COHERENCE®

Coherence5LE.dll
Version 7.13.1941.

EEG 3 files manager
SPECIAL limited edition

TABLE OF CONTENTS

INTRODUCTION

OVERVIEW

This documentation presents the functions of **Coherence5LE.dll** version 7.1.

DATA FILES

COHERENCE5LE allows reading COHERENCE® EEG data files (and writing markers).

Usually, EEG data files are located in the **\eeg2** directory.
They have the extension: *.**eeg**.

SYSTEM

COHERENCE5LE version 7.1 is a library running under Windows® XP or Windows® 7.

DEFINITIONS

All functions are described in this documentation with C++ language conventions.

This table presents all the definitions used in the following pages :

Denomination	*Type*	*Denomination*	*type*
IntegerS8	8 bits signed integer	integerU8	8 bits unsigned integer
IntegerS16	16 bits signed integer	integerU16	16 bits unsigned integer
IntegerS32	32 bits signed integer	integerU32	32 bits unsigned integer

Some functions detailed in the following pages use zero-terminated strings.
Structure types are defined in chapter '*Structures*' on page 29.

CHANGE LOG

- June 2011: Version 6.1: Support of the new file version with up to 1024 channels + adding the functions Eeg3_GETNEXTMARKER and Eeg3_TempFolderSwitch.
- November 2011: Version 7.0: Support of the long markers and real time markers.
- March 2012: Version 7.1: Improvement of the license system of Coherence5Le.
- July 2012: Version 7.1: added the possibility to anonymize files.

WARNING

In order to be able to read the new EEG file format of Deltamed, the structures used by Coherence5LE.dll have been modified.
This modification consist of expending the size of the tables of those structures from MAXELEC=128 to MAXELEC=1024.
Make sure to modify your own structures as well when using versions of Coherence5LE.dll made after the Version 6.1.0.0. Otherwise, the DLL will not work as intended, and will probably cause bugs.

USE OF THE LIBRARY

RESTRICTIONS

The library can open and manage only one EEG file at a time.
The library must be unlocked before opening a file.

VERSION

The version of the library can be retrieved through the **Eeg3_Version** function at any time.

BEGINNING

One **must** initialize the library before being able to open and read data from EEG files.
Load the library, then call the **Eeg3_Initialisation** function in order to initialize the DLL and to allocate the needed memory space.
After the initialization, **Eeg3_Unlock** must be called with the appropriate structure as a parameter.

END

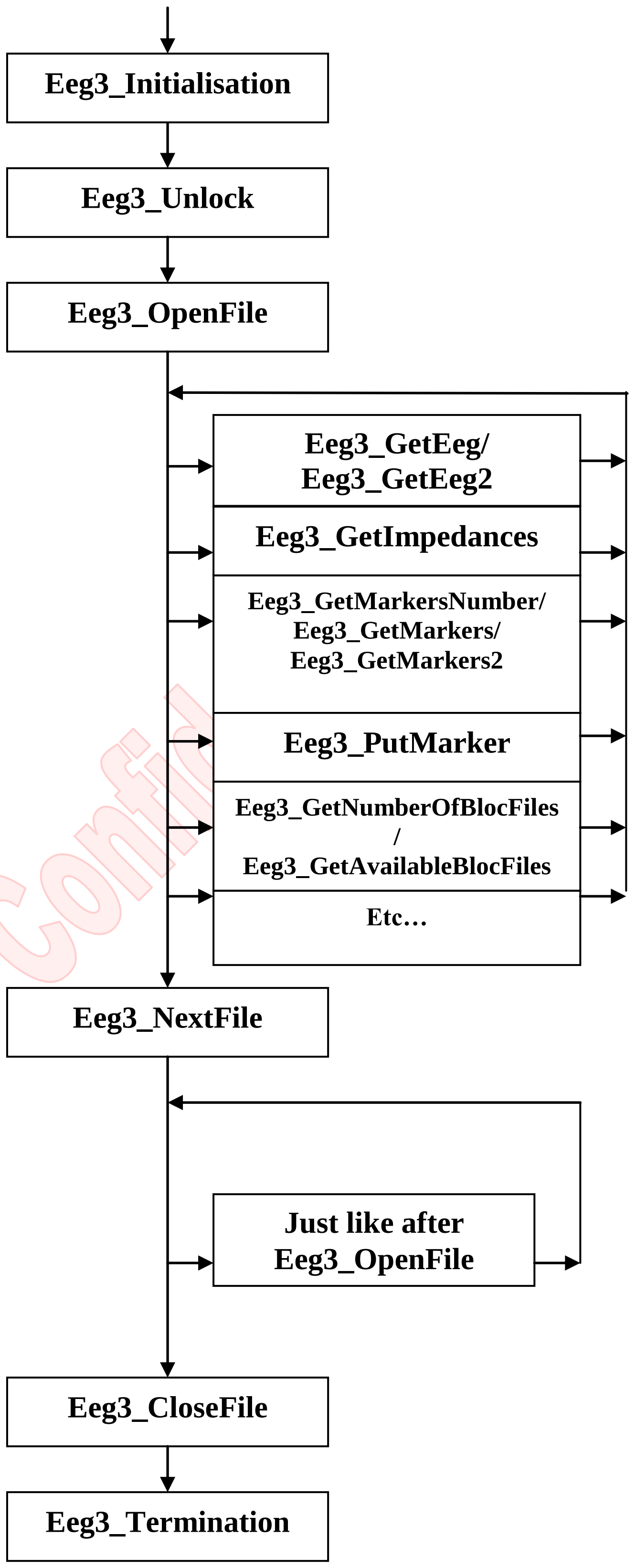
One **must** close the file before releasing the library.
Call the **Eeg3_Termination** in order to terminate the DLL and free the allocated memory space.

READ EEG FILES

The following functions can be called if the library is properly initialized.
This means, they must be called only **after initializing** and **before terminating** the DLL.

index	*Description*	*Function*
3	Open an EEG file	**Eeg3_OpenFile**
6	Get data from the EEG file (collecting them using a handle)	**Eeg3_GetEeg**
8	Get markers from the EEG file (collecting them using a handle)	**Eeg3_GetMarkers**
9	Get impedances from the EEG file	**Eeg3_GetImpedances**
7	Insert a marker in the EEG file	**Eeg3_PutMarker**
5	Close the EEG file	**Eeg3_CloseFile**
10	Unlocks the dll (no dongle needed)	**Eeg3_Unlock**
11	Activate/Desactivate the creation of the debug file	**Eeg3_DebugFileSwitch**
12	Open the next/previous file in case of bloc files	**Eeg3_NextFile**
13	Get data from the EEG file (collecting them using a pointer to a buffer of short)	**Eeg3_GetEeg2**
14	Get markers from the EEG file (collecting them using a pointer to a buffer of TMarker)	**Eeg3_GetMarkers2**
15	Get the number of markers in an interval	**Eeg3_GetMarkersNumber**
16	Get the patient related information	**Eeg3_GetPatientInfo**
17	Get the number of blocs in the recording	**Eeg3_GetNumberOfBlocFiles**
18	Get the list of all available blocs in the recording	**Eeg3_GetAvailableBlocFiles**
20	Get the first marker from the EEG file after a given sample	**Eeg3_GetNextMarker**
22	Same as Eeg3_GetMarkers, but with long text markers	**Eeg3_GetMarkersLong**
23	Same as Eeg3_GetMarkers2, but with long text markers	**Eeg3_GetMarkersLong2**
24	Same as Eeg3_GetNextMarker, but with long text markers	**Eeg3_GetNextMarkerLong**
25	Get Real time markers from the EEG file (collecting them using a handle)	**Eeg3_GetRealTimeMarkers**
26	Get the number of real time markers in the file	**Eeg3_GetNumberOfRealTimeMarkers**
27	Get real time markers from the EEG file (collecting them using a pointer to a buffer of TRealTimeMarker)	**Eeg3_GetRealTimeMarkers2**
29	Possibility to anonymize the EEG file	**Eeg3_AnonymizeFile**

The following diagram shows in which order the functions shall be called:


Eeg3_Initialisation
Eeg3_Unlock
Eeg3_OpenFile
Eeg3_GetEeg/ Eeg3_GetEeg2
Eeg3_GetImpedances
Eeg3_GetMarkersNumber/ Eeg3_GetMarkers/ Eeg3_GetMarkers2
Eeg3_PutMarker
Eeg3_GetNumberOfBlocFiles / Eeg3_GetAvailableBlocFiles
Etc…
Eeg3_NextFile
Just like after Eeg3_OpenFile
Eeg3_CloseFile
Eeg3_Termination

FUNCTIONS

EEG3_INITIALISATION

Syntax : integerS32 **Eeg3_Initialisation** (void)

Parameters :
in :
none

out :
0 if success,
<0 if an error occurred.

Comments :

The function **must** be called first.
If an error occurred, it is not possible to use the library.

EEG3_TERMINATION

Syntax : integerS32 **Eeg3_Termination** (void)

Parameters :
in :
none

out :
always returns 0.

Comments :

The function **must** be called last.

EEG3_UNLOCK

Syntax : integerS32 **Eeg3_Unlock**(TUnlock3LE Unlock3LE)

Parameters :
in :

Unlock3LE :	Structure containing 4 confidential integers to unlock the library. The structure is detailed in the chapter called 'Structures'.

out :

0	If the library was successfully unlocked
-1	If at least one of the 4 integers doesn't match the expected value.

Comments :

The function **must be called right after Eeg3_Initialisation**.

If one of the integers in Unlock3LE is wrong, it is not possible to use the library.

The 4 integers needed to unlock the library can be obtained only after signing a non-disclosure agreement. These integers are specific to a given partner.

Starting from version 7.1, it is possible to install a license for Coherence5Le.dll using the LicenceTool.exe software. Once the license is installed for a specific computer, the DLL can be unlocked by calling the Eeg3_Unlock function with all four integers equal to zero.

EEG3_OPENFILE

Syntax : integerS32 **Eeg3_OpenFile** (char *FileName, TCoh3 *aCoh3)

Parameters :
in :

FileName :	file name of an EEG file (pointer on a zero-terminated string).
aCoh3 :	pointer to a TCoh3 structure.

out :

integerS32 :	0 if success, <0 if an error occurred.
aCoh3 :	information about the record

Comments :

If the function succeeds, the **TCoh3** structure is filled with information about the EEG record. Items like number of electrodes, sample rate… are returned.

The *FileName* parameter needs full path and file name.
Example : *C:\EEG2\SAMPLE.EEG*

EEG3_CLOSEFILE

Syntax : integerS32 **Eeg3_CloseFile** (void)

Parameters :
in :
none

out :
0 if success,
<0 if an error occurred.

Comments :

Close the opened file.
Eeg3_GetEeg, **Eeg3_GetMarkers, Eeg3_PutMarker** and **Eeg3_GetImpedances** functions can not be called until the next call to **Eeg3_OpenFile**.

EEG3_GETEEG

Syntax : integerS32 **Eeg3_GetEeg** (integerS32 begin, integerS32 duration, HGLOBAL Hbuf)

Parameters :
in :

begin :	first sample to read (starts at 0)
duration :	number of samples to be read (>0)
Hbuf :	Handle to a buffer that will receive data

out :

integerS32 :	>0 if success, <0 if an error occurred
Hbuf :	contains EEG data if success

Comments :

If success, function returns the length of returned data (should be equal to the *duration* parameter) in samples.
If success, data are stored (16 bits signed integers) in the buffer in the following order :

```
sample 1 channel 1, sample 1 channel 2 … sample 1 channel n,
sample 2 channel 1, sample 2 channel 2 … sample 2 channel n,
…
sample m channel 1, sample m channel 2 … sample m channel n.
```

If the function returns the error code « -106 : End of the file », you are arrived at the end of the file, and should call « Eeg3_NextFile ».

<u>Warning</u> :

1- The buffer size is not checked. Make sure it is large enough !
2- For compressed data, as the baseline is not stored but reconstructed, the offset of the data can slightly vary from one run to another. This has no consequence as EEG has no DC component.

EEG3_GETEEG2

Syntax : integerS32 **Eeg3_GetEeg2** (integerS32 begin, integerS32 duration, short* PBuf)

Parameters :
in :

begin :	first sample to read (starts at 0)
duration :	number of samples to be read (>0)
PBuf :	Pointer to a buffer of short that will receive data

out :

integerS32 :	>0 if success, <0 if an error occurred
PBuf :	contains EEG data if success

Comments :

If success, function returns the length of returned data (should be equal to the *duration* parameter) in samples.
If success, data are stored (16 bits signed shorts) in the buffer in the following order :

```
sample 1 channel 1, sample 1 channel 2 … sample 1 channel n,
sample 2 channel 1, sample 2 channel 2 … sample 2 channel n,
…
sample m channel 1, sample m channel 2 … sample m channel n.
```

If the function returns the error code « -106 : End of the file », you are arrived at the end of the file, and should call « Eeg3_NextFile ».

Warning :

1- The buffer size is not checked. Make sure it is large enough !
2- For compressed data, as the baseline is not stored but reconstructed, the offset of the data can slightly vary from one run to another. This has no consequence as EEG has no DC component.

EEG3_GETMARKERS

Syntax : integerS32 **Eeg3_GetMarkers** (integerS32 begin, integerS32 end, HGLOBAL *Hbuf)

Parameters :

in :

begin :	read markers from *begin* (in samples)…
end :	… to *end* (in samples)
**Hbuf* :	Pointer to a handle to a buffer that will receive markers

out :

integerS32 :	return the number of markers collected if success, <0 if an error occurred
**Hbuf* :	contains marker data if success

Comments :

If success, the function returns a handle to a buffer containing a list of markers.
Each marker is a **TMarker** structure.
If success, the return value of the function is the number of markers collected.

<u>The buffer :</u>

The buffer is created with a **GlobalAlloc** Windows API function with GMEM_MOVEABLE and GMEM_ZEROINIT allocation attributes set.

DO NOT FREE this buffer with a **GlobalFree()** function for instance.
It will automatically be freed when the next **Eeg3_GetMarkers[Long][2]** or **Eeg3_Terminate** function is called.

EEG3_GETMARKERS2

Syntax : integerS32 **Eeg3_GetMarkers2** (integerS32 begin, integerS32 end, TMarker *PMrk)

Parameters :
in :

begin :	read markers from *begin* (in samples)…
end :	… to *end* (in samples)
**PMrk* :	Pointer to a buffer of **TMarker** that will receive markers

out :

integerS32 :	return the number of markers collected if success, <0 if an error occurred
**PMrk*:	contains marker data if success

Comments :

If success, the function returns a handle to a buffer containing a list of markers.
Each marker is a **TMarker** structure.
If succes, the return value of the fonction is the number of markers collected.

<u>The buffer :</u>

The size of the PMrk buffer is not checked. Make sure that the buffer is large enough to save a high number of markers !
You can also use **Eeg3_GetMarkersNumber** to predict the number of marker between *begin* and *end*, and adapt the size of the buffer

EEG3_GETMARKERSLONG

Syntax : integerS32 **Eeg3_GetMarkersLong** (integerS32 begin, integerS32 end, HGLOBAL *Hbuf)

Parameters :
in :

begin :	read markers from *begin* (in samples)…
end :	… to *end* (in samples)
**Hbuf* :	Pointer to a handle to a buffer that will receive markers

out :

integerS32 :	return the number of markers collected if success, <0 if an error occurred
**Hbuf* :	contains marker data if success

Comments :

If success, the function returns a handle to a buffer containing a list of markers.
Each marker is a **TMarkerLong** structure.
If success, the return value of the function is the number of markers collected.

<u>The buffer :</u>

The buffer is created with a **GlobalAlloc** Windows API function with GMEM_MOVEABLE and GMEM_ZEROINIT allocation attributes set.

DO NOT FREE this buffer with a **GlobalFree()** function for instance.
It will automatically be freed when the next **Eeg3_GetMarkers[Long][2]** or **Eeg3_Terminate** function is called.

EEG3_GETMARKERSLONG2

Syntax : integerS32 **Eeg3_GetMarkers2** (integerS32 begin, integerS32 end, TMarkerLong *PMrk)

Parameters :

in :

begin :	read markers from *begin* (in samples)…
end :	… to *end* (in samples)
**PMrk* :	Pointer to a buffer of **TMarkerLong** that will receive markers

out :

integerS32 :	return the number of markers collected if success, <0 if an error occurred
**PMrk*:	contains marker data if success

Comments :

If success, the function returns a handle to a buffer containing a list of markers.
Each marker is a **TMarkerLong** structure.
If success, the return value of the function is the number of markers collected.

<u>The buffer :</u>

The size of the PMrk buffer is not checked. Make sure that the buffer is large enough to save a high number of markers!
You can also use **Eeg3_GetMarkersNumber** to predict the number of marker between *begin* and *end*, and adapt the size of the buffer

EEG3_GETIMPEDANCES

Syntax : integerS32 **Eeg3_GetImpedances** (integerS32 startpos, TImpedance *imped)

Parameters :
in :

startpos :	start position of the backward search (in samples)
imped :	pointer to an impedance test structure

out :

integerS32 :	>0 if success, <0 if an error occurred
imped :	Impedance test structure is filled if success

Comments :

The function looks for the last impedance test from the *startpos* position going backwards toward the first sample.
The first impedance test found during the backward search is returned by the function.

If success, **Eeg3_GetImpedances()** fills the impedance structure passed as parameter.

The field contains impedance values (0 to 250) in the electrode set order.

EEG3_VERSION

Syntax : integerS32 **Eeg3_Version** (TVersion *version)

Parameters :
in :

version :	pointer to a TVersion structure

out :

integerS32 :	always returns 0.
version :	if success, the structure is filled.

EEG3_PUTMARKER

Syntax : integerS32 **Eeg3_PutMarker**(TMarker *evtle)

Parameters :
in :

evtle :	pointer to a TMarker structure

out :

integerS32 :	>0 if success, <0 if an error occurred

EEG3_DEBUGFILESWITCH

Syntax : integerS32 **Eeg3_DebugFileSwitch**(bool createDebugFile)

Parameters :
in :

createDebugFile:	if true, the debug file will be created. If false, it will not.

out :

integerS32 :	>0 if success, <0 if an error occurred

Warnings :

If you want to create the debug file, you should call **Eeg3_DebugFileSwitch**(TRUE) just before **Eeg3_Initialisation**, otherwise, the debug file will not be created.

EEG3_GETMARKERSNUMBER

Syntax : integerS32 **Eeg3_GetMarkersNumber** (integerS32 begin, integerS32 end)

Parameters :
in :

begin :	read markers from *begin* (in samples)…
end :	… to *end* (in samples)

out :
integerS32 : Return the number of markers between *begin* and *end* if success, <0 if an error occurred

EEG3_NEXTFILE

Syntax : integerS32 **Eeg3_NextFile** (integerS32 direction, char *FileName, TCoh3 *aCoh3)

Parameters :
in :

direction :	signed integer for the direction : if >0, the function will open the next file, if <0, the function will open the prévious file
FileName :	pointer to a zero-terminated string
aCoh3 :	pointer to a TCoh3 structure.

out :

integerS32 :	0 if success, <0 if an error occurred.
FileName :	If success, this string will contain the path to the opened file
aCoh3 :	information about the record

Comments :

If the function succeeds, the **TCoh3** structure is filled with information about the previous/next file EEG record. Items like number of electrodes, sample rate… are returned. The **FileName** string will be filled with the name and the path of the opened file (Example : *C:\EEG2\SAMPLE.EEG);*

Warnings :

The size of the **FileName** string is not checked. **It should be 260 char long**, in order to store the file name and path.

EEG3_GETPATIENTINFO

Syntax : integerS32 **Eeg3_GetPatientInfo**(TPatientInfo3LE *infopat);

Parameters :
in :

infopat :	Pointer to a TpatientInfo3LE structure.

out :

integerS32 :	0 if success, <0 if an error occurred.
infopat :	If success, this structure will contain all the information related to the patient found in the recording.

Comments :

If the function succeeds, the **TPatientInfo** structure is filled with information about the patient. Items like name, date of birth of sex are returned.

Warnings :

The recording should have been opened with **Eeg3_OpenFile** before calling this function.

EEG3_GETNUMBEROFBLOCFILES

Syntax : integerS32 **Eeg3_GetNumberOfBlocFiles**(char *FileName);

Parameters :
in :

FileName :	File name of an EEG file (pointer on a zero-terminated string).

out :

integerS32 :	Contain the number of bloc in the recording in case of success, <0 if an error occurred, or if the recording is not a bloc file (one bloc).

Warnings :

The file name (**FileName**) should be the path to a bloc recording (Example : *C:\EEG2\SAMPLE_0017.EEG);*
If the function return the error code "-1208", then the file is not a bloc file (only one bloc).

EEG3_GETAVAILABLEBLOCFILES

Syntax : integerS32 **Eeg3_GetAvailableBlocFiles**(char* FileName, String* BlocFiles);

Parameters :
in :

FileName :	File name of an EEG file (pointer on a zero-terminated string).
BlocFiles :	Pointer to a table of strings, which will be filled with the list of the bloc files of the recording.

out :

integerS32 :	0 if success, <0 if an error occurred.
BlocFiles :	If success, this structure will contain a list of zero-terminated Strings, each String being the complete name of one of the bloc of the recording.

Warnings :

Make sure that **BlocFiles** can be filled with a number of zero-terminated Strings equal to the number of blocs in the recording. This number of bloc can be found using **Eeg3_ GetNumberOfBlocFiles.**
If the function return the error code "-1208", then the file is not a bloc file (only one bloc).

EEG3_GETNEXTMARKER

Syntax : integerS32 **Eeg3_GetNextMarker**(integerS32 start, TMarker *evtle);

Parameters :
in :

start :	The function will look for the next event, starting from the 'start' sample (start sample included).
evtle :	Pointer to a **TMarker** structure. The structure will be filled with the event / marker returned.

out :

integerS32 :	0 if success, <0 if an error occurred **(-21 means that there is no next marker / event in the file).**
evtle :	Pointer to **TMarker** structure. The structure will be filled with the event / marker returned.

EEG3_GETNEXTMARKERLONG

Syntax : integerS32 **Eeg3_GetNextMarkerLong**(integerS32 start, TMarkerLong *evtle);

Parameters :
in :

start :	The function will look for the next event, starting from the 'start' sample (start sample included).
evtle :	Pointer to a **TMarkerLong** structure. The structure will be filled with the event / marker returned.

out :

integerS32 :	0 if success, <0 if an error occurred **(-21 means that there is no next marker / event in the file).**
evtle :	Pointer to **TMarkerLong** structure. The structure will be filled with the event / marker returned.

EEG3_TEMPFOLDERSWITCH

Syntax : integerS32 **Eeg3_TempFolderSwitch(bool CreateTmpFolder)**

Parameters :
in :
CreateTmpFolder: if true, the temp folder will be created in the same folder than the EEG file opened by the DLL. If false, the temp folder used will be the one of windows ("C:\Documents and Settings\Username\Local Settings\Temp" for Windows XP).

out :
integerS32 : >0 if success, <0 if an error occurred

Warnings :

For it to be taken into account, this function must be called before **Eeg3_Initialisation.** If this function is not called, the temp folder will be created in the same folder than the EEG file opened by the DLL.

EEG3_GETNUMBEROFREALTIMEMARKERS

Syntax : integerS32 **Eeg3_GetNumberOfRealTimeMarkers** (void)

Parameters :
in :
void

out :
integerS32 : Return the number of real time markers in the EEG file (bloc file) currently opened, <0 if an error occurred

EEG3_GETREALTIMEMARKERS

Syntax : integerS32 **Eeg3_GetRealTimeMarkers** (HGLOBAL *Hbuf)

Parameters :
in :

**Hbuf* :	Pointer to a handle to a buffer that will receive real time markers

out :

integerS32 :	return the number of real time markers collected if success, <0 if an error occurred
**Hbuf* :	contains real time marker data if success

Comments :

If success, the function returns a handle to a buffer containing a list of real time markers.
Each marker is a **TRealTimeMarker** structure.
If success, the return value of the function is the number of real time markers collected.

<u>The buffer :</u>

The buffer is created with a **GlobalAlloc** Windows API function with GMEM_MOVEABLE and GMEM_ZEROINIT allocation attributes set.

DO NOT FREE this buffer with a **GlobalFree()** function for instance.
It will automatically be freed when the next **Eeg3_GetRealTimeMarkers[2]** or **Eeg3_Terminate** function is called.

EEG3_GETREALTIMEMARKERS2

Syntax : integerS32 **Eeg3_ GetRealTimeMarkers2** (TRealTimeMarker *PMrk)

Parameters :
in :
PMrk* : Pointer to a buffer of **TRealTimeMarker that will receive real time markers

out :
integerS32 : return the number of real time markers collected if success, <0 if an error occurred
**PMrk*: contains real time marker data if success

Comments :

If success, the function returns a handle to a buffer containing a list of real time markers.
Each marker is a **TRealTimeMarker** structure.
If success, the return value of the function is the number of real time markers collected.

The buffer :

The size of the PMrk buffer is not checked. Make sure that the buffer is large enough to save a high number of real time markers!
You can also use **Eeg3_GetNumberOfRealTimeMarkers** to predict the number of real time markers between in the file, and adapt the size of the buffer

EEG3_ANONYMIZEFILE

Syntax : integerS32 **Eeg3_ AnonymizeFile** (integerS32 AnonymizeMode)

Parameters :
in :
AnonymizeMode: The mode of anonymization of the file :
0: Keep only the first 3 letters of the name/surname
1: Keep only the first letter of the name/surname
2 : Delete completely the name/surname

out :
integerS32 : <0 if an error occurred

STRUCTURES

UNLOCK

```
typedef struct
{
  int int1;            // first integer
  int int2;            // second integer
  int int3;            // third integer
  int int4;            // fourth integer
} TUnlock3LE;
```

Confidential integers needed to unlock the library can be obtained after the signature of a non-disclosure agreement.

LIBRARY VERSION

```
typedef struct
{
       integerS16
              major,                    // major
              minor,                    // minor version
              compile,                  // compilation
              number;                   // number
} TVersion;
```

MARKER STRUCTURE

```
typedef struct
{
      integerS32
            evttype,                // type of the marker
            pos,                    // position
            duration;               // duration
      char
            text[80];               // comment
} TMarker;
```

Type of markers

evttype contains the specification of the marker :

0 for an annotation
1 for an external marker.

The structure contains position (*pos*) and comment (*text*) of the marker.
The *duration* unit is one sample.

IMPEDANCE TEST STRUCTURE

```
typedef struct
{
      char
            text[20];               // text associated to the impedance test
      intergerS32
            pos,                    // position of the impedance test
            nbchannel,              // number of channels used in the impedance test
            imped[MAXELEC];         // impedances values (in Kohm, range : 0 to 250)
} TImpedances;
```

The value of MAXELEC is currently 1024, and not 128 anymore. Make sure to update your structure.

RECORD INFORMATION

```
typedef struct
{
      integerS32
            duration,               // duration of the record in seconds
            frequency,              // sampling rate
            timebase,               // sampling time base in seconds
            electrodes;             // number of electrodes
      char
            date[20],               // date and time of the record ('hh:mm:ss dd/mm/yyyy')
            name[MAXELEC][8], // electrode names
            type[MAXELEC];    // electrode types
      integerS32
            theta[MAXELEC],   // theta angular coordinate
            phi[MAXELEC],     // phi angular coordinate
            r[MAXELEC],       // radius
            minanal[MAXELEC],  // min analog value
            maxanal[MAXELEC], // max analog value
            minconv[MAXELEC], // min value of the converter
            maxconv[MAXELEC] ;          // max value of the converter
      char
            unit[MAXELEC][4]; // display unit ('µV', 'mmHg'…)
} TCoh3;
```

Frequency

Sampling rate is the same for all channels.

Indexes

All MAXELEC's arrays are filled from the first index to the value stored in the *electrodes* field. **The value of MAXELEC is currently 1024, and not 128 anymore. Make sure to update your structure.**

Electrode types

Values allowed for electrode types are :

0=EEG, 1=polygraphic AC channel, 2=DC channel, 3=photic, 4=depth electrode

Coordinates

Theta and Phi coordinates are defined in grades.
Range : Theta –200 to +200 Phi -100 to +100
Radius is not used.

Sensitivity

Sensitivities for each electrode are stored into 4 fields : minconv, maxconv, minanal, maxanal.

Sensitivity (in µV) for 1000 ADC bits is :

(maxanal-minanal) / (maxconv-minconv)

Two examples :

maxanal = 125
minanal = 0
maxconv = 1000
minconv = 0
 	
sample (16 bits signed short integer) = 1500
 	
Amplitude (in µV) = 1500 * (125-0)/(1000-0)
Amplitude (in µV) = 187.5 µV
	maxanal = 179
	minanal = 0
	maxconv = 1000
	minconv = 0
	
	sample (16 bits signed short integer) = 300
	
	Amplitude (in µV) = 300 * (179-0)/(1000-0)
	Amplitude (in µV) = 53.7 µV

Date

The date field is a zero-terminated text string as follow :

‘*hh:mm:ss dd/MM/yyyy*’.
means ‘*hour:minutes:seconds days/month/year*’.

PATIENT STRUCTURE

```
typedef struct
{
     char
            name[50],             // patient name
            firstname[30],        // patient first name
            date[11],             // patient date of birth
            sex,                  // patient sex (M/F)
            file[20],             // file number of recording
            center[39],           // origin of the recording
            comments[256];        // commentary
}TPatientInfo3LE;
```

Each information about the patient is a zero terminated string, except the sex, which is one character (M for male of F for female).

LONG MARKER STRUCTURE

```
typedef struct
{
     integerS32
            evttype,              // type of the marker
            pos,                  // position
            duration;             // duration
     char
            text[252];            // comment
} TMarkerLong;
```

Type of markers

evttype contains the specification of the marker :

0 for an annotation
1 for an external marker.

The structure contains position (*pos*) and comment (*text*) of the marker.
The *duration* unit is one sample.

REAL TIME MARKER STRUCTURE

```
typedef struct
{
      integerS32
                 pos,              //Position of the real time marker in sample
                 realtime;         //Absolute real time at the specified position, in second
}TRealTimeMarker;
```

This information makes it possible to calculate the absolute real time at any sample of the EEG file (bloc file) with a precision of one second, even if there are discontinuities in the recording, thanks to the following formula:

```
(Absolute real time at sample X in second) =
      PreviousRealTimeMarker.realTime
       +
       (SampleX - PreviousRealTimeMarker.pos) / Coh3.frequency.
```

Appendix

LIST OF ALL EXPORTED FUNCTIONS

Hereafter all exported functions of the library with their index.

```
integerS32   Eeg3_Initialisation (void);                                                // 1
integerS32   Eeg3_Termination (void);                                                   // 2
integerS32   Eeg3_Version (TVersion *version);                                          // 3
integerS32   Eeg3_OpenFile (char *FileName, TCoh3 *aCoh3);                              // 4
integerS32   Eeg3_CloseFile (void);                                                     // 5
integerS32   Eeg3_GetEeg (integerS32 begin, integerS32 duration, HGLOBAL Hbuf);         // 6
integerS32   Eeg3_PutMarker (TMarker *evtle);                                           // 7
integerS32   Eeg3_GetMarkers (integerS32 begin, integerS32 end, HGLOBAL *Hbuf);         // 8
integerS32   Eeg3_GetImpedances (integerS32 pos, TImpedances3 *imped);                  // 9
integerS32   Eeg3_Unlock(TUnlock3LE Unlock3LE);                                         // 10
integerS32   Eeg3_DebugFileSwitch(bool CreateDbgFile);                                  // 11
integerS32   Eeg3_NextFile (integerS32 direction, char *fileName, TCoh3Le *acoh3le);    // 12
integerS32   Eeg3_GetEeg2 (integerS32 debut, integerS32 duree, short *PBuf);            // 13
integerS32   Eeg3_GetMarkers2(integerS32 begin, integerS32 end, Tmarker *Pevt);         // 14
integerS32   Eeg3_GetMarkersNumber(integerS32 begin, integerS32 end);                   // 15
integerS32   Eeg3_GetPatientInfo(TPatientInfo3LE *infopat);                             // 16
integerS32   Eeg3_GetNumberOfBlocFiles(char *FileName);                                 // 17
integerS32   Eeg3_GetAvailableBlocFiles(char* FileName, String* BlocFiles);             // 18
integerS32   Eeg3_ModeDoubleEventFile(bool mode);                                        // 19
integerS32   Eeg3_GetNextMarker(integerS32 debut,TMarker *evtle);                        // 20
integerS32   Eeg3_TempFolderSwitch(bool CreateTmpFolder);                                // 21
integerS32   Eeg3_GetMarkersLong(integerS32 debut, integerS32 fin, HGLOBAL *Hbuf);  // 22
integerS32   Eeg3_GetMarkersLong2(integerS32 debut, integerS32 fin, TMarkerLong *evtle); //23
integerS32   Eeg3_GetNextMarkerLong(integerS32 debut,TMarkerLong *evtle);               // 24
integerS32   Eeg3_GetRealTimeMarkers(HGLOBAL *HrealTimeMrk);                            // 25
integerS32   Eeg3_GetNumberOfRealTimeMarkers(void);                                     // 26
integerS32   Eeg3_GetRealTimeMarkers2(TRealTimeMarker *realTimeMrk);                    // 27
```

ERROR CODES

The following list presents error codes returned by the library :

-100	open file for reading error
-101	open file for writing error
-102	file doen't exist or not opened
-103	reading from the file
-104	writing to the file
-105	moving in the file
-106	end of file
-109	read only file
-110	wrong path
-200 to -299	memory management
-400 to -499	library use not allowed
-1100 to -1999	EEG file management
-1203	wrong data file version
-1204	this is not an EEG 3 data file
-1207	the duration of the record is 0
-1211	this file is being acquired, opening it is not allowed with that dongle contents
-1300	get EEG data : values for the starting point and/or duration are out of range
-2000 to -2999	event management
-2000	no event found
-2001	position is out of the record limits
-5000	EEG data file is acquired with more than 48 electrodes. Review is not allowed with that dongle contents.