# EEG algorithm SDK for iOS: Development Guide

**September 7, 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 iOS 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 iOS** *with* EEG data collected by NeuroSky Biosensor System (e.g. TGAM module or MindWave Mobile Headset).

This development guide is intended for *iOS application developers* who are already familiar with standard iOS development using **Xcode** and Apple's iOS SDK. If you are not already familiar with developing for iOS, please first visit Apple's web site for instruction and tools to develop iOS apps.

**Important:** .

- Requires iOS 8.0 or later

## EEG Algorithm SDK for iOS Contents

- EEG Algorithm SDK for iOS: Development Guide (this document)
- EEG algorithms descriptions
- EEG Algorithm SDK framework: AlgoSdk.framework (compatible with both iOS device and simulator)
- Readme: Readme file
- Algo SDK Sample project

## Application development

### Introduction

We recommend developers to use our COMM SDK for iOS 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 iOS device, please refer to our COMM SDK for iOS document.

**Algo SDK Sample** is an sample iOS 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 the iOS device
2. On the iOS app development machine (e.g. macbook), double click the Algo SDK Sample Xcode project (**"Algo SDK Sample.xcodeproj"**) to launch the project with Xcode
3. Select the iOS device (recommended to use iPad) connecting to the working headset as the target device
4. Update the code signing options in the project target settings
5. Select **Product** —> **Run** to build and install the "Algo SDK Sample" app
6. In the sample app, realtime Attention and Meditation indices would be shown on the progress bar. Also, realtime Appreciation and Mental Effort indices will be plotted on a graph.

**Important:** .

- The sample project requires XCode 6.0 or later.
- The sample project only demonstrates how to iterate with the EEG Algo SDK.
- **Comm SDK** enclosed in the sample project is **version 1.1.7**. Please make sure you are using the latest stable Comm SDK version and make proper changes on sample project if needed.
- It may not be completely compliant with Apple's guidelines for building deploy-able applications.

# API Documentation

The **EEG Algorithm SDK API Reference** in this section contains descriptions of the classes and protocols available in the EEG Algorithm iOS API.

## Data Types

```
/* EEG data signal quality definitions */
typedef NS_ENUM(NSInteger, NskAlgoSignalQuality) {
    NskAlgoSignalQualityGood,           /* Signal quality is in good level */
    NskAlgoSignalQualityMedium,         /* Signal quality is in medium level */
    NskAlgoSignalQualityPoor,           /* Signal quality is in poor level */
    NskAlgoSignalQualityNotDetected     /* Sensor signal is not detected */
};

/* SDK state definitions */
typedef NS_ENUM(NSInteger, NskAlgoState) {
    NskAlgoStateInited = 1,             /* Algo SDK initialized */
    NskAlgoStateRunning,                /* Algo SDK is performing analysis (i.e. startProcess()
invoked) */
    NskAlgoStateCollectingBaselineData, /* Algo SDK is collecting baseline data */
    NskAlgoStateStop,                   /* Algo SDK stops data analysis/baseline collection */
    NskAlgoStatePause,                  /* Algo SDK pauses data analysis */
    NskAlgoStateUninited,               /* Algo SDK is uninitialized */
    NskAlgoStateAnalysingBulkData       /* Algo SDK is analysing a bulk of EEG data */
};

/* SDK state change reason definitions */
typedef NS_ENUM(NSInteger, NskAlgoReason) {
    NskAlgoReasonConfigChanged = 1,      /* RESERVED: SDK configuration changed */
    NskAlgoReasonUserProfileChanged,     /* RESERVED: Active user profile has been changed */
    NskAlgoReasonUserTrigger,            /* User triggers */
    NskAlgoReasonBaselineExpired,        /* RESERVED: Baseline expired */
    NskAlgoReasonNoBaseline,             /* RESERVED: No baseline data collected yet */
    NskAlgoReasonSignalQuality,          /* Due to signal quality */
    NskAlgoReasonExpired,                /* FOR EVALUATION ONLY: SDK has been expired */
    NskAlgoReasonInternetError,          /* FOR EVALUATION ONLY: internet connection error */
    NskAlgoReasonKeyError                /* FOR EVALUATION ONLY: evaluation license key error */
};

/* EEG algorithm type definitions */
typedef NS_ENUM(NSInteger, NskAlgoEegType) {
    NskAlgoEegTypeAP    = 0x0001,          /* Appreciation */
    NskAlgoEegTypeME    = 0x0002,          /* Mental Effort */
    NskAlgoEegTypeME2   = 0x0004,          /* Mental Effort Secondary Algorithm */
    NskAlgoEegTypeAtt   = 0x0008,          /* Attention */
    NskAlgoEegTypeMed   = 0x0010,          /* Meditation */
    NskAlgoEegTypeF     = 0x0020,          /* Familiarity */
    NskAlgoEegTypeF2    = 0x0040,          /* Familiarity Secondary Algorithm */
    NskAlgoEegTypeBlink = 0x0080           /* Eye Blink Detection */
```

```
    NskAlgoEegTypeCR    = 0x0100,          /* Creativity */
    NskAlgoEegTypeAL    = 0x0200,          /* Alertness */
    NskAlgoEegTypeCP    = 0x0400,          /* Cognitive Preparedness */
    NskAlgoEegTypeBP    = 0x0800,          /* EEG Bandpower */
    NskAlgoEegTypeET    = 0x1000,          /* eTensity */
    NskAlgoEegTypeYY    = 0x2000           /* Yin-Yang */
};

/* EEG data type definitions (data from COMM SDK) */
typedef NS_ENUM(NSInteger, NskAlgoDataType) {
    NskAlgoDataTypeEEG,                 /* Raw EEG data */
    NskAlgoDataTypeAtt,                 /* Attention data */
    NskAlgoDataTypeMed,                 /* Meditation data */
    NskAlgoDataTypePQ,                  /* Poor signal quality data */
    NskAlgoDataTypeBulkEEG,             /* Bulk EEG data (must be multiple of 512, i.e. Ns of
continuous GOOD EEG data */
};

/* Familiary secondary algorithm progress level definitions */
typedef NS_ENUM(NSInteger, NskAlgoF2ProgressLevel) {
    NskAlgoF2ProgressLevelVeryBad = 1,
    NskAlgoF2ProgressLevelBad,
    NskAlgoF2ProgressLevelFlat,
    NskAlgoF2ProgressLevelGood,
    NskAlgoF2ProgressLevelGreat
};

/* Brain Conditioning Quantification threshold */
typedef NS_ENUM(NSInteger, NskAlgoBCQThreshold) {
    NskAlgoBCQThresholdLight = 0,
    NskAlgoBCQThresholdMedium,
    NskAlgoBCQThresholdHigh
};

/* Brain Conditioning Quantification return type */
typedef NS_ENUM(NSInteger, NskAlgoBCQIndexType) {
    NskAlgoBCQIndexTypeValue = 0,    /* only cr_value/al_value/cp_value is valid */
    NskAlgoBCQIndexTypeValid,        /* only BCQ_valid is valid */
    NskAlgoBCQIndexTypeBoth          /* both cr_value/al_value/cp_value and BCQ_valid are valid */
};
```

# SDK Delegate Methods

## stateChanged

EEG Algo SDK state change notification delegate method

```
// Required
- (void) stateChanged: (NskAlgoState)state reason:(NskAlgoReason)reason;
```

**Note:** .

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

## apAlgoIndex

Appreciation Algorithm index notification delegate method

```
// Required
- (void) apAlgoIndex: (NSNumber*)ap_index;
```

**Note:** .

- Appreciation algorithm has an optional output interval of 1,2,3,4 or 5 seconds
- Algorithm output: 1-Not at all, 2-Low, 3-Medium, and 4-High
- Please refer to EEG algorithm description for the definition and further details of the algorithm.
- **No** algorithm index will be returned until the SDK state equals to **RUNNING/ANALYSING BULK DATA** with EEG data fed

## meAlgoIndex

Mental Effort Algorithm index notification delegate method

```
// Optional
- (void) meAlgoIndex: (NSNumber*)abs_me diff_me:(NSNumber*)diff_me max_me:(NSNumber*)max_me
min_me:(NSNumber*)min_me;
```

**Note:** .

- Mental effort algorithm has an optional output interval of 1,2,3,4 or 5 seconds
- Algorithm output: -100 to 100
- Please refer to EEG algorithm description for the definition and further details of the algorithm.
- **No** algorithm index will be returned until the SDK state equals to **RUNNING/ANALYSING BULK DATA** with EEG data fed

## me2AlgoIndex

Mental Effort Secondary Algorithm index notification delegate method

```
// Optional
- (void) me2AlgoIndex: (NSNumber*)total_me me_rate:(NSNumber*)me_rate
changing_rate:(NSNumber*)changing_rate;
```

**Note:** .

- Please refer to EEG algorithm description for the definition and further details of the algorithm.
- Algorithm indices will be output based on configured output interval (i.e. output rate.) between 30 seconds and 36000 seconds (10 hours) with a step of 1 second (i.e. 30s, 31s, 32s, …, 36000s)
- **No** algorithm index will be returned until the SDK state equals to **RUNNING/ANALYSING BULK DATA** with EEG data fed

## fAlgoIndex

Familiarity Algorithm index notification delegate method

```
// Required
- (void) fAlgoIndex: (NSNumber*)abs_f diff_me:(NSNumber*)diff_f max_f:(NSNumber*)max_f
min_f:(NSNumber*)min_f;
```

**Note:** .

- Familiarity algorithm has an optional output interval of 1,2,3,4 or 5 seconds
- Algorithm output: -100 to 100
- Please refer to EEG algorithm description for the definition and further details of the algorithm.
- **No** algorithm index will be returned until the SDK state equals to **RUNNING/ANALYSING BULK DATA** with EEG data fed

## f2AlgoIndex

Familiarity Secondary Algorithm index notification delegate method

```
// Optional
- (void) f2AlgoIndex: (NSNumber*)progress_level f_degree:(NSNumber*)f_degree;
```

**Note:** .

- For the definition of algorithm indices, please refer to EEG algorithm description.
- Algorithm indices will be output based on configured output interval (i.e. output rate.) between 30 seconds and 36000 seconds (10 hours) with a step of 1 second (i.e. 30s, 31s, 32s, …, 36000s)
- **No** algorithm index will be returned until the SDK state equals to **RUNNING/ANALYSING BULK DATA** with EEG data fed

## attAlgoIndex

Attention Algorithm index notification delegate method

```
// Optional
- (void) attAlgoIndex: (NSNumber*)att_index;
```

**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
- In situation when there are more than one algorithm running at the same time (e.g. Attention and Appreciation), algorithm indices from Attention will still be returned when SDK state is **COLLECTING BASELINE**.

## medAlgoIndex

Meditation Algorithm index notification delegate method

```
// Optional
- (void) medAlgoIndex: (NSNumber*)med_index;
```

**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
- In situation when there are more than one algorithm running at the same time (e.g. Meditation and Appreciation), algorithm indices from Meditation will still be returned when SDK state is **COLLECTING BASELINE**.

## eyeBlinkDetect

Eye blink detection notification delegate method

```
// Optional
- (void) eyeBlinkDetect: (NSNumber*)strength;
```

**Note:** .

- **eyeBlinkDetect** will be invoked when eye blink is detected from the provided EEG data

## bpAlgoIndex

EEG Bandpower Algorithm notification delegate method

```
// Required
- (void) bpAlgoIndex: (NSNumber*)delta theta:(NSNumber*)theta alpha:(NSNumber*)alpha
beta:(NSNumber*)beta gamma:(NSNumber*)gamma;
```

**Note:** .

- EEG bandpower algorithm has a fix output rate as 1 second
- Algorithm output unit: dB
- **No** algorithm index will be returned until the SDK state equals to **RUNNING/ANALYSING BULK DATA** with EEG data fed

## crAlgoIndex

Creativity Algorithm value notification delegate method

```
// Optional
 (void) crAlgoIndex: (NskAlgoBCQIndexType)cr_index_type cr_value:(NSNumber*)cr_value
BCQ_valid:(BOOL)BCQ_valid;
```

**Note:** .

- Creativity algorithm value (i.e. cr_value) has an optional output interval of 1,2,3,4 or 5 seconds
- BCQ validation (i.e. cr_index_type == NskAlgoBCQIndexTypeValid or NskAlgoBCQIndex-TypeBoth) will only be outputed once during the trial
- Algorithm output: -1.0 to 1.0
- Please refer to EEG algorithm description for the definition and further details of the algorithm
- **No** algorithm index will be returned until the SDK state equals to **RUNNING/ANALYSING BULK DATA** with EEG data fed

## alAlgoIndex

Alertness Algorithm value notification delegate method

```
// Optional
 (void) alAlgoIndex: (NskAlgoBCQIndexType)al_index_type al_value:(NSNumber*)al_value
BCQ_valid:(BOOL)BCQ_valid;
```

**Note:** .

- Alertness algorithm value (i.e. al_value) has an optional output interval of 1,2,3,4 or 5 seconds
- BCQ validation (i.e. al_index_type == NskAlgoBCQIndexTypeValid or NskAlgoBCQIndex-TypeBoth) will only be outputed once during the trial
- Algorithm output: -1.0 to 1.0
- Please refer to EEG algorithm description for the definition and further details of the algorithm
- **No** algorithm index will be returned until the SDK state equals to **RUNNING/ANALYSING BULK DATA** with EEG data fed

## cpAlgoIndex

Cognitive Preparedness Algorithm value notification delegate method

```
// Optional
 (void) cpAlgoIndex: (NskAlgoBCQIndexType)cp_index_type cp_value:(NSNumber*)cp_value
BCQ_valid:(BOOL)BCQ_valid;
```

**Note:** .

- Cognitive Preparedness algorithm value (i.e. cp_value) has an optional output interval of 1,2,3,4 or 5 seconds
- BCQ validation (i.e. cp_index_type == NskAlgoBCQIndexTypeValid or NskAlgoBCQIndex-TypeBoth) will only be outputed once during the trial
- Algorithm output: -1.0 to 1.0
- Please refer to EEG algorithm description for the definition and further details of the algorithm
- **No** algorithm index will be returned until the SDK state equals to **RUNNING/ANALYSING BULK DATA** with EEG data fed

## etAlgoIndex

eTensity Algorithm value notification delegate method

```
// Optional
 (void) etAlgoIndex: (NSNumber*)et_index;
```

**Note:** .

- eTensity algorithm value (i.e. et_index) has an optional output interval of 1,2,3,4 or 5 seconds
- Algorithm output: 1 to 4
- Please refer to EEG algorithm description for the definition and further details of the algorithm
- **No** algorithm index will be returned until the SDK state equals to **RUNNING/ANALYSING BULK DATA** with EEG data fed

## yyAlgoIndex

Yin-Yang Algorithm value notification delegate method

```
// Optional
 (void) yyAlgoIndex: (NSNumber*)yy_index;
```

**Note:** .

- Yin-Yang algorithm value (i.e. yy_index) has an optional output interval of 5 or 10 seconds
- Algorithm output: -1 (Unpleasant emotion), 0 (Neutral) or 1 (Pleasant emotion)
- Please refer to EEG algorithm description for the definition and further details of the algorithm
- **No** algorithm index will be returned until the SDK state equals to **RUNNING/ANALYSING BULK DATA** with EEG data fed

## signalQuality

EEG data signal quality notification delegate method

```
// Required
- (void) signalQuality: (NskAlgoSignalQuality)signalQuality;
```

**Note:** .

- Signal Quality was measured and reported at a fixed output interval of 1 second
- SDK state will be changed from **RUNNING/COLLECTING BASELINE** 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
- There will be no signal quality notification when SDK state is **ANALYSING BULK DATA**

# SDK Utility Methods

## sharedInstance

Developer could always use this method to get the EEG Algorithm instance throughout the app

```
/*
 * Return: EEG Algo instance
 */
+ (id) sharedInstance;
```

**Example**

```
NskAlgoSdk *nskAlgoInstance = [NskAlgoSdk sharedInstance];
```

## getAlgoVersion

Get the specific EEG algorithm version

```
/*
 * Return: Specific EEG algorithm version
 */
- (NSString*) getAlgoVersion: (NskAlgoEegType)algoType;
```

**Example**

```
NSString *version = [NSString stringWithFormat:@"AP Algo Ver.: %@", [[NskAlgoSdk sharedInstance]
getAlgoVersion:NskAlgoEegTypeAP]];
UIAlertView *alert = [[UIAlertView alloc] initWithTitle:@""
                                         message:version
                                         delegate:nil
                                         cancelButtonTitle:@"OK"
                                         otherButtonTitles:nil];
[alert show];
```

## getSdkVersion

Get Algo SDK version

```
/*
 * Return: Algo SDK version
 */
- (NSString*) getSdkVersion;
```

**Example**

```
NSString *version = [NSString stringWithFormat:@"SDK Ver.: %@", [[NskAlgoSdk sharedInstance]
getSdkVersion]];
UIAlertView *alert = [[UIAlertView alloc] initWithTitle:@""
                                          message:version
                                          delegate:nil
                                          cancelButtonTitle:@"OK"
                                          otherButtonTitles:nil];
[alert show];
```

## setAlgorithmTypes

Select specific EEG algorithm(s)

```
/*
 * Return: 0 - Algo SDK is initialized successfully; Otherwise, something wrong with SDK
initialization (please contact with NeuroSky technical support)
 */
- (NSInteger) setAlgorithmTypes: (NskAlgoEegType)algoTypes licenseKey:(char*)licenseKey;
```

**Note:** .

- Parameter **licenseKey** is **optional** for evaluation build
- NeuroSky will provide a client specific **license key** to initialize the SDK

**Example 1 - Single algorithm**

```
// getting the Algo SDK instance
NskAlgoSdk *nskAlgo = [NskAlgoSdk sharedInstance];
// setting self as delegate
[nskAlgo setDelegate: self];
// selecting Appreciation algorithms
[nskAlgo setAlgorithmTypes: NskAlgoEegTypeAP licenseKey:(char*)"LICENSE_KEY_CHAIN"];
```

**Example 2 - Multiple algorithms**

```
// getting the Algo SDK instance
NskAlgoSdk *nskAlgo = [NskAlgoSdk sharedInstance];
// setting self as delegate
[nskAlgo setDelegate: self];
// selecting Appreciation, Attention and Meditation algorithms
[nskAlgo setAlgorithmTypes: NskAlgoEegTypeAP| NskAlgoEegTypeAtt| NskAlgoEegTypeMed
licenseKey:(char*)"LICENSE_KEY_CHAIN"];
```

**Note:** .

- By invoking **setAlgorithmTypes:** with supported EEG algorithm(s), the SDK will always change back to **INITED**

## setAlgoIndexOutputInterval

Configure the output interval (in seconds) of the specific algorithm index

```
/*
 * Return: TRUE on success; Otherwise, FALSE
 */
- (BOOL) setAlgoIndexOutputInterval: (NskAlgoEegType)algoType
outputInterval:(NSInteger)outputInterval;
```

**Note:** .

- Please refer to specific algorithm description for the specific range of output interval supported, e.g. Attention and Meditation support only output rate = 1s, while Appreciation supports an output rate from 1 to 5s.

**Example**

```
// getting the Algo SDK instance
NskAlgoSdk *nskAlgo = [NskAlgoSdk sharedInstance];
// setting self as delegate
[nskAlgo setDelegate: self];
// selecting Appreciation algorithm only
[nskAlgo setAlgorithmTypes: NskAlgoEegTypeAP];
// setting Appreciation index output interval be every 2 seconds
[nskAlgo setAlgoIndexOutputInterval: NskAlgoEegTypeAP outputInterval:2];
```

## setCreativityAlgoConfig

Configure the Creativity algorithm

```
/*
 * Return: TRUE on success; Otherwise, FALSE
 */
- (BOOL) setCreativityAlgoConfig: (NSInteger)outputInterval
threshold:(NskAlgoBCQThreshold)threshold window:(NSInteger)window;
```

**Note:** .

- Please refer to specific algorithm description for the specific range of output interval supported, e.g. Attention and Meditation support only output rate = 1s, while Appreciation supports an output rate from 1 to 5s.

**Example**

```
// getting the Algo SDK instance
NskAlgoSdk *nskAlgo = [NskAlgoSdk sharedInstance];
// setting self as delegate
[nskAlgo setDelegate: self];
// selecting Appreciation algorithm only
[nskAlgo setAlgorithmTypes: NskAlgoEegTypeCR];
// setting Creativity value output interval be every 5 seconds with light threshold and validate
every 30 seconds
[nskAlgo setCreativityAlgoConfig:5 threshold:NskAlgoBCQThresholdLight window:30];
```

## setAlertnessAlgoConfig

Configure the Alertness algorithm

```
/*
 * Return: TRUE on success; Otherwise, FALSE
 */
- (BOOL) setAlertnessAlgoConfig: (NSInteger)outputInterval threshold:(NskAlgoBCQThreshold)threshold
window:(NSInteger)window;
```

**Note:** .

- Please refer to specific algorithm description for the specific range of output interval supported, e.g. Attention and Meditation support only output rate = 1s, while Appreciation supports an output rate from 1 to 5s.

### Example

```
// getting the Algo SDK instance
NskAlgoSdk *nskAlgo = [NskAlgoSdk sharedInstance];
// setting self as delegate
[nskAlgo setDelegate: self];
// selecting Appreciation algorithm only
[nskAlgo setAlgorithmTypes: NskAlgoEegTypeAL];
// setting Creativity value output interval be every 5 seconds with light threshold and validate
every 30 seconds
[nskAlgo setAlertnessAlgoConfig:5 threshold:NskAlgoBCQThresholdLight window:30];
```

## setCognitivePreparednessAlgoConfig

Configure the Cognitive Preparedness algorithm

```
/*
 * Return: TRUE on success; Otherwise, FALSE
 */
- (BOOL) setCognitivePreparednessAlgoConfig: (NSInteger)outputInterval
threshold:(NskAlgoBCQThreshold)threshold window:(NSInteger)window;
```

**Note:** .

- Please refer to specific algorithm description for the specific range of output interval supported, e.g. Attention and Meditation support only output rate = 1s, while Appreciation supports an output rate from 1 to 5s.

### Example

```
// getting the Algo SDK instance
NskAlgoSdk *nskAlgo = [NskAlgoSdk sharedInstance];
// setting self as delegate
[nskAlgo setDelegate: self];
// selecting Appreciation algorithm only
[nskAlgo setAlgorithmTypes: NskAlgoEegTypeCP];
// setting Creativity value output interval be every 5 seconds with light threshold and validate
every 30 seconds
[nskAlgo setCognitivePreparednessAlgoConfig:5 threshold:NskAlgoBCQThresholdLight window:30];
```

## dataStream

Feed-in realtime (from COMM SDK) or offline (recorded) EEG data to the EEG Algo SDK

**Important:** .

- EEG Algo SDK handles only the following **4** realtime data output from NeuroSky Biosensor System for now:
  - Poor Signal Quality
  - EEG Raw Data
  - Attention
  - Meditation

```
/*
 * Return: TRUE on success; Otherwise, FALSE
 */
- (BOOL) dataStream: (NskAlgoDataType)type data:(int16_t*)data length:(int16_t)length;
```

**Example 1 - Handling realtime EEG data**

```
// In the COMM SDK **onDataReceived** delegate method
-(void) onDataReceived:(NSInteger)datatype data:(int)data obj:(NSObject *)obj
deviceType:(DEVICE_TYPE)deviceType {
    if (deviceType != DEVICE_TYPE_MindWaveMobile) {
        return;
    }
    switch (datatype) {
        case MindDataType_CODE_POOR_SIGNAL:
        {
            int16_t poor_signal[1];
            poor_signal[0] = (int16_t)data;
            [[NskAlgoSdk sharedInstance] dataStream:NskAlgoDataTypePQ data:poor_signal length:1];
        }
        break;

        case MindDataType_CODE_RAW:
        {
            int16_t eeg_data[1];
            eeg_data[0] = (int16_t)data;
            [[NskAlgoSdk sharedInstance] dataStream:NskAlgoDataTypeEEG data:eeg_data length:1];
        }
        break;

        case MindDataType_CODE_ATTENTION:
        {
            int16_t attention[1];
            attention[0] = (int16_t)data;
            [[NskAlgoSdk sharedInstance] dataStream:NskAlgoDataTypeAtt data:attention length:1];
        }
        break;

        case MindDataType_CODE_MEDITATION:
        {
```

```
            int16_t meditation[1];
            meditation[0] = (int16_t)data;
            [[NskAlgoSdk sharedInstance] dataStream:NskAlgoDataTypeMed data:meditation length:1];
        }
        break;
}
```

**Example 2 - Analysing recorded EEG data (bulk data analysis)**

```
- (void) sendBulkData:(int16_t*)raw_data raw_data_len:(int32_t)raw_data_len {
    if ([[NskAlgoSdk sharedInstance] dataStream:NskAlgoDataTypeBulkEEG data:raw_data
length:(int32_t)raw_data_len] == TRUE) {
        NSLog(@"Bulk data has been sent successfully");
    }
}
```

## startProcess

Start analysing feed-in EEG data with selected EEG algorithm(s)

```
/*
 * Return: TRUE on success; Otherwise, FALSE
 */
- (BOOL) startProcess;
```

**Example**

```
// getting the Algo SDK instance
NskAlgoSdk *nskAlgo = [NskAlgoSdk sharedInstance];
// setting self as delegate
[nskAlgo setDelegate: self];
// selecting Appreciation algorithm only
[nskAlgo setAlgorithmTypes: NskAlgoEegTypeAP];
// start analysing EEG data
[nskAlgo startProcess];
```

**Note:** .

- SDK state will only change to **RUNNING/COLLECTING BASELINE/ANALYSING BULK DATA** by invoking **startProcess**

## pauseProcess

Pause analysing feed-in EEG data

```
/*
 * Return: TRUE on success; Otherwise, FALSE
 */
- (BOOL) pauseProcess;
```

**Example**

```
// User presses PAUSE button on app to pause the data analysis
- (IBAction)pausePress:(id)sender {
    [[NskAlgoSdk sharedInstance] pauseProcess];
}
```

**Note:** .

- SDK state will change to **PAUSE**
- No **algoIndex** delegate method will not be invoked unless **startProcess** method is invoked again
- When SDK state is **ANALYSING BULK DATA**, then **pauseProcess:** will always return FALSE (i.e. no effect)

## stopProcess

Stop analysing feed-in EEG data

```
/*
 * Return: TRUE on success; Otherwise, FALSE
 */
- (BOOL) stopProcess;
```

**Example**

```
// User presses STOP button on app to stop the data analysis
- (IBAction)stopPress:(id)sender {
    [[NskAlgoSdk sharedInstance] stopProcess];
}
```

**Note:** .

- SDK state will change to **STOP**
- No **algoIndex** delegate method will be invoked unless **startProcess** method is invoked again
- **stopProcess** requires recollection of baseline data once restart (Exception for Attention and Meditation) while **pauaseProcess** doesn't.

# Applications

## SDK State Diagram

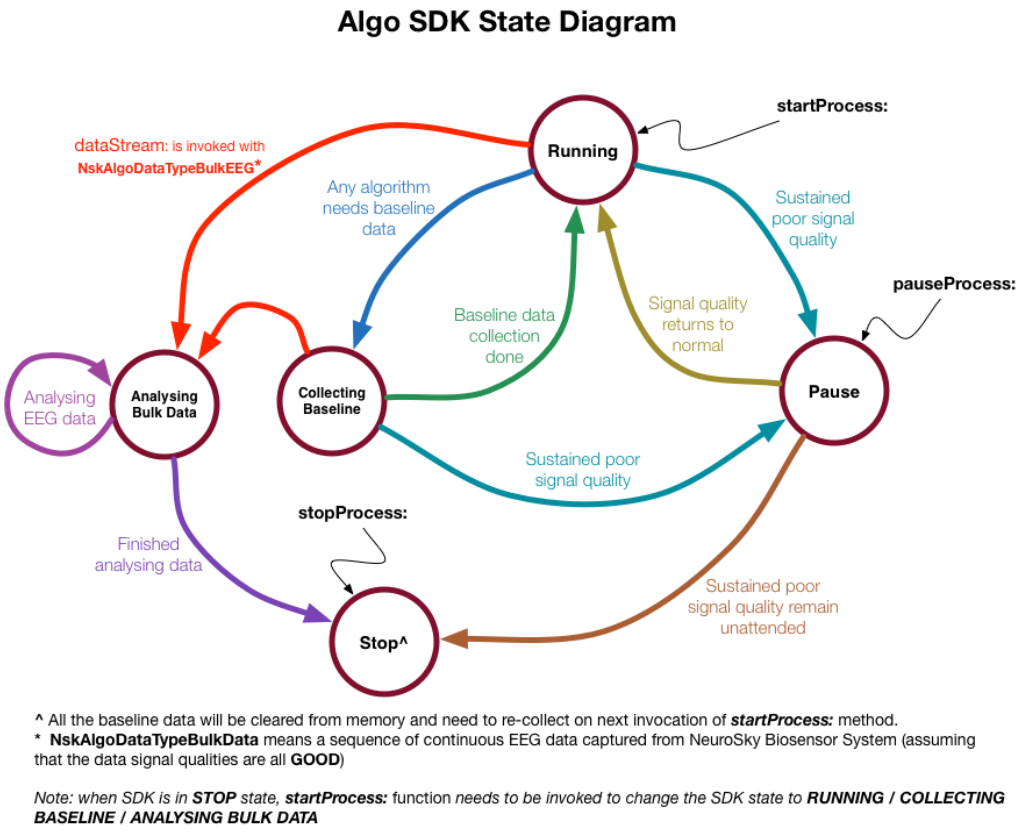


Figure 3.1: Algo SDK state diagram

## Application of Appreciation Algorithm

- Selecting Appreciation Algorithm by invoking **setAlgorithmTypes:** method
- Starting EEG data analysis by invoking **startProcess:** method
- **Procedure**
    1. **Baseline Data Collection**

  - When previous Algo SDK state is **INITED/STOPPED**, then the first 5s EEG raw data will be used as baseline data (i.e. SDK state will be **COLLECTING BASELINE**)
  - Suggested baseline data collection protocol

2. **Appreciation Index Computation**
  - Appreciation indexes are computed and returned at a configurable **algorithm output interval**

**Note:** .

- The default output interval of Appreciation is 1 second, i.e. one new Appreciation index every second (it also represents the minimal output interval)

- Definition of Appreciation index

| Appreciation Index | Appreciation or Enjoyment level |
|---|---|
| 1 | Not at all |
| 2 | Low |
| 3 | Medium |
| 4 | High |

- Please find below the block diagram showing the operation procedures and interaction between EEG data blocks, function calls, algorithm and application outputs in different output rate conditions.

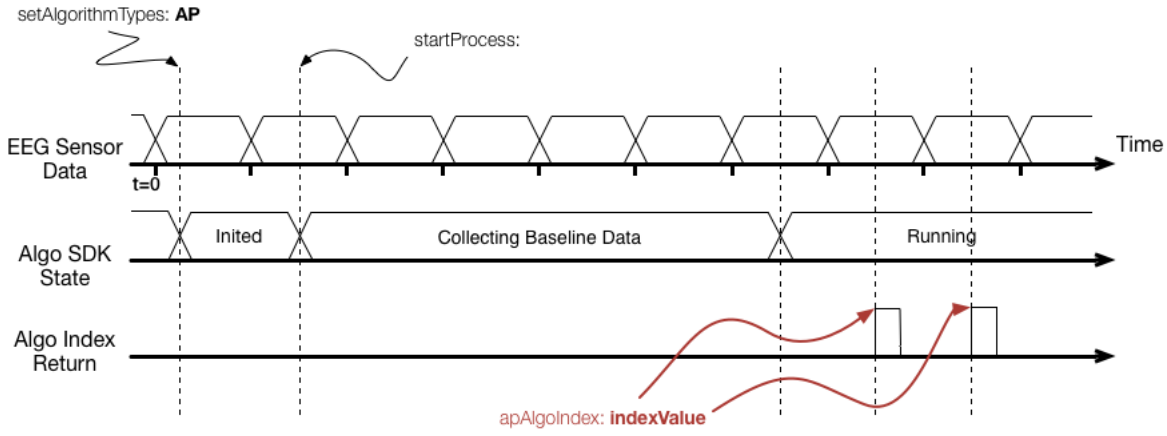


Figure 3.2: Time diagram on getting Appreciation Indices

**Note:** .

- Please note that user might not have fully engaged in the application (e.g. video watching) in the first few seconds. In such case, it's possible that the obtained algorithm results for that period might not truly reflect user's *expected* mental state induced by the application.

# Application of Attention Algorithm

- Selecting Attention Algorithm by invoking **setAlgorithmTypes:** method
- Starting EEG data analysis by invoking **startProcess:** 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
- No baseline data collection will be needed
- In the case of selecting multiple algorithms (e.g. Attention and Appreciation), there would still be Attention index returning while Appreciation or other algorithms is collecting baseline data (i.e. Algo SDK state is **COLLECTING BASELINE**)

# Application of Meditation Algorithm

- Selecting Meditation Algorithm by invoking **setAlgorithmTypes:** method
- Starting EEG data analysis by invoking **startProcess:** 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
- No baseline data collection will be needed
- In the case of selecting multiple algorithms (e.g. Meditation and Appreciation), there would still be Meditation index returning while Appreciation or other algorithms is collecting baseline data (i.e. Algo SDK state is **COLLECTING BASELINE**)

# Application of Eye Blink Detection

- Selecting Eye Blink Detection by invoking **setAlgorithmTypes:** method
- Starting EEG data analysis by invoking **startProcess:** 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

## Application of EEG Bandpower Algorithm

- Selecting EEG bandpower algorithm by invoking **setAlgorithmTypes:** method
- Starting EEG data analysis by invoking **startProcess:** method
- EEG bandpowers (delta, theta, alpha, beta and gamma in dB) will be returned in 5 seconds after Algo SDK state is **RUNNING** and the EEG bandpower values will be returned in every second

**Note:** .

- No baseline data collection will be needed

## Application of Creativity Algorithm

- Selecting Creativity Algorithm by invoking **setAlgorithmTypes:** method
- Starting EEG data analysis by invoking **startProcess:** method
- **Procedure**
  1. **Baseline Data Collection**
     - When previous Algo SDK state is **INITED/STOPPED**, then the first 5s EEG raw data will be used as baseline data (i.e. SDK state will be **COLLECTING BASELINE**)
     - Suggested baseline data collection protocol
  2. **Creativity Value and Quantification Computation**
     - Creativity values are computed and returned at a configurable **algorithm output interval**
     - Creativity quantification validation are returned at a configurable **algorithm window interval**

**Note:** .

- The default output interval of Creativity value and quantification is 1 seconds and 30 seconds respectively.

## Application of Alertness Algorithm

- Selecting Alertness Algorithm by invoking **setAlgorithmTypes:** method
- Starting EEG data analysis by invoking **startProcess:** method

- **Procedure**
  1. **Baseline Data Collection**
     - When previous Algo SDK state is **INITED/STOPPED**, then the first 5s EEG raw data will be used as baseline data (i.e. SDK state will be **COLLECTING BASELINE**)
     - Suggested baseline data collection protocol
  2. **Alertness Value and Quantification Computation**
     - Alertness values are computed and returned at a configurable **algorithm output interval**
     - Alertness quantification validation are returned at a configurable **algorithm window interval**

**Note:** .

- The default output interval of Alertness value and quantification is 1 seconds and 30 seconds respectively.

## Application of Cognitive Preparedness Algorithm

- Selecting Cognitive Preparedness Algorithm by invoking **setAlgorithmTypes:** method
- Starting EEG data analysis by invoking **startProcess:** method
- **Procedure**
  1. **Baseline Data Collection**
     - When previous Algo SDK state is **INITED/STOPPED**, then the first 5s EEG raw data will be used as baseline data (i.e. SDK state will be **COLLECTING BASELINE**)
     - Suggested baseline data collection protocol
  2. **Cognitive Preparedness Value and Quantification Computation**
     - Cognitive Preparedness values are computed and returned at a configurable **algorithm output interval**
     - Cognitive Preparedness quantification validation are returned at a configurable **algorithm window interval**

**Note:** .

- The default output interval of Cognitive Preparedness value and quantification is 1 seconds and 30 seconds respectively.

## Application of eTensity Algorithm

- Selecting eTensity Algorithm by invoking **setAlgorithmTypes:** method

- Starting EEG data analysis by invoking **startProcess:** method
- **Procedure**
  1. **Baseline Data Collection**
     - When previous Algo SDK state is **INITED/STOPPED**, then the first 5s EEG raw data will be used as baseline data (i.e. SDK state will be **COLLECTING BASELINE**)
     - Suggested baseline data collection protocol
  2. **eTensity Index**
     - eTensity indexes are computed and returned at a configurable **algorithm output interval**

**Note:** .

- The default output interval of eTensity index is 1 second.

## Application of Yin-Yang Algorithm

- Selecting Yin-Yang Algorithm by invoking **setAlgorithmTypes:** method
- Starting EEG data analysis by invoking **startProcess:** method
- **Procedure**
  1. **Baseline Data Collection**
     - When previous Algo SDK state is **INITED/STOPPED**, then the first 5s EEG raw data will be used as baseline data (i.e. SDK state will be **COLLECTING BASELINE**)
     - Suggested baseline data collection protocol
  2. **Yin-Yang Index**
     - Yin-Yang indexes are computed and returned at a configurable **algorithm output interval**

**Note:** .

- The default output interval of Yin-Yang index is 5 second.

## Application of Mental Effort Algorithm

- Selecting Mental Effort Algorithm by invoking **setAlgorithmTypes:** method
- Starting EEG data analysis by invoking **startProcess:** method
- **Procedure**
  1. **Baseline Data Collection**

  - When previous Algo SDK state is **INITED/STOPPED**, then the first 5s EEG raw data will be used as baseline data (i.e. SDK state will be **COLLECTING BASELINE**)
  - Suggested baseline data collection protocol

2. **Mental Effort Index Computation**
  - Mental Effort index will be output as a group of four values: **Maximum value of Absolute Mental Effort**, **Minimum value of Absolute Mental Effort**, **Absolute Mental Effort** & **Differential Mental Effort**
  - Mental Effort indexes are computed and returned at a configurable **algorithm output interval**

**Note:** .

- The default output interval of Mental Effort is 1 second, i.e. one new Mental Effort index every second (it also represents the minimal output interval)

- Definition of Mental Effort index: Mental Effort Application Development Guide
- Please find below the block diagram showing the operation procedures and interaction between EEG data blocks, function calls, algorithm and application outputs in different output rate conditions.

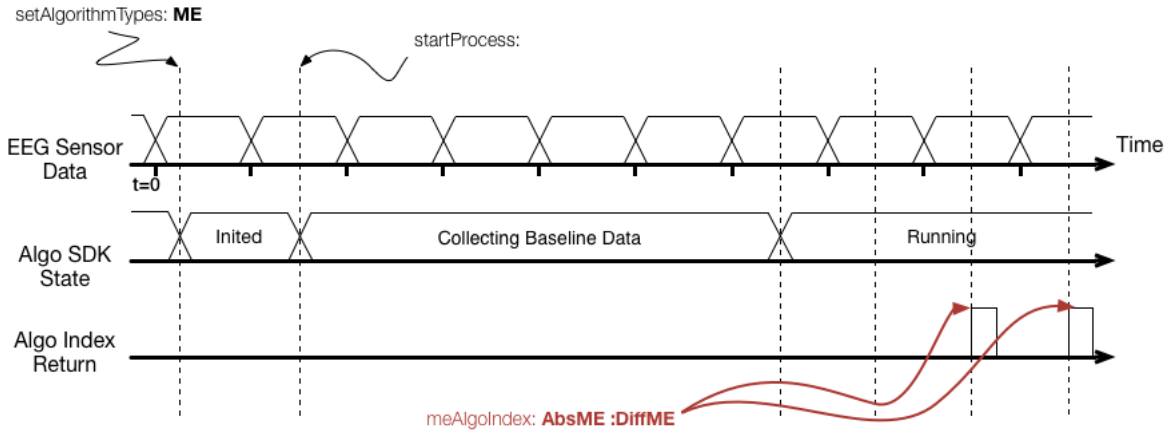


Figure 3.3: Time diagram on getting Mental Effort Indices

# Application of Mental Effort Secondary Algorithm

- Selecting Mental Effort Secondary Algorithm (*NskAlgoEegTypeME2*) by invoking **setAlgorithmTypes:** method
- Starting EEG data analysis by invoking **startProcess:** method
- **Procedure**

1. **Baseline Data Collection**

- When previous Algo SDK state is **INITED/STOPPED**, then the first 5s EEG raw data will be used as baseline data (i.e. SDK state will be **COLLECTING BASELINE**)
- Suggested baseline data collection protocol

2. **Mental Effort Secondary Algorithm Index Computation**
   - Mental Effort Secondary Algorithm Indexes consist of three output values: **Total Mental Effort**, **Mental Effort Rate** & **Mental Effort Changing Rate**
   - They are computed and returned at a configurable **algorithm output interval**

**Note:** .

- Mental Effort Secondary Algorithm has a configurable output interval from 30 seconds to 36000 seconds (10 hours) with a step of 1 second (i.e. 30s, 31s, 32s, …, 36000s).

- For the definition of algorithm indices, please refer to EEG algorithm description.
- Please find below the block diagram showing the operation procedures and interaction between EEG data blocks, function calls, algorithm and application outputs in different output rate conditions.

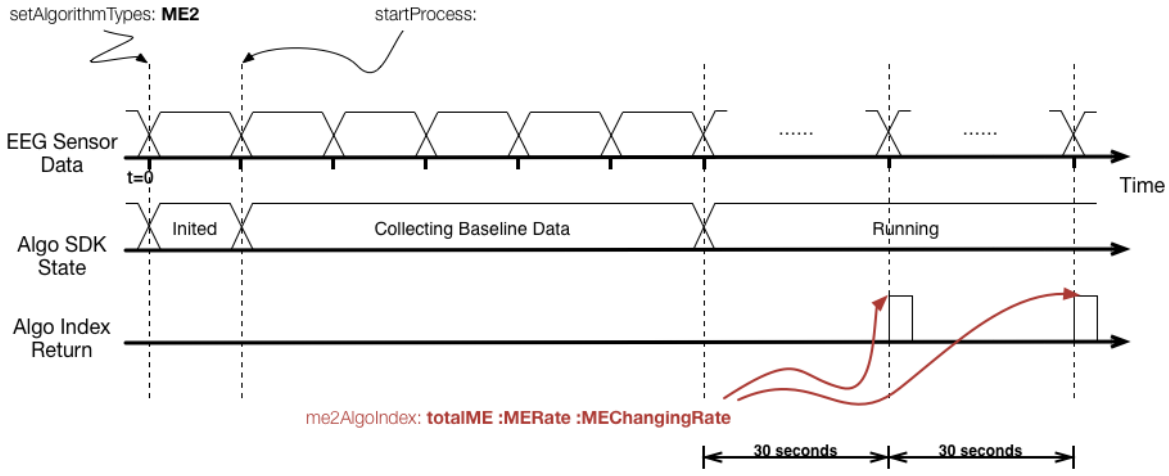


Figure 3.4: Time diagram on getting Mental Effort Indices

**Note:** .

- Different with Mental Effort and Appreciation that the first valid Mental Effort Secondary Index will **only** be ready after *output interval* seconds after finishing *baseline collection*

**Important:** .

- There are some special handles on Mental Effort Secondary Algorithm, as it's an extended version of Mental Effort:
  - **Realtime Data Analysis**
    1. selecting ***both*** **Mental Effort** and **Mental Effort Secondary** algorithms, **Mental Effort** algorithm will always with an output rate of **5 seconds** and **Mental Effort Secondary** algorithm will be output based on the configured output interval
    2. selecting ***only*** **Mental Effort Secondary** algorithm, ***only*** **Mental Effort Secondary** Index will be output based on the corresponding output interval
  - **Bulk Data Analysis (offline mode)**
    1. selecting ***both*** **Mental Effort** and **Mental Effort Secondary** algorithms, **Mental Effort** algorithm will always with an output rate of **5 seconds** and **Mental Effort Secondary** algorithm will be output based on the configured output interval
    2. selecting ***only*** **Mental Effort Secondary** algorithm, ***only*** **Mental Effort Secondary** Index will be output based on the corresponding output interval

# Application of Familiarity Algorithm

- Selecting Familiarity Algorithm (*NskAlgoEegTypeF*) by invoking **setAlgorithmTypes:** method
- Starting EEG data analysis by invoking **startProcess:** method
- **Procedure**
  1. **Baseline Data Collection**
     - When previous Algo SDK state is **INITED/STOPPED**, then the first 5s EEG raw data will be used as baseline data (i.e. SDK state will be **COLLECTING BASELINE**)
     - Suggested baseline data collection protocol
  2. **Familiarity Index Computation**
     - Familiarity index will be output as a group of four values: **Maximum value of Absolute Familiarity**, **Minimum value of Absolute Familiarity**, **Absolute Familiarity** & **Differential Familiarity**
     - Familiarity indexes are computed and returned at a configurable **algorithm output interval**

**Note:** .

- The default output interval of Familiarity is 1 second, i.e. one new Familiarity index every second (it also represents the minimal output interval)

- Definition of Familiarity index: Familiarity Application Development Guide

- Please find below the block diagram showing the operation procedures and interaction between EEG data blocks, function calls, algorithm and application outputs in different output rate conditions.

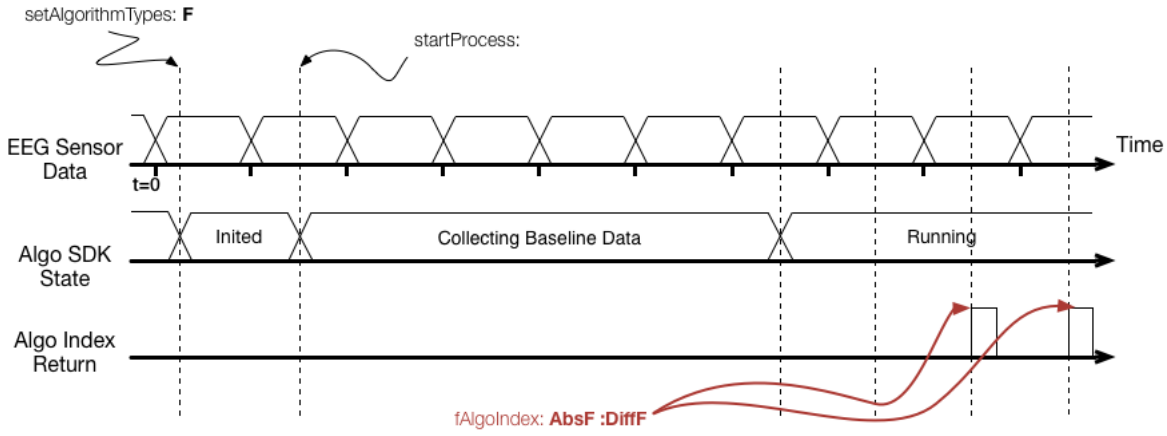


Figure 3.5: Time diagram on getting Familiarity Indices

# Application of Familiarity Secondary Algorithm

- Selecting Familiarity Secondary Algorithm (*NskAlgoEegTypeF2*) by invoking **setAlgorithmTypes:** method
- Starting EEG data analysis by invoking **startProcess:** method
- **Procedure**
  1. **Baseline Data Collection**
     - When previous Algo SDK state is **INITED/STOPPED**, then the first 5s EEG raw data will be used as baseline data (i.e. SDK state will be **COLLECTING BASELINE**)
     - Suggested baseline data collection protocol
  2. **Familiarity Secondary Algorithm Index Computation**
     - Familiarity Secondary Algorithm Indexes consist of three output values: **Progress Level** & **Familiarity Degree**
     - They are computed and returned at a configurable **algorithm output interval**

**Note:** .

- The output interval of Familiarity Secondary Algorithm can be configured in between 30 seconds and 36000 seconds (10 hours) with a step of 1 second (i.e. 30s, 31s, 32s, …, 36000s)

- For the definition of algorithm indices, please refer to EEG algorithm description.

- Please find below the block diagram showing the operation procedures and interaction between EEG data blocks, function calls, algorithm and application outputs in different output rate conditions.

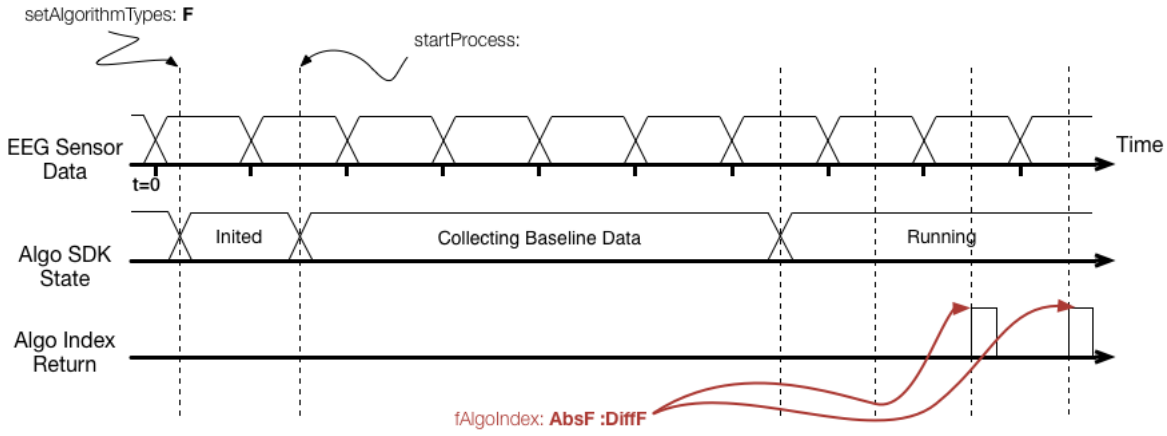


Figure 3.6: Time diagram on getting Familiarity with Secondary Algorithm Indices

**Note:** .

- Similar as Mental Effort Secondary Algorithm that Familiarity Secondary Algorithm Index will **only** be ready after *output interval* seconds after finishing *baseline collection*

**Important:** .

- There are some special handles on Familiarity Secondary Algorithm, as it's an extended version of Familiarity:
  - **Realtime Data Analysis**
    1. selecting ***both*** **Familiarity** and **Familiarity Secondary** algorithms, **Familiarity** algorithm will always with an output rate of **5 seconds** and **Familiarity Secondary** algorithm will be output based on the configured output interval
    2. selecting ***only*** **Familiarity Secondary** algorithm, ***only*** **Familiarity Secondary** Index will be output based on the corresponding output interval
  - **Bulk Data Analysis (offline mode)**
    1. selecting ***both*** **Familiarity** and **Familiarity Secondary** algorithms, **Familiarity** algorithm will always with an output rate of **5 seconds** and **Familiarity Secondary** algorithm will be output based on the configured output interval
    2. selecting ***only*** **Familiarity Secondary** algorithm, ***only*** **Familiarity Secondary** Index will be output based on the corresponding output interval

# SDK Operations

## Pause and Resume

- Assuming SDK is in **RUNNING/COLLECTING BASELINE** state
- Pausing EEG algorithm data analysis by invoking **pauseProcess:** method
- Resuming EEG algorithm data analysis by invoking **startProcess:** method

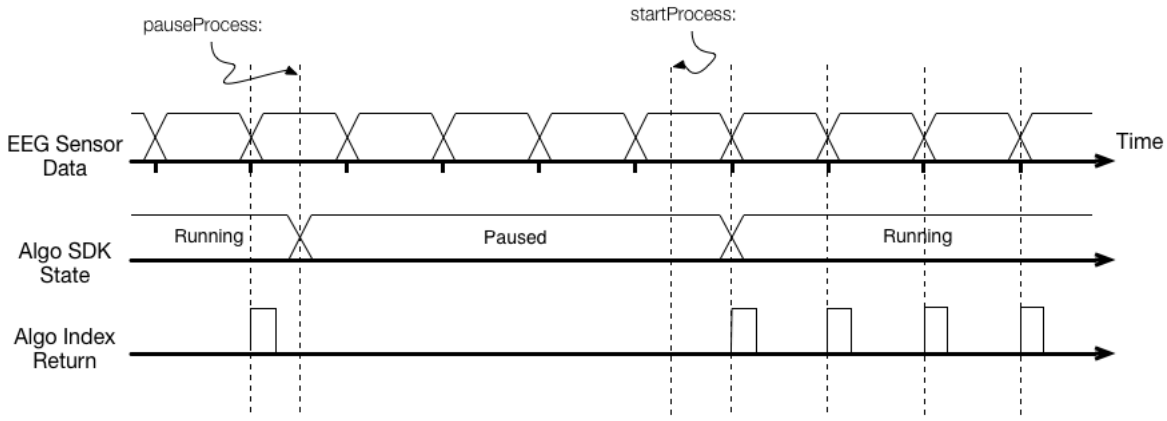


Figure 3.7: Time diagram on Pause/Resume SDK

**Note:** .

- There will be no effect on **pauseProcess:** when previous SDK state is not **RUNNING/COLLECTING BASELINE**
- When SDK state is **ANALYSING BULK DATA**, then **pauseProcess:** will always return FALSE (i.e. no effect)

## Stop and Start

- Assuming SDK is in **RUNNING/COLLECTING BASELINE/ANALYSING BULK DATA** state
- Stopping EEG algorithm data analysis by invoking **stopProcess:** method
- Restart EEG algorithm data analysis by invoking **startProcess:** method

**Note:** .

- SDK will need to re-collect baseline data after **stopProcess:** (i.e. SDK state will change to **COLLECTING BASELINE** right after invoking **startProcess:** method)

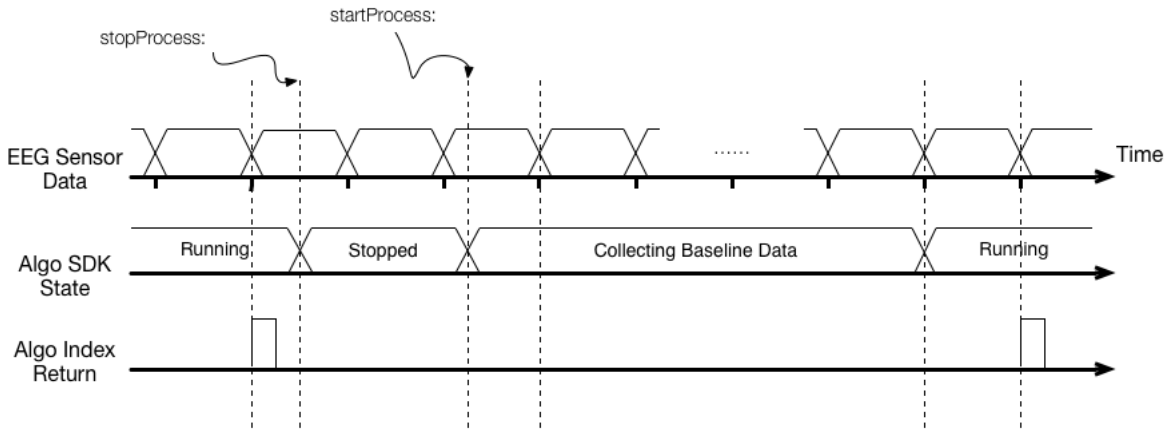


Figure 3.8: Time diagram on Stop/Start SDK

**Note:** .

- There will be no effect on **stopProcess:** when previous SDK state is not **RUNNING/COLLECTING BASELINE/ANALYSING BULK DATA**

# 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

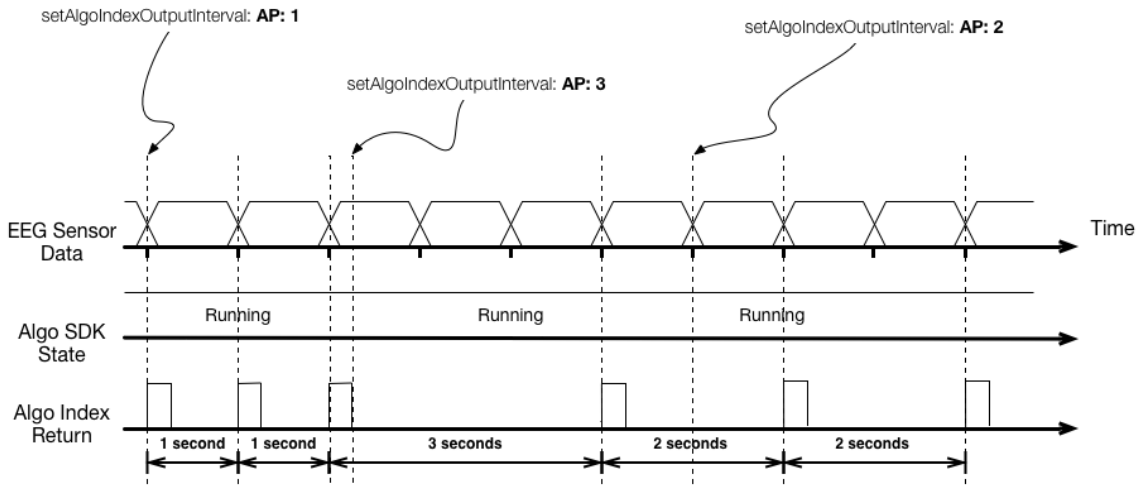


Figure 3.9: Time diagram on configuring algorithm output interval

**Note:** .

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

# Bulk EEG Data Analysis

Bulk EEG data analysis allows the application to perform **offline** analysis of recorded EEG data.

Below shows difference between handling realtime (directly from NeuroSky Biosensor System) and offline (recorded EEG data from NeuroSky Biosensor System) EEG raw data:

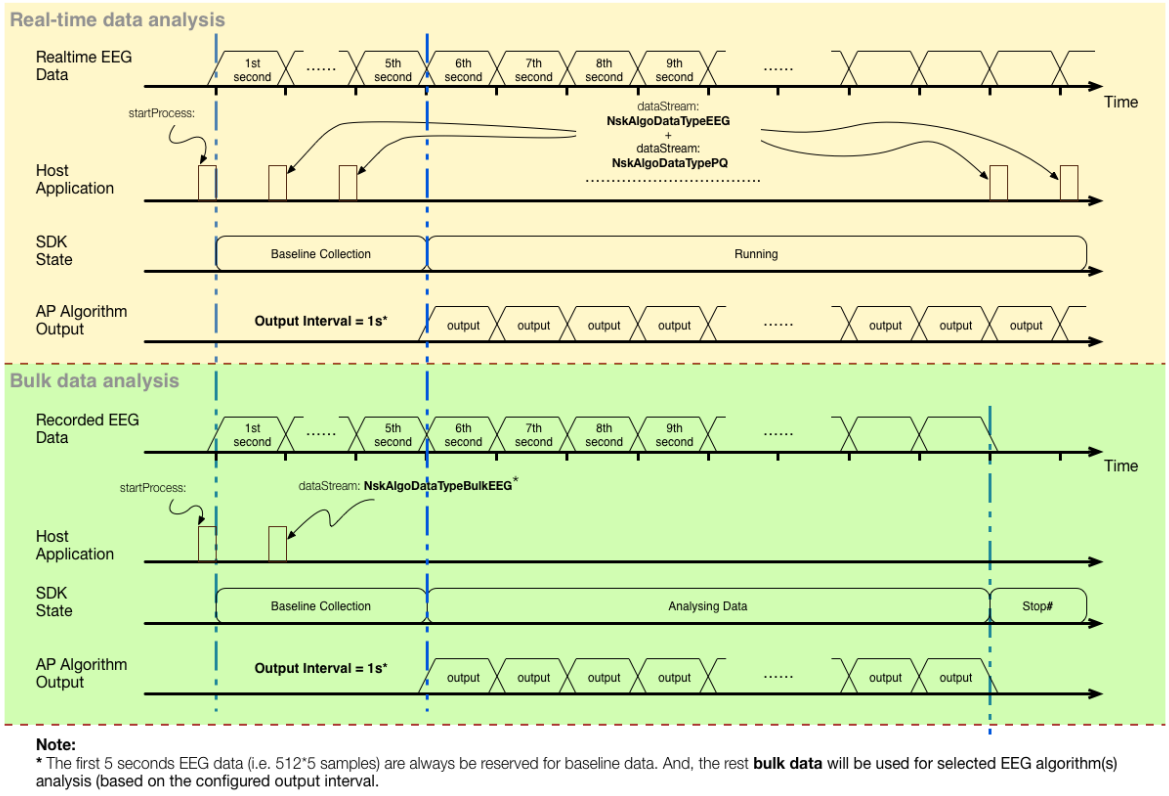


Figure 3.10: Time diagram on analysing EEG in real-time / offline

**Note:** .

- Attention, Meditation and Eye blink detection algorithm **doesn't** support *bulk EEG data analysis*

# Frequently Asked Questions