*
 * Algo SDK is collecting baseline data.
 * NSK_ALGO_Start() is invoked and some algorithms are collecting baseline data.
 * When all baseline data collection are done, SDK state will change to NSK_ALGO_STATE_RUNNING
 */
    NSK_ALGO_STATE_COLLECTING_BASELINE_DATA         = 0x0300,
/*
 * Algo SDK stops data analysis/baseline collection.
 * State will only change to stop if previous state is NSK_ALGO_STATE_RUNNING
 * or NSK_ALGO_STATE_COLLECTING_BASELINE_DATA */
    NSK_ALGO_STATE_STOP                             = 0x0400,
/* Algo SDK pauses data analysis due to poor signal quality or paused by user.
 * State will only change to pause if previous state is NSK_ALGO_STATE_RUNNING */
    NSK_ALGO_STATE_PAUSE                            = 0x0500,
/* Algo SDK is uninitialized */
    NSK_ALGO_STATE_UNINTIED                         = 0x0600,
/* Algo SDK is analysing provided bulk data (i.e. NSK_ALGO_DataStream() is invoked with
NSK_ALGO_DATA_TYPE_BULK_EEG.
 * Note: SDK state will change to NSK_ALGO_STATE_STOP after analysing data */
    NSK_ALGO_STATE_ANALYSING_BULK_DATA              = 0x0800,
    NSK_ALGO_STATE_MASK                             = 0xFF00,

/* Reason for state change */
/*
 * RESERVED: SDK configuration changed (i.e. NSK_ALGO_Config() is invoked)
 */
    NSK_ALGO_REASON_CONFIG_CHANGED                  = 0x0001,
/*
 * RESERVED: Active user profile has been changed (i.e. NSK_ALGO_Profile()
 * is invoked and active user profile is affected)
 */
    NSK_ALGO_REASON_USER_PROFILE_CHANGED            = 0x0002,
/*
 * RESERVED: Callback registration has been changed (i.e. NSK_ALGO_RegisterCallback() is invoked)
 */
    NSK_ALGO_REASON_CB_CHANGED                      = 0x0003,
/*
 * Stopped/Paused by user (i.e. NSK_ALGO_Stop()/NSK_ALGO_Pause() is invoked)
 */
    NSK_ALGO_REASON_BY_USER                         = 0x0004,
/*
 * RESERVED: Active user baseline data expired
 */
    NSK_ALGO_REASON_BASELINE_EXPIRED                = 0x0005,
/*
 * RESERVED: There is no baseline data for current active user
 */
    NSK_ALGO_REASON_NO_BASELINE                     = 0x0006,
/*
 * SDK state changes due to signal quality changes.
 * e.g. NSK_ALGO_STATE_PAUSE + NSK_ALGO_REASON_SIGNAL_QUALITY
 * means SDK pauses data analysis due to poor signal quality
 * e.g. NSK_ALGO_STATE_RUNNING + NSK_ALGO_REASON_SIGNAL_QUALITY
 * means SDK resumes data analysis due to signal resuming from poor signal quality */
    NSK_ALGO_REASON_SIGNAL_QUALITY                  = 0x0007,
    NSK_ALGO_REASON_MASK                            = 0x00FF
} eNSK_ALGO_STATE;
```

Note: .

- Developer will always need to check with the SDK state and perform proper GUI handling

EEG Data Signal Qualities

```
typedef enum _eNSK_ALGO_SIGNAL_QUALITY {
        NSK_ALGO_SQ_GOOD = 0,       /* Good signal quality */
        NSK_ALGO_SQ_MEDIUM,         /* Medium signal quality */
        NSK_ALGO_SQ_POOR,           /* Poor signal quality */
        NSK_ALGO_SQ_NOT_DETECTED    /* Sensor off-head detected */
} eNSK_ALGO_SIGNAL_QUALITY;
```

Attention Index Data Type

```
/* Data type for Attention index */
typedef NS_ALGO_INDEX NS_ALGO_ATT_INDEX;
```

Meditation Index Data Type

```
/* Data type for Meditation index */
typedef NS_ALGO_INDEX NS_ALGO_MED_INDEX;
```

Eye Blink Strength

```
/* Data type for Eye Blink strength */
typedef NS_ALGO_INDEX NS_ALGO_EYE_BLINK_STRENGTH;
```

EEG Bandpower

```
/* EEG Bandpower in dB */
typedef struct _NS_ALGO_BP_INDEX {
    NS_ALGO_INDEX delta_power;
    NS_ALGO_INDEX theta_power;
    NS_ALGO_INDEX alpha_power;
    NS_ALGO_INDEX beta_power;
    NS_ALGO_INDEX gamma_power;
} NS_ALGO_BP_INDEX;
```

EEG Algorithm Index Group Data Type

```
/* Data type for group of EEG algorithm indices */
typedef struct _sNSK_ALGO_INDEX_GROUP {
    union {
        NS_ALGO_AP_INDEX ap_index;                      /* RESERVED */
        NS_ALGO_ME_INDEX me_index;                      /* RESERVED */
        NS_ALGO_ME2_INDEX me2_index;                    /* RESERVED */
        NS_ALGO_F_INDEX f_index;                        /* RESERVED */
        NS_ALGO_F2_INDEX f2_index;                      /* RESERVED */
        NS_ALGO_ATT_INDEX att_index;
        NS_ALGO_MED_INDEX med_index;
        NS_ALGO_EYE_BLINK_STRENGTH eye_blink_strength;
        NS_ALGO_CR_INDEX cr_index;                      /* RESERVED */
        NS_ALGO_AL_INDEX al_index;                      /* RESERVED */
        NS_ALGO_CP_INDEX cp_index;                      /* RESERVED */
        NS_ALGO_BP_INDEX bp_index;
    } group;
} sNSK_ALGO_INDEX_GROUP;
```

EEG Algorithm SDK Callback Algorithm Index Data Type

```
/* Data type for EEG algorithm index returned to host */
typedef struct _sNSK_ALGO_INDEX {
    eNSK_ALGO_TYPE type;           /* Reported Algo type */
    sNSK_ALGO_INDEX_GROUP value;   /* Reported Algo index, depending on the eNSK_ALGO_TYPE */
    NS_INT32 timeSpent;
} sNSK_ALGO_INDEX;
```

EEG Algorithm SDK Callback Parameter

```
typedef struct _sNSK_ALGO_CB_PARAM {
    /**
     * Different cbType should map to corresponding param
     * NSK_ALGO_CB_TYPE_STATE -> state
     * eNSK_ALGO_SIGNAL_QUALITY -> sq
     * NSK_ALGO_CB_TYPE_ALGO -> index
     */
    eNSK_ALGO_CB_TYPE cbType;
        union {
            /* SDK reported state */
            eNSK_ALGO_STATE state;
            /* SDK reported signal quality level */
            eNSK_ALGO_SIGNAL_QUALITY sq;
            /* SDK reported algorithm index */
            sNSK_ALGO_INDEX index;
        } param;
    NS_VOID *userData;
} sNSK_ALGO_CB_PARAM;
```

API Definitions

NSK_ALGO_Init

Initialize the Algo SDK with supported algorithm types.

```
/**
  * @brief    Required: Initialize the Algo SDK with supported algorithm types.
  *
  * @param    type      : Bit-wise OR the supported algorithm types
  * @param    dataPath  : An user data path to store user data
  * @retval   NSK_ALGO_RET_SUCCESS on operation success or else fail
  */
eNSK_ALGO_RET NSK_ALGO_Init (eNSK_ALGO_TYPE type, const NS_STR dataPath);
```

Example 1 - Single algorithm

```
eNSK_ALGO_RET ret = NSK_ALGO_Init ((eNSK_ALGO_TYPE)NSK_ALGO_TYPE_ATT,
"C:\\Users\\NeuroSky\\Documents");
ASSERT(ret==NSK_ALGO_RET_SUCCESS);
```

Example 2 - Multiple algorithms

```
eNSK_ALGO_RET ret = NSK_ALGO_Init ((eNSK_ALGO_TYPE)(NSK_ALGO_TYPE_ATT | NSK_ALGO_TYPE_MED),
"C:\\Users\\NeuroSky\\Documents");
ASSERT(ret==NSK_ALGO_RET_SUCCESS);
```

NSK_ALGO_Uninit

Uninitialize the Algo SDK

```
/**
  * @brief    Required: Uninitialize the Algo SDK
  *                 Note: if SDK state is NSK_ALGO_STATE_RUNNING, then SDK state will change to
NSK_ALGO_STATE_STOP with reason NSK_ALGO_REASON_BY_USER before SDK is uninitialized
  * @retval    NSK_ALGO_RET_SUCCESS on operation success or else fail
  */
eNSK_ALGO_RET NSK_ALGO_Uninit (NS_VOID);
```

NSK_ALGO_RegisterCallback

Register the callback for the host application to get notification on algorithm result, SDK state change and data signal quality.

```
/**
  * @brief    Required: Register the callback for the host application to get notification on
algorithm result, SDK state change and data signal quality
  *                 Note: if SDK state is NSK_ALGO_STATE_RUNNING, then SDK state will be changed to
NSK_ALGO_STATE_STOP with reason NSK_ALGO_REASON_CB_CHANGED before updating exiting callback (new
callback will be invoked instead)
  *                 Note: This function can be invoked before NSK_ALGO_Init() in order to collect
the NSK_ALGO_STATE_INITED SDK state changed event.
  *
  * @param    cbFunc      : Function pointer of the callback
  * @param    userData    : User data to be pass in the callback parameter
  * @retval   NSK_ALGO_RET_SUCCESS on operation success or else fail
  */
eNSK_ALGO_RET NSK_ALGO_RegisterCallback (NskAlgo_Callback cbFunc, NS_VOID *userData);
```

Example

```
// Implementation of the callback function
static NS_VOID nskSdkFuncCB(sNSK_ALGO_CB_PARAM param) {
        if (param.cbType == NSK_ALGO_CB_TYPE_STATE) {
                // Update the GUI components and application state accordingly
        } else if (param.cbType == NSK_ALGO_CB_TYPE_SIGNAL_LEVEL) {
                switch (param.param.sq) {
                        case NSK_ALGO_SQ_GOOD:
                                // Good signal quality
                                break;
                        case NSK_ALGO_SQ_MEDIUM:
                                // Medium signal quality
                                break;
                        case NSK_ALGO_SQ_POOR:
                                // Poor signal quality
                                break;
                        case NSK_ALGO_SQ_NOT_DETECTED:
                                // Sensor not detected
                                break;
                }
        } else if (param.cbType == NSK_ALGO_CB_TYPE_ALGO) {
                if (param.param.index.type == NSK_ALGO_TYPE_ATT) {
                        // Attention index
                } else if (param.param.index.type == NSK_ALGO_TYPE_MED) {
                        // Meditation index
                } else if (param.param.index.type == NSK_ALGO_TYPE_BLINK) {
                        // Eye blink detected
                }
        }
}

// Register the callback function
void main () {
        // Register the EEG Algo callback function first such no SDK state will be missed
        eNSK_ALGO_RET ret = NSK_ALGO_RegisterCallback(&nskSdkFuncCB, NULL);
        ASSERT(ret==NSK_ALGO_RET_SUCCESS);
        // Initialize the EEG Algo SDK
        ret = NSK_ALGO_Init((eNSK_ALGO_TYPE)NSK_ALGO_TYPE_ATT, "C:\\Users\\NeuroSky\\Documents");
        ASSERT(ret==NSK_ALGO_RET_SUCCESS);

        // More application logic below
        :
        :
}
```

NSK_ALGO_SdkVersion

Get the Algo SDK version.

```
/**
  * @brief    Optional: Get the Algo SDK version
  *                 Format: M.m.p, where M is major version, m is minor version and p is patch
version
  *
  * @retval    Null terminated string
  */
NS_STR NSK_ALGO_SdkVersion (NS_VOID);
```

Example

```
eNSK_ALGO_RET ret = NSK_ALGO_Init((eNSK_ALGO_TYPE)NSK_ALGO_TYPE_ATT,
"C:\\Users\\NeuroSky\\Documents");
NS_STR sdkVersionStr = NS_NULL;
ASSERT(ret==NSK_ALGO_RET_SUCCESS);

sdkVersionStr = NSK_ALGO_SdkVersion();
if (sdkVersionStr != NS_NULL) {
        printf("EEG Algo SDK ver.: %s\n", sdkVersionStr);
}
```

NSK_ALGO_AlgoVersion

Get the Algo SDK version.

```
/**
  * @brief    Optional: Get the Algo SDK version
  *                     Format: M.m.p, where M is major version, m is minor version and p is patch
version
  *
  * @param    type    : Specify the supported Algo type version to be queried
  * @retval   Null terminated string
  */
NS_STR NSK_ALGO_AlgoVersion (eNSK_ALGO_TYPE type);
```

Example

```
eNSK_ALGO_RET ret = NSK_ALGO_Init((eNSK_ALGO_TYPE)NSK_ALGO_TYPE_ATT,
"C:\\Users\\NeuroSky\Documents");
NS_STR apVersionStr = NS_NULL;
ASSERT(ret==NSK_ALGO_RET_SUCCESS);

attVersionStr = NSK_ALGO_AlgoVersion((eNSK_ALGO_TYPE)NSK_ALGO_TYPE_ATT);
if (attVersionStr != NS_NULL) {
        printf("Attention Algo ver.: %s\n", attVersionStr);
}
```

NSK_ALGO_Start

Start processing data from NSK_ALGO_DataStream() call.

```
/**
  * @brief    Required: Start processing data from NSK_ALGO_DataStream() call
  *                 Note: SDK state will changed to NSK_ALGO_STATE_RUNNING when previous state is
NSK_ALGO_STATE_STOP / NSK_ALGO_STATE_PAUSE / NSK_ALGO_STATE_INITED
  *
  * @param    bBaseline: Always be NS_FALSE [RESERVED]
  * @retval   NSK_ALGO_RET_SUCCESS on operation success or else fail
  */
eNSK_ALGO_RET NSK_ALGO_Start (NS_BOOL bBaseline);
```

Note: .

- SDK state will only change to **RUNNING** by invoking **NSK_ALGO_Start()**

NSK_ALGO_Pause

Pause processing/collecting data.

```
/**
  * @brief    Required: Pause processing/collecting data
  *                Note: SDK state will changed to NSK_ALGO_STATE_PAUSE with reason
NSK_ALGO_REASON_BY_USER
  *
  * @retval   NSK_ALGO_RET_SUCCESS on operation success or else fail
  */
eNSK_ALGO_RET NSK_ALGO_Pause (NS_VOID);
```

Note: .

- SDK state will change to **PAUSE**
- No algorithm index callback will not be invoked unless **NSK_ALGO_Start()** function is invoked again
- When SDK state is ANALYSING BULK DATA, then **NSK_ALGO_Pause()** will always return *NSK_ALGO_RET_FAIL* (i.e. no effect)

NSK_ALGO_Stop

Stop processing/collecting data.

```
/**
  * @brief    Required: Stop processing data.
  *                Note: SDK state will changed to NSK_ALGO_STATE_STOP with reason
NSK_ALGO_REASON_BY_USER
  *
  * @retval   NSK_ALGO_RET_SUCCESS on operation success or else fail
  */
eNSK_ALGO_RET NSK_ALGO_Stop (NS_VOID);
```

Note: .

- SDK state will change to **STOP**
- No algorithm index callback will be invoked unless **NSK_ALGO_Start** method is invoked again
- **NSK_ALGO_Stop** requires recollection of baseline data once restart (Exception for Attention and Meditation) while **NSK_ALGO_Pause** doesn't.

NSK_ALGO_DataStream

EEG data stream input from NeuroSky Biosensor System (e.g. TGAM or MindWave Mobile headset).

```
/**
  * @brief    Required: EEG data stream input from NeuroSky Biosensor System (e.g. TGAM or MindWave
Mobile headset)
  *              When type = NSK_ALGO_DATA_TYPE_PQ, dataLength = 1
  *              When type = NSK_ALGO_DATA_TYPE_EEG, dataLength = 512 (i.e. 1 second EEG raw data)
  *              When type = NSK_ALGO_DATA_TYPE_ATT, dataLength = 1
  *              When type = NSK_ALGO_DATA_TYPE_MED, dataLength = 1
  *              When type = NSK_ALGO_DATA_TYPE_BULK_EEG, dataLength = N*512 (i.e. N continous
seconds of EEG raw data)
  *              Note 1: In case of type = NSK_ALGO_DATA_TYPE_BULK_EEG, caller should NOT release
the data buffer until SDK state changes back to NSK_ALGO_STATE_STOP
  *              Note 2: In case of type = NSK_ALGO_DATA_TYPE_BULK_EEG, the first 5 seconds of data
will be used as baseline data
  *
  * @param    type       : Data type
  * @param    data       : Data stream
  * @param    dataLenght : Size of the data stream
  * @retval   NSK_ALGO_RET_SUCCESS on operation success or else fail
  */
eNSK_ALGO_RET NSK_ALGO_DataStream (eNSK_ALGO_DATA_TYPE type, NS_INT16 *data, NS_INT dataLenght);
```

Note: .

- For the data format from NeuroSky Biosensor System, please refer to TGAM Communication Protocol

Important: .

- There are different data output giving out from NeuroSky Biosensor System.
- EEG Algo SDK handles only the following **4** data output for now. They are:
  - Poor Signal Quality
  - EEG Raw Data
  - Attention
  - Meditation

Handling realtime EEG data

```
static DWORD ThreadReadPacket(LPVOID lpdwThreadParam) {
    int rawCount = 0;
    short rawData[512] = { 0 };
    while (true) {
        /* Read a single Packet from the connection */
        packetsRead = TG_ReadPackets(connectionId, 1);

        /* If TG_ReadPackets() was able to read a Packet of data... */

        if (packetsRead == 1) {
            /* If the Packet contained a new raw wave value... */
            if (TG_GetValueStatus(connectionId, TG_DATA_RAW) != 0) {
                /* Get and print out the new raw value */
                rawData[rawCount++] = (short)TG_GetValue(connectionId, TG_DATA_RAW);
                if (rawCount == 512) {
                    /* Send one second (512 samples) of EEG raw data to the SDK */
                    NSK_ALGO_DataStream(NSK_ALGO_DATA_TYPE_EEG, rawData, rawCount);
                    rawCount = 0;
                }
            }
            if (TG_GetValueStatus(connectionId, TG_DATA_POOR_SIGNAL) != 0) {
                short pq = (short)TG_GetValue(connectionId, TG_DATA_POOR_SIGNAL);
                NSK_ALGO_DataStream(NSK_ALGO_DATA_TYPE_PQ, &pq, 1);
            }
            if (TG_GetValueStatus(connectionId, TG_DATA_ATTENTION) != 0) {
                short att = (short)TG_GetValue(connectionId, TG_DATA_ATTENTION);
                NSK_ALGO_DataStream(NSK_ALGO_DATA_TYPE_ATT, &att, 1);
            }
            if (TG_GetValueStatus(connectionId, TG_DATA_MEDITATION) != 0) {
                short med = (short)TG_GetValue(connectionId, TG_DATA_MEDITATION);
                NSK_ALGO_DataStream(NSK_ALGO_DATA_TYPE_MED, &med, 1);
            }
        }
    }
}
```

Applications

Application of Attention Algorithm

- Initialize the EEG Algo SDK with selected EEG algorithm - Attention Algorithm by invoking **NSK_ALGO_Init(NSK_ALGO_TYPE_ATT)** function
- Starting EEG data analysis by invoking **NSK_ALGO_Start()** method
- Attention index will be returned every 1 second when Algo SDK state is **RUNNING**
- Attention index ranges from **0 to 100**. The higher the index, the higher the attention level

Note: .

- Attention has a fixed output interval of 1 second, i.e. one new Attention index every second

Application of Meditation Algorithm

- Initialize the EEG Algo SDK with selected EEG algorithm - Meditation Algorithm by invoking **NSK_ALGO_Init(NSK_ALGO_TYPE_MED)** function
- Starting EEG data analysis by invoking **NSK_ALGO_Start()** function
- Meditation index will be returned every 1 second when Algo SDK state is **RUNNING**
- Meditation index ranges from **0 to 100**. The higher the index, the higher the meditation level

Note: .

- Meditation has a fixed output interval of 1 second, i.e. one new Meditation index every second

Application of EEG Bandpower Algorithm

- Initialize the EEG Algo SDK with selected EEG algorithm - EEG Bandpower Algorithm by invoking **NSK_ALGO_Init(NSK_ALGO_TYPE_BP)** function
- Starting EEG data analysis by invoking **NSK_ALGO_Start()** function
- EEG bandpowers (delta, theta, alpha, beta and gamma in dB) index will be returned every 1 second when Algo SDK state is **RUNNING**

Application of Eye Blink Detection

- Initialize the EEG Algo SDK with selected EEG algorithm - Eye Blink Detection by invoking **NSK_ALGO_Init(NSK_ALGO_TYPE_BLINK)** function
- Starting EEG data analysis by invoking **NSK_ALGO_Start()** function
- Eye blink strength will be returned once eye blink is detected when Algo SDK state is **RUNNING**

Note: .

- No baseline data collection will be needed

SDK Operations

Pause and Resume

- Assuming SDK is in **RUNNING** state
- Pausing EEG algorithm data analysis by invoking **NSK_ALGO_Pause()** function
- Resuming EEG algorithm data analysis by invoking **NSK_ALGO_Start()** function

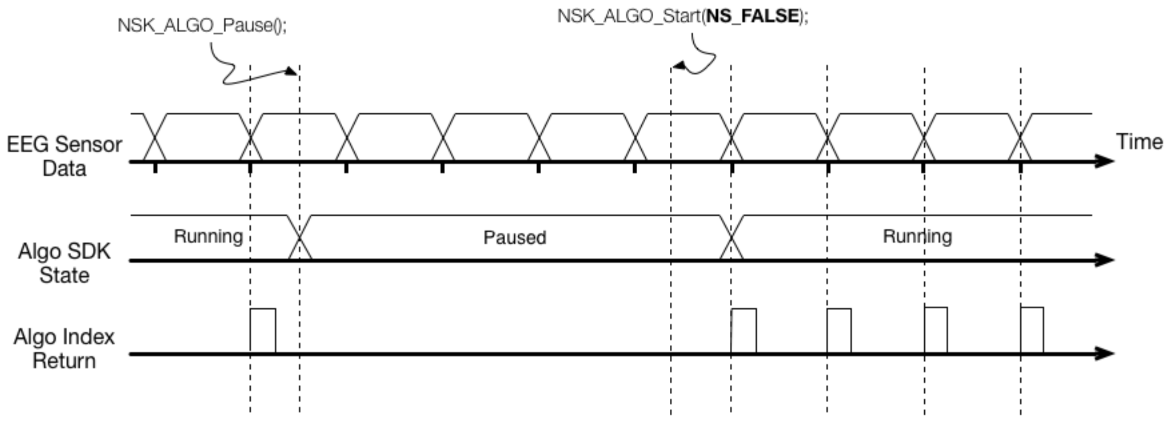

Figure 3.1: Time diagram on Pause/Resume SDK

Note: .

- There will be no effect on **NSK_ALGO_Pause()** when previous SDK state is not **RUNNING**
- When SDK state is ANALYSING BULK DATA, then **NSK_ALGO_Pause()** will always return *NSK_ALGO_RET_FAIL* (i.e. no effect)

Stop and Start

- Assuming SDK is in **RUNNING** state
- Stopping EEG algorithm data analysis by invoking **NSK_ALGO_Stop()** function
- Restart EEG algorithm data analysis by invoking **NSK_ALGO_Start()** function

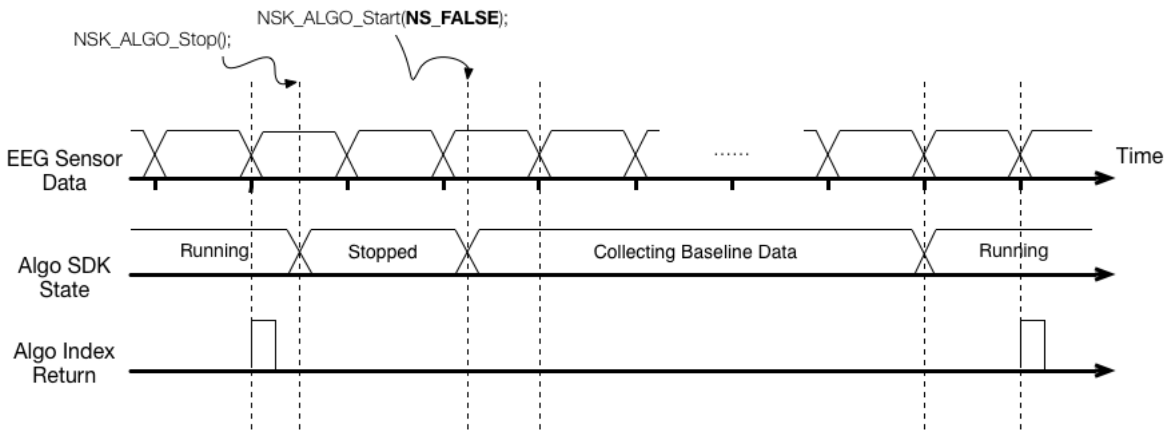

Figure 3.2: Time diagram on Stop/Start SDK

Note: .

- There will be no effect on **NSK_ALGO_Stop()** when previous SDK state is not **RUNNING**

Frequently Asked Questions