

---

# EEG algorithm SDK for Android: Development Guide

February 2, 2016

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.

**NO WARRANTIES: THE NEUROSKY PRODUCT FAMILIES AND RELATED DOCUMENTATION IS PROVIDED "AS IS" WITHOUT ANY EXPRESS OR IMPLIED WARRANTY OF ANY KIND INCLUDING WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT OF INTELLECTUAL PROPERTY, INCLUDING PATENTS, COPYRIGHTS OR OTHERWISE, OR FITNESS FOR ANY PARTICULAR PURPOSE. IN NO EVENT SHALL NEUROSKY OR ITS SUPPLIERS BE LIABLE FOR ANY DAMAGES WHATSOEVER (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF PROFITS, BUSINESS INTERRUPTION, COST OF REPLACEMENT GOODS OR LOSS OF OR DAMAGE TO INFORMATION) ARISING OUT OF THE USE OF OR INABILITY TO USE THE NEUROSKY PRODUCTS OR DOCUMENTATION PROVIDED, EVEN IF NEUROSKY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. , SOME OF THE ABOVE LIMITATIONS MAY NOT APPLY TO YOU BECAUSE SOME JURISDICTIONS PROHIBIT THE EXCLUSION OR LIMITATION OF LIABILITY FOR CONSEQUENTIAL OR INCIDENTAL DAMAGES.**

**USAGE OF THE NEUROSKY PRODUCTS IS SUBJECT OF AN END-USER LICENSE AGREEMENT.**

**“Made for iPod,” “Made for iPhone,” and “Made for iPad” mean that an electronic accessory has been designed to connect specifically to iPod, iPhone, or iPad, respectively, and has been certified by the developer to meet Apple performance standards. Apple is not responsible for the operation of this device or its compliance with safety and regulatory standards. Please note that the use of this accessory with iPod, iPhone, or iPad may affect wireless performance.**

# Contents

<b>About the Android SDK</b>	<b>4</b>
EEG Algorithm SDK for Android Contents	4
Application development	4
Introduction	4
EEG Algorithm Sample Project	5
<b>API Documentation</b>	<b>6</b>
Data Types	6
SDK Listener Methods	8
setOnStateChangeListener	8
setOnAttAlgoIndexListener	8
setOnMedAlgoIndexListener	9
setOnBPAlgoIndexListener	9
setOnEyeBlinkDetectionListener	10
NskAlgoSignalQualityListener	10
SDK Utility Methods	11
NskAlgoInit	11
NskAlgoUninit	11
NskAlgoAlgoVersion	12
NskAlgoSdkVersion	12
NskAlgoDataStream	13
NskAlgoStart	14
NskAlgoPause	14
NskAlgoStop	15
<b>Applications</b>	<b>16</b>
Application of Attention Algorithm	16
Application of Meditation Algorithm	16
Application of EEG Bandpower Algorithm	16
Application of Eye Blink Detection	17
SDK Operations	17
Pause and Resume	17
Stop and Start	18
Customize Algorithm Output Interval	18
<b>Frequently Asked Questions</b>	<b>20</b>

# About the Android 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 Android** *with* EEG data collected by NeuroSky Biosensor System (e.g. TGAM module or MindWave Mobile Headset).

This development guide is intended for *Android application developers* who are already familiar with standard Android development using **Android Studio/Eclipse**. If you are not already familiar with developing for Android, please first visit Android's developer web site for instruction and tools to develop Android apps.

**Important:** .

- Requires minimum Android API version 16 or later

## EEG Algorithm SDK for Android Contents

- EEG Algorithm SDK for Android: Development Guide (this document)
- EEG Algorithm SDK library: **libs/**
  - `<CPUARCH>/libNskAlgo.so` (compatible CPU architectures: *arm64-v8a, armeabi, armeabi-v7a, mips, mips64, x86, x86\_64*)
- EEG Algorithm SDK Java interface: **jar/**
  - `NskAlgoSdk.jar`
- Algo SDK Sample project

**Note:** .

- The sample project was created with **Android Studio**. However, developers can still use any familiar IDE to start his own Android project
- The minimum requirements for the sample project is **API 19**

## Application development

### Introduction

We recommend developers to use our [COMM SDK for Android](#) in their application. Comm SDK reduces the complexity of managing EEG Algorithm SDK connections and handles data stream pars-

ing. 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 Android device, please refer to our [COMM SDK for Android](#) document.

**Algo SDK Sample** is an sample Android 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 (with secondary algorithm) and Familiarity (with secondary algorithm).

1. Pairing NeuroSky MindWave Mobile Headset with Android device
2. Import the Algo SDK Sample project (gradle build) with Android Studio
3. Select Build -> Rebuild Project to build the "app" and install the "Algo SDK Sample" app by Run Run "app"

**Important: .**

- The sample project compiles with **Android Studio**. However, the sample code inside still compiles with any other Android application development IDEs (e.g. **Eclipse**)
- The sample project only demonstrates how to iterate with the EEG Algo SDK.
- **Comm SDK** enclosed in the sample project is **version 1.0.4**. 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 classes and protocols available in the EEG Algorithm Android API.

## Data Types

See the **NskAlgoSdk.jar** in the SDK package

```

/* EEG data signal quality definitions */
public enum NskAlgoSignalQuality {
    NSK_ALGO_SQ_GOOD          (0), /* Good signal quality */
    NSK_ALGO_SQ_MEDIUM        (1), /* Medium signal quality */
    NSK_ALGO_SQ_POOR          (2), /* Poor signal quality */
    NSK_ALGO_SQ_NOT_DETECTED  (3); /* No signal detected. It probably is caused by bad sensor
contact */
}

/* SDK state definitions */
public enum NskAlgoState {
    /* 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. */
    NSK_ALGO_STATE_RUNNING         (0x0200),
    /*
        Algo SDK is collecting baseline data (Reason code is omitted).

        When baseline data collection is done, SDK state should change
        to NSK_ALGO_STATE_RUNNING and start data analysis
    */
    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 (Reason code is omitted) */
    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 */
NSK_ALGO_REASON_CONFIG_CHANGED          (0x0001),
/* RESERVED */
NSK_ALGO_REASON_USER_PROFILE_CHANGED    (0x0002),
/* RESERVED */
NSK_ALGO_REASON_CB_CHANGED              (0x0003),
/* Stopped/Paused by user (i.e. NskAlgoStop()/NskAlgoPause() is invoked) */
NSK_ALGO_REASON_BY_USER                 (0x0004),
/* RESERVED */
NSK_ALGO_REASON_BASELINE_EXPIRED        (0x0005),
/* RESERVED */
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);
}

/* EEG algorithm type definitions */
public enum NskAlgoType {
    NSK_ALGO_TYPE_INVALID      (0x0000),
    NSK_ALGO_TYPE_ATT          (0x0100),    /* Attention */
    NSK_ALGO_TYPE_MED          (0x0200),    /* Meditation */
    NSK_ALGO_TYPE_BLINK        (0x0400),    /* Eye blink detection */
    NSK_ALGO_TYPE_BP           (0x4000);    /* EEG Bandpower */
}

/* Incoming EEG data type definitions (data from COMM SDK or recorded EEG data) */
public enum NskAlgoDataType {
    NSK_ALGO_DATA_TYPE_EEG      (0x01),    /* Raw EEG data */
    NSK_ALGO_DATA_TYPE_ATT      (0x02),    /* Attention data */
    NSK_ALGO_DATA_TYPE_MED      (0x03),    /* Meditation data */
    NSK_ALGO_DATA_TYPE_PQ       (0x04),    /* Poor signal quality data */
    NSK_ALGO_DATA_TYPE_BULK_EEG (0x05),    /* Bulk of EEG data */
    NSK_ALGO_DATA_TYPE_MAX      (0x06);
}

```

## SDK Listener Methods

See the **NskAlgoSdk.jar** in the SDK package.

### setOnStateChangeListener

EEG Algo SDK state change notification listener method

```
// Required
public void setOnStateChangeListener(NskAlgoSdk.OnStateChangeListener listener);
```

**Note:** .

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

#### Example

```
NskAlgoSdk nskAlgoSdk = new NskAlgoSdk();

nskAlgoSdk.setOnStateChangeListener(new NskAlgoSdk.OnStateChangeListener() {
    @Override
    public void onStateChange(int state, int reason) {
        Log.i(TAG, "On State Change: " + state + " [reason: " + reason + "]");
        runOnUiThread(new Runnable() {
            @Override
            public void run() {
                // change UI elements here
            }
        });
    }
});
```

### setOnAttAlgoIndexListener

Attention Algorithm index notification listener method

```
// Optional
public void setOnAttAlgoIndexListener(OnAttAlgoIndexListener listener);
```

**Note:** .

- Attention algorithm has a fixed output interval of 1 second
- Attention algorithm index ranges from 0 to 100 where higher the attention index, higher the attention level

#### Example

```
NskAlgoSdk nskAlgoSdk = new NskAlgoSdk();

nskAlgoSdk.setOnAttAlgoIndexListener(new NskAlgoSdk.OnAttAlgoIndexListener() {
    @Override
    public void onAttAlgoIndex(int value) {
```



```
Log.i(TAG, "NskAlgoAttAlgoIndexListener: Attention:" + value);
runOnUiThread(new Runnable() {
    @Override
    public void run() {
        // change UI elements here
    }
});
}
```

### setOnMedAlgoIndexListener

Meditation Algorithm index notification listener method

```
// Optional
public void setOnMedAlgoIndexListener(OnMedAlgoIndexListener listener);
```

#### Note: .

- Meditation algorithm has a fixed output interval of 1 second
- Meditation algorithm index ranges from 0 to 100 where higher the meditation index, higher the meditation level

#### Example

```
NskAlgoSdk nsKAlgoSdk = new NskAlgoSdk();

nsKAlgoSdk.setOnMedAlgoIndexListener(new NskAlgoSdk.OnMedAlgoIndexListener() {
    @Override
    public void onMedAlgoIndex(int value) {
        Log.i(TAG, "NskAlgoMedAlgoIndexListener: Meditation:" + value);
        runOnUiThread(new Runnable() {
            @Override
            public void run() {
                // change UI elements here
            }
        });
    }
});
```

### setOnBPAlgoIndexListener

EEG Bandpower Algorithm index notification listener method

```
// Optional
public void setOnBPAlgoIndexListener(OnBPAlgoIndexListener listener);
```

#### Note: .

- EEG bandpower (in dB) algorithm has a fixed output interval of 1 second

#### Example

```
NskAlgoSdk nskAlgoSdk = new NskAlgoSdk();

nskAlgoSdk.setOnBPAlgoIndexListener(new NskAlgoSdk.OnBPAlgoIndexListener() {
    @Override
    public void onBPAlgoIndex(float delta, float theta, float alpha, float beta, float gamma) {
        Log.i(TAG, "NskAlgoBPAlgoIndexListener: BP: D[ " + delta + " dB] T[ " + theta + " dB] A[ " +
alpha + " dB] B[ " + beta + " dB] G[ " + gamma + " ]");
        runOnUiThread(new Runnable() {
            @Override
            public void run() {
                // change UI elements here
            }
        });
    }
});
```

### setOnEyeBlinkDetectionListener

Eye blink detection notification listener method

```
// Optional
public void setOnEyeBlinkDetectionListener(OnEyeBlinkDetectionListener listener);
```

#### Note: .

- No baseline data collection will be needed
- Eye blink strength will be returned once eye blink is detected when Algo SDK state is **RUNNING**

#### Example

```
NskAlgoSdk nskAlgoSdk = new NskAlgoSdk();

nskAlgoSdk.setOnEyeBlinkDetectionListener(new NskAlgoSdk.OnEyeBlinkDetectionListener() {
    @Override
    public void onEyeBlinkDetection(int value) {
        Log.i(TAG, "NskAlgoEyeBlinkDetectionListener: Eye blink detected");
        runOnUiThread(new Runnable() {
            @Override
            public void run() {
                // change UI elements here
            }
        });
    }
});
```

### NskAlgoSignalQualityListener

EEG data signal quality notification listener method

```
// Optional
public void setOnSignalQualityListener(OnSignalQualityListener listener);
```

**Note:** .

- Signal Quality was measured and reported at a fixed output interval of 1 second
- SDK state will be changed from **RUNNING** to **PAUSE** when signal quality is poor or sensor off-head is detected. It would return to its previous state (e.g. **RUNNING**) when the signal quality returns to normal

**Example**

```
NskAlgoSdk nskAlgoSdk = new NskAlgoSdk();

nskAlgoSdk.setOnSignalQualityListener(new NskAlgoSdk.OnSignalQualityListener() {
    @Override
    public void onSignalQuality(int level) {
        Log.i(TAG, "On Signal Quality: " + level);
        runOnUiThread(new Runnable() {
            @Override
            public void run() {
                // change UI elements here
            }
        });
    }
});
```

## SDK Utility Methods

### NskAlgoInit

Initialize the Algo SDK with supported algorithm type(s).

```
/**
 * @brief      Required: Initialize the Algo SDK with supported algorithm types.
 *
 * @param      type      : Algorithm type(s) (see NskAlgoType)
 * @param      dataPath  : An user data path to store user data
 * @retval     Return 0 on operation success or else fail
 */
int NskAlgoInit(int algoTypes, String dataPath);
```

**Example 1 - Single algorithm**

```
int ret = NskAlgoInit (NSK_ALGO_TYPE_ATT, "/User/Documents");
ASSERT(ret==0);
```

**Example 2 - Multiple algorithms**

```
int ret = NskAlgoInit (NSK_ALGO_TYPE_ATT+ NSK_ALGO_TYPE_MED, "/User/Documents");
ASSERT(ret==0);
```

### NskAlgoUninit

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      Return 0 on operation success or else fail
 */
int NskAlgoUninit ();
```

### NskAlgoAlgoVersion

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
 */
String NskAlgoAlgoVersion (int type);
```

#### Example

```
int ret = NskAlgoInit(NSK_ALGO_TYPE_AP, "/User/Documents");
String apVersionStr = NS_NULL;
ASSERT(ret==0);

apVersionStr = NskAlgoAlgoVersion(NSK_ALGO_TYPE_AP);
if (apVersionStr != NS_NULL) {
    Log.d(TAG, "Appreciation Algo ver.: %s\n", apVersionStr);
}
```

### NskAlgoSdkVersion

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
 */
String NskAlgoSdkVersion ();
```

#### Example

```
int ret = NskAlgoInit(NSK_ALGO_TYPE_AP, "/User/Documents");
String sdkVersionStr = null;
ASSERT(ret==0);

sdkVersionStr = NskAlgoSdkVersion();
if (sdkVersionStr != NS_NULL) {
    Log.d(TAG, "EEG Algo ver.: %s\n", sdkVersionStr);
}
```

## NskAlgoDataStream

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 continuous
 * 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 array
 * @param dataLength : Size of the data stream
 * @retval Return 0 on operation success or else fail
 */
int NskAlgoDataStream (int type, short data[], int dataLength);
```

### 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

### Example 1 - Handling realtime EEG data

```
public void onDataReceived(int datatype, int data, Object obj) {
    // You can handle the received data here
    // You can feed the raw data to algo sdk here if necessary.
    //Log.i(TAG, "onDataReceived");
    switch (datatype) {
        case MindDataType.CODE_ATTENTION:
            short attValue[] = {(short) data};
            NskAlgoDataStream(NskAlgoDataType.NSK_ALGO_DATA_TYPE_ATT.value, attValue, 1);
            break;
        case MindDataType.CODE_MEDITATION:
```

```

        short medValue[] = {(short) data};
        NskAlgoDataStream(NskAlgoDataType.NSK_ALGO_DATA_TYPE_MED.value, medValue, 1);
        break;
    case MindDataType.CODE_POOR_SIGNAL:
        short pqValue[] = {(short) data};
        NskAlgoDataStream(NskAlgoDataType.NSK_ALGO_DATA_TYPE_PQ.value, pqValue, 1);
        break;
    case MindDataType.CODE_RAW:
        raw_data[raw_data_index++] = (short) data;
        if (raw_data_index == 512) {
            NskAlgoDataStream(NskAlgoDataType.NSK_ALGO_DATA_TYPE_EEG.value, raw_data,
raw_data_index);
            raw_data_index = 0;
        }
        break;
    default:
        break;
}
}

```

## NskAlgoStart

Start processing data from NskAlgoDataStream() call.

```

/**
 * @brief      Required: Start processing data from NskAlgoDataStream() 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 false [RESERVED]
 * @retval     Return 0 on operation success or else fail
 */
int NskAlgoStart (boolean bBaseline);

```

### Note: .

- SDK state will only change to **RUNNING** by invoking **NSK\_ALGO\_Start()**

## NskAlgoPause

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     Return 0 on operation success or else fail
 */
int NskAlgoPause ();

```

**Note: .**

- SDK state will change to **PAUSE**
- No algorithm index callback will not be invoked unless **NskAlgoStart()** function is invoked again

## NskAlgoStop

Stop processing/collecting data.

```
/**
 * @brief      Required: Stop processing/collecting data.
 *             Note: SDK state will changed to NSK_ALGO_STATE_STOP with reason
 *             NSK_ALGO_REASON_BY_USER
 *
 * @retval     Return 0 on operation success or else fail
 */
int NskAlgoStop ();
```

**Note: .**

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

# Applications

---

## Application of Attention Algorithm

- Selecting Attention Algorithm by invoking **NskAlgoInit()** method
- Starting EEG data analysis by invoking **NskAlgoStart()** 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

- Selecting Meditation Algorithm by invoking **NskAlgoInit()** method
- Starting EEG data analysis by invoking **NskAlgoStart()** method
- 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

- Selecting Meditation Algorithm by invoking **NskAlgoInit()** method
- Starting EEG data analysis by invoking **NskAlgoStart()** method
- EEG bandpowers (in dB) index will be returned every 1 second when Algo SDK state is **RUNNING**

**Note:** .

- EEG Bandpower algorithm has a fixed output interval of 1 second



## Application of Eye Blink Detection

- Selecting Eye Blink Detection by invoking **NskAlgoInit()** method
- Starting EEG data analysis by invoking **NskAlgoStart()** method
- 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 **NskAlgoPause()** method
- Resuming EEG algorithm data analysis by invoking **NskAlgoStart()** method

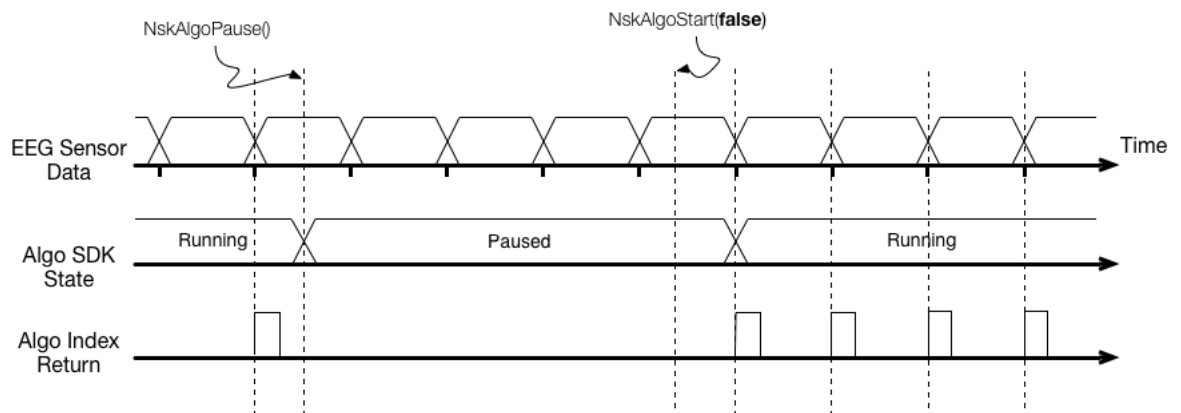


Figure 3.1: Time diagram on Pause/Resume SDK

**Note:** .

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

## Stop and Start

- Assuming SDK is in **RUNNING** state
- Stopping EEG algorithm data analysis by invoking **NskAlgoStop()** method
- Restart EEG algorithm data analysis by invoking **NskAlgoStart()** method

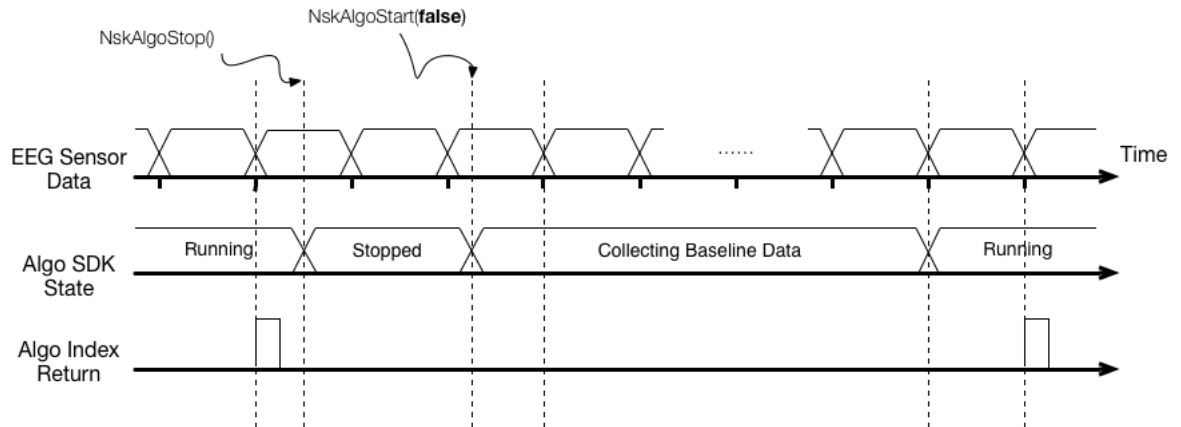


Figure 3.2: Time diagram on Stop/Start SDK

**Note: .**

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

## Customize Algorithm Output Interval

- Algorithm output interval can be configured at any time once SDK has been initialized
- The new configured output interval will become effective based on last index returned

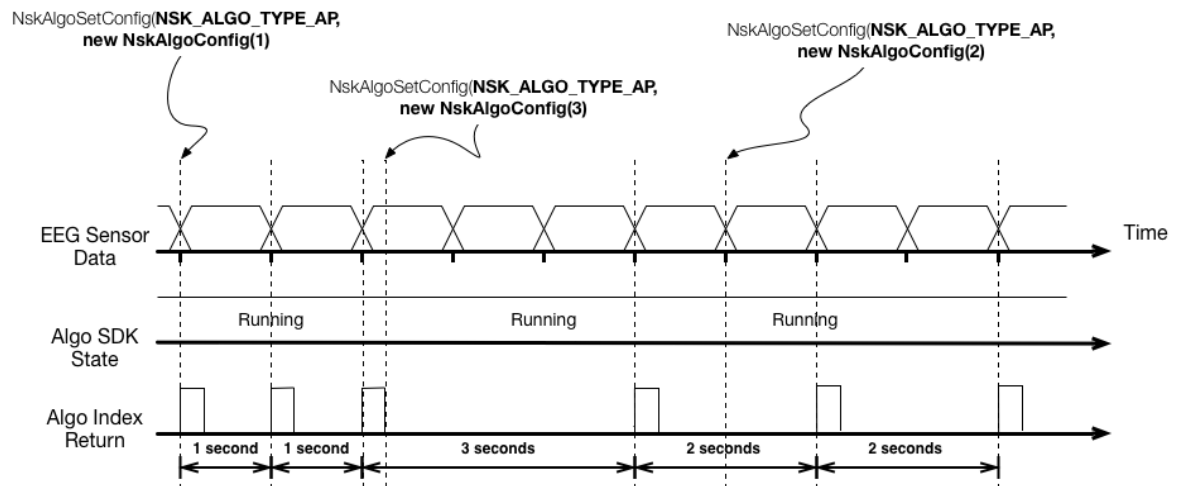


Figure 3.3: Time diagram on configuring algorithm output interval

**Note:** .

- Different algorithm may have different **minimum/default** output interval

# Frequently Asked Questions

---