Closed-Loop Library Documentation

BWClosedLoop.dll Version 1.1.0.1

Date June 2014

3Brain GmbH Schulstrasse 76 7302 Landquart Switzerland
Tel +41 813227086 Fax +41 813227087 info@3brain.com 3brain.com

3Brain
Tel +41 813227086 · Fax +41 813227087 · info@3brain.com · www.3brain.com

Contents

1. Introduction 5

1.1. About this manual 5

1.2. Requirements 5

1.3. Use the library for development 5

2. Overview 6

2.1. Specifications and data format 6

2.2. Tests 7

3. Function Reference 8

3.1. Subscribe 8

3.2. Unsubscribe 8

3.3. SetCallbackThreadPriority 9

3.4. GetNBuffers 9

3.5. GetBufferLevel 9

3.6. GetSamplingRate 9

3.7. GetMVOffset 9

3.8. GetADCCountsToMV 10

3.9. GetArrayNRows 10

3.10. GetArrayNCols 10

3.11. GetNAcqChannels 10

3.12. GetIndexesOfAcqChannels 11

3.13. ChannelIndexToSubscripts 11

3.14. ChannelSubscriptsToIndex 11

3.15. GetAcqFramePositionOfChannel 12

3 Brain Tel +41 81 3227086 · Fax +41 81 3227087 · info@3brain.com · www.3brain.com

3 Brain
Tel +41 813227086 · Fax +41 813227087 · info@3brain.com · www.3brain.com

1. Introduction

1.1. About this manual

This manual comprises information about the BWClosedLoop library that can be used to access BioCAM acquired data in real-time. It is assumed that the reader already has a basic understanding of technical and software terms.

BWClosedLoop.dll has been written in the C++ programming language and it extends the functionality of BrainWave.

BWClosedLoop.dll can be used to implement a custom application in order to access online data, process it and provide a feedback by means of external stimulation to your biological preparation, hence closing the loop.

1.2. Requirements

The library is intended to be used under Windows operating systems with 64-bit architecture. In particular the library was tested to work on Windows 7 and 8.

1.3. Use the library for development

To use the library you need to include ClosedLoop.h and to link to the BWClosedLoop.lib library.

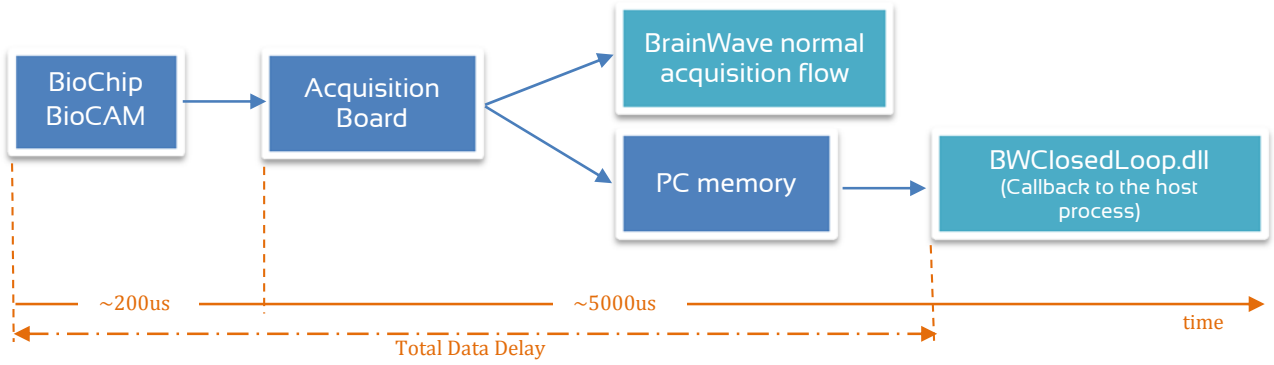
3 Brain Tel +41 81 3227086 · Fax +41 81 3227087 · info@3brain.com · www.3brain.com

Tel +41 813227086 · Fax +41 813227087 · info@3brain.com · www.3brain.com

2. Overview

2.1. Specifications and data format

The library extends BrainWave and allows the user to interact with the acquisition process by subscribing to Callbacks in order to read directly the data before it will be processed by normal acquisition flow, as in the following picture.

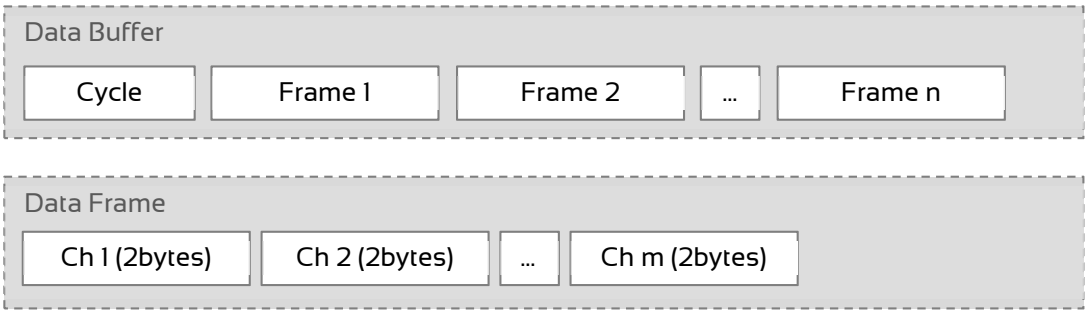


The *Total Data Delay* represents the moment the data is read from the BioChip and the moment the data is available through the BWClosedLoop library and depends from several factors.
The intrinsic delay between the reading of a BioChip activity frame and the moment data is received by the acquisition board is in the order of 200µs. Afterwards, the data is buffered inside the acquisition board, delivered to the host PC and buffered again in the PC memory. The BWClosedLoop.dll accesses this second buffer directly, which is referred from here on Closed-Loop Buffer. The Closed-Loop Buffer contains a user-defined number of subunit Data Buffers. Each of the Data Buffers contains in turn 5ms of acquired data and consequently the Callback is fired every 5ms as soon as one subunit Data Buffer is ready. This is to be intended as an **average** time since Windows is not a real-time OS, which means that occasionally the period between two consecutive callbacks might be longer.

In normal circumstances, when the callback is processed rapidly enough, the acquisition board buffer and the PC memory buffer are empty. In such case, the Total Data Delay is < 6ms.

However, in the case of a delay in the callback processing, the PC memory buffer can start filling up. This implies firstly that the Total Data Delay increases. Secondly, if the buffer goes overrun, it causes also a data loss.

Data obtained with the callback are in the following form:



Cycle index inside the DataBuffer is a progressive number that allows to know whether a buffer has been eventually lost since the previous callback.

2.2. Tests

Tests were performed on a quad-core (8 separated threads) 3.6GHz processor, while running BrainWave and an application tester using the BWClosedLoop.dll. Other applications were limited as possible and only basic background processes were running.

Callback processing was simulated by putting the callback thread on sleep. Hence while reading the results, it should be considered that i) on Windows a resolution lower than 1ms is not achievable with sleep functions, ii) the sleep time is the minimum time for sleeping and there is no guarantee that after the sleep time the scheduler will immediately execute the thread, iii) results might change in real applications when making extensive use of the processor and hence indirectly affecting also the BrainWave threads.

Results showed that the Closed-Loop buffer was steadily empty while having a sleep time of 2ms, while with a 3ms of sleep time the buffer periodically showed higher levels of buffered data, eventually leading to data loss. Setting the callback thread priority to `THREAD_PRIORITY_TIME_CRITICAL` (see `SetCallbackThreadPriority`) allowed to achieve a stable empty buffer with a 3ms of sleep. Higher level of sleep times always led to data loss.

3 Brain Tel +41 81 3227086 · Fax +41 81 3227087 · info@3brain.com · www.3brain.com

3 Brain
Tel +41 813227086 · Fax +41 813227087 · info@3brain.com · www.3brain.com

3. Function Reference

3.1. Subscribe

```
bool Subscribe(callback cb)
```

Parameters

cb [in]

a pointer to a function of type `void*(long long, unsigned char*, int)`. The callback function's three parameters are:

dataIndex

a progressive number representing the index of the acquired data chunk, since the beginning of the running acquisition

data

a pointer to an array of unsigned char representing a sequence of Data Frames.

dataLength

the length of the data.

Return value

If the function succeeds, the return value is true, false otherwise.

Description

By subscribing to the capture process, the caller receives a callback every time a Data Buffer has been acquired. The data index can be monitored in order to check that no Data Buffer is lost, which might happen one the time to process the callback is longer than the callback period.

A Data Buffer consists of consecutive Data Frames, where each Data Frame in turn is made up by a sequence of pair of bytes, one pair for each channel in the acquisition process. To know the position of an acquired channel within the Data Frame refer to `GetAcqFramePositionOfChannel`. Each pair of bytes codes the digital value of the recorded voltage amplitude with a 12 bits resolution. To convert into analog values, refer to `GetMVOffset` and `GetADCCountsToMV` functions.

The `ChCoord` class is used to represent electrodes of the MEA device. A channel is identified by its spatial coordinates, i.e. from the row and the column of its position.

3.2. Unsubscribe

```
void UnSubscribe()
```

Description

Unsubscribe from the callback routine.

3.3. SetCallbackThreadPriority

```
void SetCallbackThreadPriority(int priority)
```

Parameters
Priority [in]
the priority of the thread to set (see Windows function `SetThreadPriority`, `THREAD_PRIORITY_xxxx`)

Description
Change the thread priority of the internal callback thread. The default priority of the internal callback thread is `THREAD_PRIORITY_HIGHEST`.

3.4. GetNBuffers

```
int GetNBuffers()
```

Return value
The number of subunit Data Buffers that make up the Closed-Loop Buffer.

3.5. GetBufferLevel

```
double GetBufferLevel()
```

Return value
A value in the range [0 1] indicating the Closed-Loop Buffer fill level. In conjunction with the number of buffer this value can help to estimate the Total Data Delay.

3.6. GetSamplingRate

```
double GetSamplingRate()
```

Return value
`The sampling rate of the ongoing acquisition.`

3.7. GetMVOffset

```
double GetMVOffset()
```

3 Brain Tel +41 81 3227086 · Fax +41 81 3227087 · info@3brain.com · www.3brain.com

3 Brain
Tel +41 813227086 · Fax +41 813227087 · info@3brain.com · www.3brain.com

Return value
The microvolt offset used for digital to analog conversion according to the formula:
analogValue = MVOffset + digitalValue * ADCCountsToMV.

3.8. GetADCCountsToMV

```
double GetADCCountsToMV()
```

Return value
The conversion factor used for digital to analog conversion according to the formula:
analogValue = MVOffset + digitalValue * ADCCountsToMV.

3.9. GetArrayNRows

```
int GetArrayNRows()
```

Return value
The number of rows making up the BioChip Microelectrode Array.

3.10. GetArrayNCols

```
int GetArrayNCols()
```

Return value
The number of rows making up the BioChip Microelectrode Array.

3.11. GetNAcqChannels


```
int GetNAcqChannels()
```

Return value
The number of acquired channels.

Description
The number of acquired channels equals the number of array's channel when the full-array is acquired, i.e. no Region Of Interest (ROI) is set on the BrainWave acquisition settings.
Whenever instead a ROI is set the number of acquired channels will depend from that ROI and might not equal exactly the ROI, being the ROI a superset of the effective acquired channels. Refer to the GetIndexesOfAcqChannels function to know the effective acquired channels.

3.12. GetIndexesOfAcqChannels

```
int* GetIndexesOfAcqChannels()
```

Return value

A pointer to an array containing the linear indexes (referred to the BioChip array) of the current acquired channels.

Description

Each channel has an array linear index associated to it. It is possible to convert from linear index to row and column subscripts by using the function.

3.13. ChannelIndexToSubscripts

```
void ChannelIndexToSubscripts(int index, int& row, int& col)
```

Parameters

index [in]

the linear index of the channel within the BioChip array.

row [out]

the equivalent row subscript value (in the range [1 - 64]) corresponding to the specified index whether exists; otherwise, -1.

col [out]

the equivalent col subscript value (in the range [1 - 64]) corresponding to the specified index whether exists; otherwise, -1.

Description

Convert from linear index to subscript values.

3.14. ChannelSubscriptsToIndex

```
int ChannelSubscriptsToIndex(int row, int col)
```

Parameters

row [in]

the row subscript value (in the range [1 - 64]) of the channel within the BioChip array.

col [in]

the column subscript value (in the range [1 - 64]) of the channel within the BioChip array.

Return value

The linear index equivalents to the row and column subscript values whether exists; otherwise, -1.

Description

Convert from subscript values to linear index.

3 Brain Tel +41 8132270 86 · Fax +41 8132270 87 · info@3brain.com · www.3brain.com

3 Brain
Tel +41 813227086 · Fax +41 813227087 · info@3brain.com · www.3brain.com

3.15. GetAcqFramePositionOfChannel

```
a) int GetAcqIndexOfArrayChannel(int index)
```

Parameters (a)

index [in]

the linear index of the channel within the BioChip array.

```
b) int GetAcqIndexOfArrayChannel(int row, int col)
```

Parameters (b)

row [in]

the row subscript value (in the range [1 - 64]) of the channel within the BioChip array.

col [in]

the column subscript value (in the range [1 - 64]) of the channel within the BioChip array.

Return value (a, b)

The position within the acquired channels of the specified array channel whether the channel is acquired; otherwise, .1.

Description (a, b)

Each Data Frame within a Data Buffer is made up by a sequence of pair of bytes, or in other terms by a sequence of `Int16`, which code the channel digital voltage. This function return the position of the `Int16` within a Data Frame.