LXDeviceAPI

LXDeviceAPI 개발자 메뉴얼

Doc. ID. LXD64 V1

Release Date. 2017-04-25 .

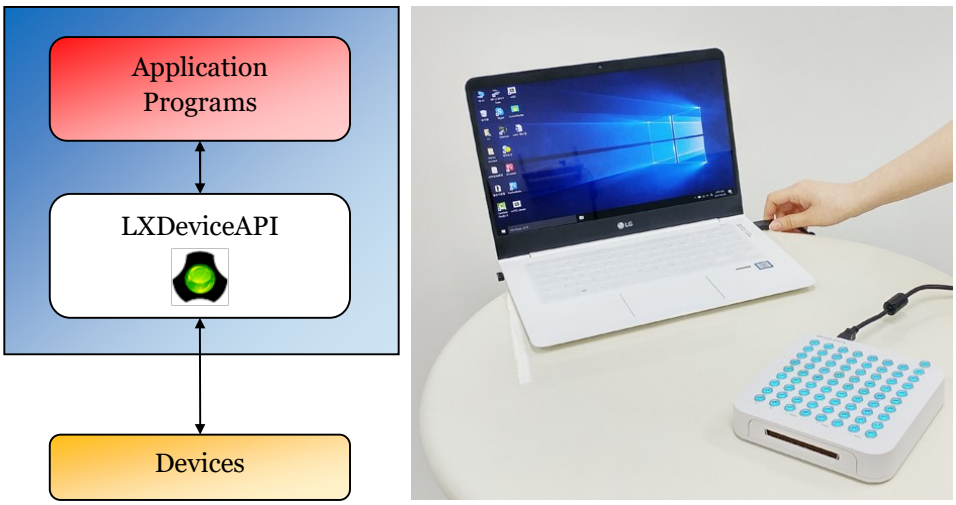
Abstract – 생체신호 측정기기 연동 API(Application Programming Interface)

Site for LXDeviceAPI : http://laxtha.net/ko/lxdeviceapi/

목차

LXDeviceAPI 개요

응용프로그램과 생체신호측정기기 연동 API.



System Architecture

LXDeviceAPI 특징.

기기에서 측정된 생체신호의 고속실시간 스트리밍, 기기제어.

표준 C 함수 라이브러리와 메시지 기반 처리.

API 내장 구현된 UI 이용하여, 해당 기능 추가 코딩 작업 없이 즉시 사용가능.

- 신호 필터 설정.
- 전극-피부 임피던스 모니터링.
- 기기의 자동보정 처리.
- API 소프트웨어 원격 업데이트 .
- 기기 펌웨어 원격 업데이트 .
- 설정 정보의 기기저장.

API 형식 : Regular DLL - 표준 C 함수와 메시지 기반 처리.

API 제작툴: Visual C++ 2015 , MFC Regular DLL .

지원 플랫폼 : 32bit , 64bit 모두 지원.

지원 운영체제 : 윈도우 10, 8.1, 8, 7

지원 기기 : QEEG-32FX, QEEG-64FX.

시작하기

LXDeviceAPI 압축파일 다운로드하고 Visual C++ 프로젝트에서 DLL 임포팅 하기 전체과정 상세.

단계 1. 다운로드 / 차단해제 / 압축풀기

단계 1.1 다운로드.

API 활용할 응용프로그램의 비트수와 동일한 비트수의 LXDeviceAPI 다운로드(아래 링크 클릭).

LXDeviceAPI_32bit.zip 다운로드

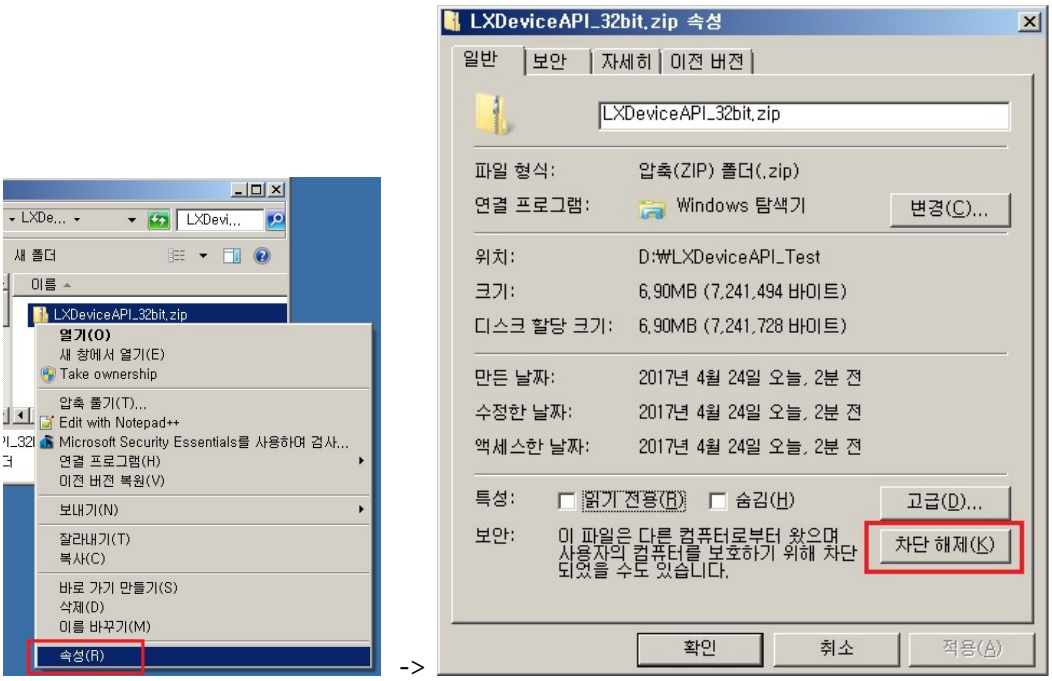
https://github.com/LAXTHA/LXDeviceAPI/raw/master/LXDeviceAPI_32bit.zip

LXDeviceAPI_64bit.zip 다운로드

https://github.com/LAXTHA/LXDeviceAPI/raw/master/LXDeviceAPI_64bit.zip

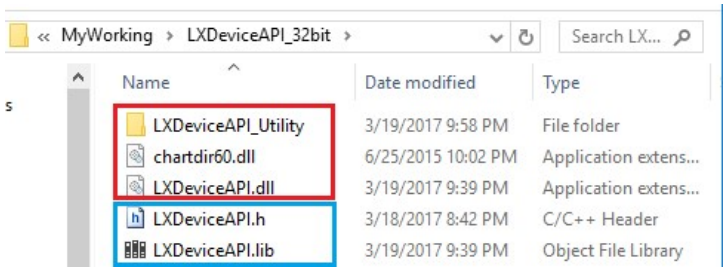
단계 1.2 차단해제.

다운로드 받은 압축파일의 속성 창에서 차단해제 하기.



단계 1.3 압축풀기.

차단해제된 압축파일 풀기.



단계 2 . 프로젝트 폴더로 API 파일들 복사하기.

파일	설명
• LXDeviceAPI_Utility (폴더) • LXDeviceAPI.DLL • chartdir60.dll	개발프로젝트의 실행 파일 경로에 복사.
• LXDeviceAPI.LIB • LXDeviceAPI.h	개발 프로젝트의 소스파일 경로에 복사.

단계 3. Visual C++ 프로젝트에서 DLL importing.

DLL 임포팅 방법 중 가장 간단한 implicit linking 하고, LXDeviceAPI.h 인클루드하기. (아래 코드)

```
#pragma comment(lib,"LXDeviceAPI.lib") // implicit linking.

#include "LXDeviceAPI.h" // for using LXDeviceAPI
```

상기 설정 으로 LXDeviceAPI 에서 제공되는 모든 함수 및 메시지를 Visual C++ 프로젝트에서 활용 가능 상태 달성.

API 스트림 데이터

스트림 구조체: ST_STREAMDATA_LXDAPI

API 는 기기로부터 실시간 측정치들을 수집함과 동시에 이들 데이터들을 응용프로그램으로 스트림 데이터 전송한다. 응용프로그램 스트림 전송시 데이터 형식은 구조체 ST_STREAMDATA_LXDAPI 가 사용되며, LXDeviceAPI.h 에 아래처럼 선언되어있다.

```
// struct for stream.
typedef struct _stStreamData_LXDAPI
{
…
        unsigned int*       Event_StreamData_CS; // array for event stream
…
        double*             Wave_StreamData_CS; //array for measured multi channel bio signal

…
}ST_STREAMDATA_LXDAPI;
```

ST_STREAMDATA_LXDAPI 의 멤버 중 가장 중요한 변수는 2 개의 1 차원 스트림배열이다.

double * Wave_StreamData_CS

unsigned int* Event_StreamData_CS

기기에서 측정된 다채널 생체신호 샘플링 데이터 들은 Wave_StreamData_CS[] 에 기록되며, 이벤트 마킹 정보는 Event_StreamData_CS[] 에 저장된다.

스트림 구조체 사용방법.

응용프로그램에서 구조체 변수 선언하고 함수 OpenDevice 호출시 인자로 구조체 변수 주소 전달한다.

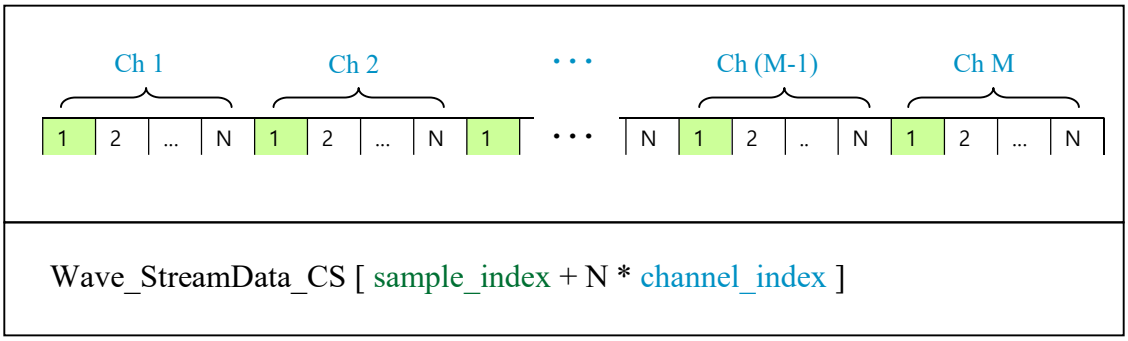
OpenDevice 호출되면 API 내부적으로 구조체 변수 동적 메모리 할당 이뤄지며, 이때 메모리 크기는 OpenDevice 의 인자 int numsample_return 에 의하여 결정된다. API 내부적으로 동적할당된 메모리는 CloseDevice 함수 호출시점에 API 내부적으로 자동 제거 처리된다.

```
ST_STREAMDATA_LXDAPI  stStreamData;    // 구조체 변수선언.
int NumSampleReturn = 32 ; // 가능한 값의 범위 : 1 ~ 128
OpenDevice_LXDeviceAPI( , &stStreamData, NumSampleReturn, );  // OpenDevice 함수 호출시 구조체변수 주소전달.
```

메모리 맵.

Wave_StreamData_CS 데이터 배치.

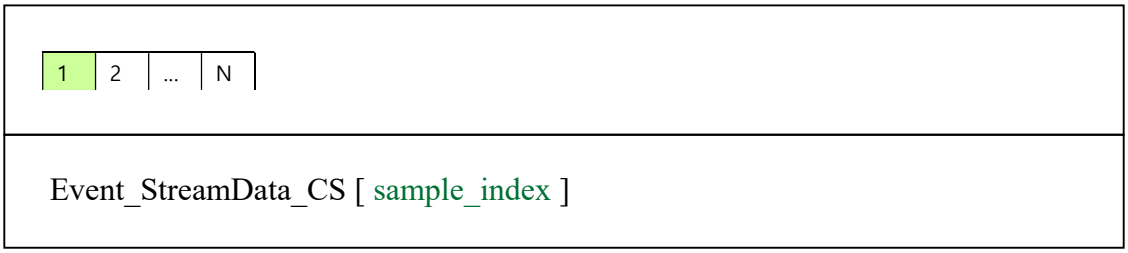
Wave Stream Data. Memory Map & Array Indexing.



- **N** : Number of Samples
- **M** : Number of Channels
- **sample_index :** index for samples. from 0 to N-1
- **channel_index** : index for channels. from 0 to M-1

Event_StreamData_CS 데이터 배치.

Event Stream Data. Memory Map & Array Indexing.



- **N** : Number of Samples
- **sample_index :** index for samples. from 0 to N-1

스트림 배열 인덱싱.

기기에서 측정된 다채널 생체신호 중 특정 채널의 1 개의 샘플 데이터 획득하는 코드예.

```
double one_sample_wave = stStreamData.Wave_StreamData_CS[sample_index + NumSampleReturn *
channel_index];
```

이벤트 데이터 1 개의 샘플 데이터 획득하는 코드예.

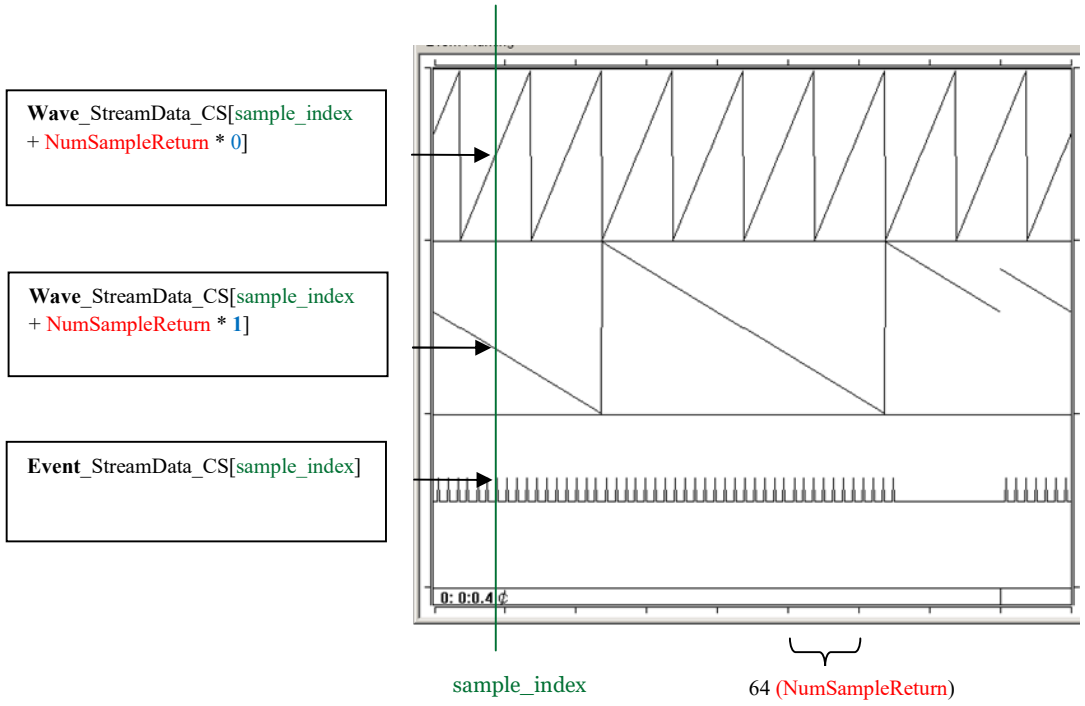
```
unsigned int one_sample_event = stStreamData.Event_StreamData_CS[sample_index];
```

channel_index 가능한 값의 범위 : 0 ~ (최대채널수-1). 최대채널수는 기기마다 다르며, QEEG-64FX (64ch)인 경우 최대 채널수 67. API 지원기기들의 최대채널수는 본 문서 Appendix 1. 에서 제공.

sample_index 가능한 값의 범위 : 0 ~ (NumSampleReturn -1). NumSampleReturn 은 OpenDevice 호출시 인자 int numsample_return 에 입력한 값으로 결정된다.

스트림 데이터 표현예.

아래 응용프로그램의 챠트 출력은 sample_index 를 X 축(시간)에 대응시키고, Wave 채널 1, 채널 2 , Event 표현한 예.



API 메시지.

API 메시지 인자 wParam

API 에서 응용프로그램 측으로 메시지 전송시 메시지 1 번 인자 wParam 에는 unsigned int(4 바이트) 형의 정수가 전달되며, wParam 의 하위 바이트부터 wParam_Byte0, wParam_Byte1, wParam_Byte2, wParam_Byte3 이라고 하였을 때, 각 바이트에는 아래표와 같은 데이터가 기록되어 있다.

wParam 의 바이트	기록된 데이터.	설명.
wParam_Byte3	장치핸들링 ID	Device Handling ID = wParam_Byte3 * 256 + wParam_Byte2. 값 범위 1~65535 API 를 이용하여 동시에 2 개 이상의 장치와 통신하는 경우, API 에서 전송된 메시지를 수신한 응용프로그램에서어떤 장치에서 전송된 메시지인지 식별 할 때 사용된다. 관련 : 장치핸들링 ID 는 OpenDevice 함수 호출시 API 내부적으로 자동생성되어 반환되는 정수 값이며, 통신 개설된 장치에 할당되는 고유한 값.
wParam_Byte2		
wParam_Byte1	메시지타입 ID.	API 에서 전송된 메시지가 어떤 "메시지타입 ID"에서 발생한 메시지인지 식별시 활용. 값범위 : MSGTYPEID0_DEVICE_LXDAPI ~ MSGTYPEID9_DEVICE_LXDAPI 관련 : SetMessageDevice 함수호출시 "메시지타입 ID" 별로 메시지 수신받을 윈도우핸들, 메시지아이디, 메시지 수신여부를 지정하게 되어있다.
wParam_Byte0	메시지타입 서브 ID.	1 개의 메시지타입 ID 에서도 여러 종류의 데이터가 메시지로 전송되며, 메시지타입서브 ID 를 확인하여 전송된 메시지가 어떤 종류의 데이터인지 식별할 때 필수로 활용된다. 값 범위 0~255.

메시지 타입별 의미.

wParam Byte 1 (Message Type ID)	wParam Byte 0 (Message Type Sub ID)	설명
MSGTYPEID0_DEVICE_LXDAPI	0	API 함수 OpenDevice 호출시 인자 int numsample_return 에 설정한 반환샘플링 수량만큼의 데이터가 모두 수집된 시점에 발생. 응용프로그램의 메시지 핸들러 내에서 GetStreamData 함수 호출해야만 구조체변수(ST_STREAMDATA_LXDAPI 타입) 데이터가 신규 갱신된다.

Code Examples

OpenDevice 함수 호출 이후 SetMessageDevice 호출하여 기기에서 발생하는 스트림 메시지 수신 설정해둔다. StartStream 호출하면 API 내부적으로 기기로부터 실시간 수집된 데이터 수량이 지정수량 (OpenDevice 함수 호출시 지정한 int numsample_return)이 된 경우 응용프로그램 측으로 메시지 전송한다. 응용프로그램측의 메시지 핸들러 에서는 함수 GetStreamData 호출하여 스트림데이터 확보처리 한다.

Code Example 1 . SetMessageDevice

```
#define WM_STREAM_DEVICE WM_USER+203  // Define Message to get message from LXDeviceAPI.

void CLXDeviceAPI_Sample1View::OnMenuSetmessagedevice()
{
	SetMessageDevice_LXDeviceAPI(m_iDeviceHandlingID, MSGTYPEID0_DEVICE_LXDAPI ,this->m_hWnd,
WM_STREAM_DEVICE,1); //
}
```

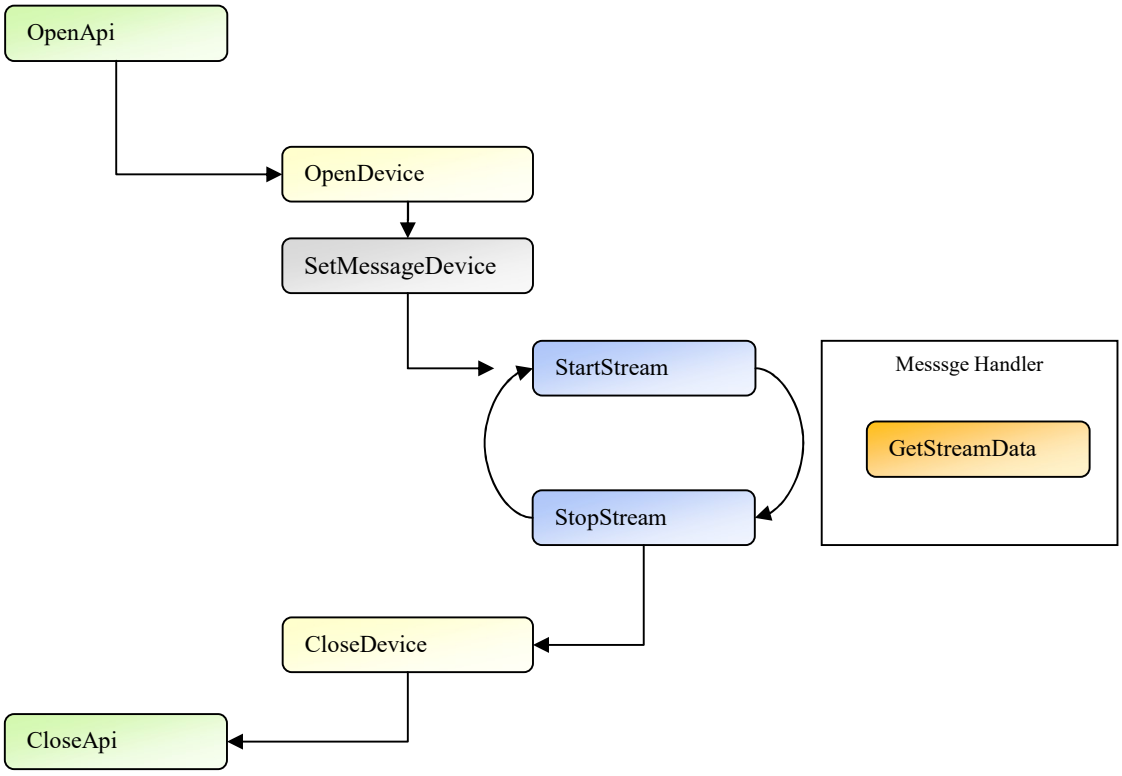
Code Example 2. StartStream

```
void CLXDeviceAPI_Sample1View::OnMenuStartstream()
{

	StartStream_LXDeviceAPI(m_iDeviceHandlingID);
}
```

Code Example 3. Message Handler

```
/* Real Time Acquisition */

afx_msg LRESULT CLXDeviceAPI_Sample1View::OnStreamData(WPARAM wParam, LPARAM lParam)
{
	unsigned int uintWPARAM = (unsigned int)wParam;

	unsigned char msgtype_id = (unsigned char)(uintWPARAM >> 8); //get the lowest 2'nd byte(message type id).

	unsigned char msgtype_subid = (unsigned char)(uintWPARAM); //get the lowest 1st byte(message type sub id).

	switch (msgtype_id)
	{
	case MSGTYPEID0_DEVICE_LXDAPI: // for real time stream type messages.

		switch (msgtype_subid)
		{
		case 0:
			GetStreamData_LXDeviceAPI(uintWPARAM); //  the new stream data is allocated on stStreamData
which is ST_STREAMDATA_LXDAPI type variable.
			// 이 지점에서 확보된 데이터 stStreamData 활용
			break;
		} // switch (msgtype_subid)
		break;  // case MSGTYPEID0_DEVICE_LXDAPI: // for real time stream type messages.
	}// switch (msgtype_id)

	return 0;
}
```

API 함수



OpenDevice 로 기기와 통신 개설된 상태에서, StartStream 과 StopStrem 은 필요에 따라 교번식으로 여러 번 반복 호출 가능하다.

StartStream 호출 이후에 API 에서는 MSGTYPEID0_DEVICE_LXDAPI 의 메시지가 전송되며 응용 프로그램에서는 메시지 핸들러가 구현되어야 한다. 메시지 핸들러 내에서 메시지 수신될 때마다 GetStreamData 함수를 호출하여 스트림데이터를 확보하여 응용프로그램에서 활용한다. StopStream 함수 호출하면 API 는 Stream 메시지 발생하지 않는다.

CloseDevice 는 응용프로그램에서 해당 기기를 사용하지 않을 때(예: 응용프로그램 종료시)점 CloseDevice 를 호출하고 이후 CloseApi 호출한다.

OpenApi

int OpenApi_LXDeviceAPI(int api_window, int api_selfupdate, int mode)

설명

LXDevice API 를 사용하기 위해서 가장 먼저 호출되어야 하는 함수. OpenApi 이후에 API 에서 제공되는 다른 함수 호출 가능.

인자

인자	가능한 값.	설명
int api_window	0 :API 윈도우 비활성화. 1 : API 윈도우 활성화. (기본값 : 1)	OpenApi 호출 초기 API 윈도우 보이기 여부 지정.
int api_selfupdate	0: check update 실행않음. 1: check update 실행함. (기본값 : 1)	LXDeviceAPI 는 자동원격업데이트 기능으로 항상 최신상태로 유지할 수 있다. 인자 api_selfupdate 를 1 로 한 경우 API 원격업데이트 창이 실행되어 업데이트 유무 점검처리된다. 응용프로그램에서 S/W 업데이트를 직접 관리하는 경우에는 본 인자를 0 으로 설정하여 API 의 자체 원격 업데이트 기능 해제한다.
int mode	(기본값 : 0)	Don't care

LXDeviceAPI Self Update 기능.

LXDeviceAPI Self Update 란 인터넷 기반 원격 API 업데이트 기능을 의미하며, 실행환경에서 항상 최신 버전의 LXDeviceAPI 상태로 유지할 수 있다. LXDevice Self Update 기능은 응용 프로그램측에서 함수 OpenAPI 호출시 2 번 인자를 1 로 하여 활성화 하여 사용할 수도 있고 비활성화 시킬 수도 있다. 응용프로그램에서 이미 원격 업데이트 기능이 구현되어 있고, LXDeviceAPI 신규 버전 배포도 응용프로그램에서 직접 수행할 경우에는 LXDeviceAPI Self Update 기능은 비활성화 하고, 응용프로그램이 업데이트 기능이 없는 경우엔 LXDeviceAPI Self Update 기능 활성화 시킨다.

반환값

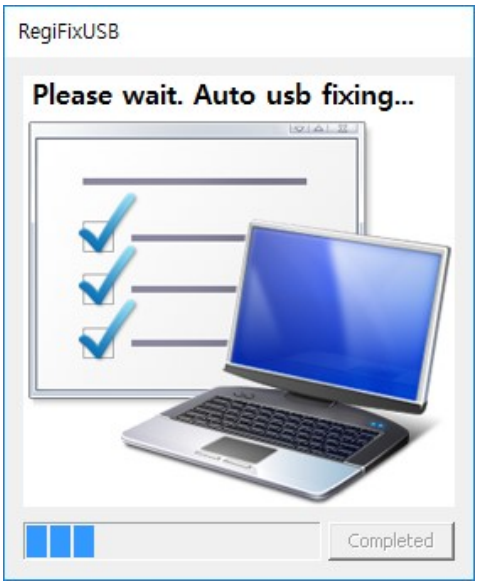
반환값	의미	설명
1	성공	
-1	실패	
-2	실패. 중복호출	OpenApi 가 이미 한 번 호출되어 정상수행된 이후에는 재 호출 불가. OpenApi 를 다시 실행하려면 CloseApi 호출 이후에 가능.
-5	윈도우 레지스트리 정상화조치중.	LXDeviceAPI 에서 RegiFixUSB.exe 실행하여 윈도우 레지스트리 정상화 처리 수행 중. USB 기기 통신 정상화 위하여 윈도우 8/10 에서 필수 조치 사항.

반환값 -5 상세설명.

LXDeviceAPI 에서 윈도우 운영체제(버전 8/10) 레지스트리 점검하여 USB 기기 정상 통신 처리를 위하여 RegiFixUSB.exe 를 자동 실행시킨 경우, 모프로그램에서 API 함수 OpenDevice 호출해도 기기와의 정상 통신 불가함 의미.

RegiFixUSB.exe 가 실행되어 레지스트리 정상화 완료되면 아래 창과 같이 표시되며, 반드시 기기의 USB 를 PC 에서 분리했다가 재 연결해야 USB 통신 정상 작동한다.

RegiFixUSB.exe 가 실행되는 경우 : PC 에 기기 USB 가 처음 연결 된 경우 1 회만 실행됨. RegiFixUSB.exe 가 1 회 실행되어 레지스트리 정상화 조치된 이후에는 RegiFixUSB.exe 는 실행되지 않는다.



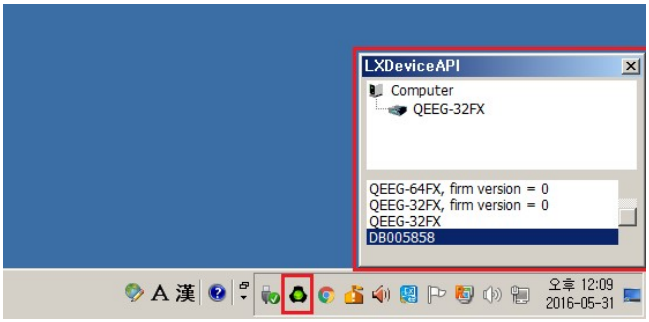
RegiFixUSB.exe 실행중 화면.

RegiFixUSB.exe 실행 완료 화면.

Code Example

```
void CLXDeviceAPI_Sample1View::OnMenuOpenapi()
{
		OpenApi_LXDeviceAPI(1,0,0);
}
```

Result



LXDeviceAPI Window Enabled.

CloseApi

int CloseApi_LXDeviceAPI()

설명

LXDeviceAPI 를 더 이상 사용하지 않는 경우 가장 마지막으로 호출한다.

반환값

반환값	의미	설명
1	성공	
-10	실패	OpenApi 호출이후에 CloseApi 호출가능함.

Code Example

```
void CLXDeviceAPI_Sample1View::OnMenuCloseapi()
{
		CloseApi_LXDeviceAPI();
}
```

OpenDevice

int OpenDevice_LXDeviceAPI(int LXDeviceID,ST_STREAMDATA_LXDAPI* p_streamdata,int numsample_return,int mode)

설명

함수의 인자로 전달된 LXDeviceID(Appendix1 기기모델별 정보참조.)에 해당하는 장치와 통신가능 상태 달성.

함수의 반환값으로 DeviceHandlingID(장치핸들링 아이디) 반환되며, 장치핸들링 아이디는 이후

장치제어함수들 호출시 입력 인자로 필수 요구되므로 응용프로그램에서 확보해둬야 한다.

인자

인자	가능한 값.	설명
int LXDeviceID	1~65535 범위의 장치 고유아이디.	통신 개설할 장치를 지정하는 것이며, 기기별 LXDeviceID 는 Appendix 1.참조.

ST_STREAMDATA_LXDAPI* p_stream_data	구조체 변수의 주소.	API 의 스트림 전송 상태에서 "스트림데이터"를 받기 위한 구조체 변수를 선언하고 변수의 주소를 함수호출시 전달한다.
		예.
		S T_STREAMDATA_LXDAPI stStreamData; // 구조체 변수 선언.
		OpenDevice_LXDevieAPI(, &stStreamData, ,); // 구조체변수 주소전달.

int numsample_return	1~128	반환 받을 샘플 수량 지정. 만일 32 를 기록했다면, API 에서는 장치로부터 각 채널당 32 샘플링의 데이터가 수집될 때마다 MSGTYPEID0_DEVICE_LXDAPI 의 메시지 전송됨.
int mode	(기본값 : 0)	Don't care

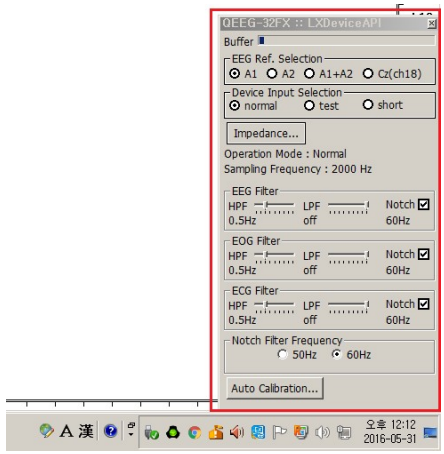
반환값

반환값	의미	설명
1 이상	성공	OpenDevice 호출결과 성공적인 경우, Device Handling ID(값의 범위 1~65535 사이의 임의 정수) 반환되며 , 앱에서 필수로 저장해야함. 장치 제어함수 호출시 인자로 필수 요구되는 값.
-1	지원장치 아님.	LXDeviceID 에 해당하는 장치가 LXDeviceAPI 에서 지원하는 장치 아님.
-2	중복호출오류	동일장치에 대해서 이미 OpenDevice 가 실행되어 통신개설된 상태인 경우.
-3	장치없음	LXDeviceID 에 해당하는 장치가 발견되지 않음. PC 에 연결되어있지 않음.
-5	기기펌업뎃해야함.	기기 펌웨어 업데이트 완료해야 OpenDevice 가능함.
-10	함수호출순서 오류	OpenApi 호출이후에 OpenDevice 호출되어야 함.
-20	인자값 범위오류	numsample_return (스트림 반환 수량) 값의 범위 1~128 까지만 가능함.

Code Example1

```
void CLXDeviceAPI_Sample1View::OnMenuOpendevice()
{
	int retv = OpenDevice_LXDeviceAPI(300, &stStreamData, NumSampleReturn, 0);// QEEG-32FX(LXDeviceID=300) Open

	if(retv > 0) // if success
	{
		m_iDeviceHandlingID = retv;
	}
	else
	{
		AfxMessageBox(_T("Fail to OpenDevice"));
	}
}
```

Result .



Device Control Panel Enabled.

Code Example2

```
/*
OpenAPI-> OpenDevice -> StartStream 한번에 모두 시행.
*/
void CLXDeviceAPI_Sample1View::OnMenuOneclickstart()
{
	int retv_openapi, retv_opendevice;

	retv_openapi = OpenApi_LXDeviceAPI(1, 0, 0); // API Open.

	if (retv_openapi > 0)
	{
		retv_opendevice = OpenDevice_LXDeviceAPI(300, &stStreamData, NumSampleReturn, 0); // QEEG-32FX Open
		if (retv_opendevice > 0)
		{
			m_iDeviceHandlingID = retv_opendevice;
			SetMessageDevice_LXDeviceAPI(m_iDeviceHandlingID, 0, this->m_hWnd,
WM_LXDAPI_MSGTYPEID_0, 1);

			StartStream_LXDeviceAPI(m_iDeviceHandlingID);
		}
	}
}
```

CloseDevice

int CloseDevice_LXDeviceAPI(int device_handling_id);

설명

OpenDevice 로 통신개설했던 장치 연결 해제.

인자

인자	가능한 값.	설명
int device_handling_id	OpenDevice 함수 반환값.	OpenDevice 가 성공적으로 수행된 경우 반환된 값을 인자로 전달한다.

반환값

반환값	의미	설명
1	성공	
-3	장치없음	device_handling_id 로 연결된 장치없음.
-10	함수호출순서 오류	OpenApi 호출이후에 OpenDevice 호출되어야 함.

Code Example

```
void CLXDeviceAPI_Sample1View::OnMenuClosedevice()
{
		CloseDevice_LXDeviceAPI(m_iDeviceHandlingID);
}
```

StartStream

int StartStream_LXDeviceAPI(int device_handling_id, int mode)

설명

OpenDevice 로 통신개설된 장치에서 스트림데이터 전송시작시킨다.

OpenDevice 에서 지정한 int numsample_data 가 수집될때마다 API 는 MSGTYPEID0_DEVICE_LXDAPI 의 메시지를 응용프로그램측으로 전송하며, 응용프로그램의 메시지 핸들러 내에서 스트림데이터 확보함수 GetStreamData 호출 필수.

인자

인자	가능한 값.	설명
int device_handling_id	OpenDevice 함수 반환값.	OpenDevice 가 성공적으로 수행된 경우 반환된 값을 인자로 전달한다.
int mode	-	Reserved. Don't care.

반환값

반환값	의미	설명
1	성공	
-3	장치없음	device_handling_id 로 연결된 장치없음.
-5	임피던스 모드.	기기의 임피던스 측정모드에서는 StartStream 불가.
-6	자동보정 모드.	기기 Auto Calibration 중에는 StartStream 불가.
-10	함수호출순서 오류	OpenApi 호출이후에 OpenDevice 호출되어야 함.

Code Example

```
void CLXDeviceAPI_Sample1View::OnMenuStartstream()
{
		StartStream_LXDeviceAPI(m_iDeviceHandlingID);
}
```

StoptStream

int StopStream_ LXDeviceAPI (int device_handling_id);

설명

API 에서의 스트림 전송 중지시킨다.

StopStream 호출하여 스트림 전송 중지 된 상태에서, 스트림 전송을 다시 시작시키려면 StartStream 호출한다.

인자

인자	가능한 값.	설명
int device_handling_id	OpenDevice 함수 반환값.	OpenDevice 가 성공적으로 수행된 경우 반환된 값을 인자로 전달한다.

반환값

반환값	의미	설명
1	성공	메시지 설정성공.
-3	장치없음	device_handling_id 로 연결된 장치없음.
-10	함수호출순서 오류	OpenApi 호출이후에 OpenDevice 호출되어야 함.

Code Example

```
void CLXDeviceAPI_Sample1View::OnMenuStopstream()
{
		StopStream_LXDeviceAPI(m_iDeviceHandlingID);
}
```

GetStreamData

int GetStreamData_LXDeviceAPI(unsigned int message_wparam);

설명

응용프로그램 측의 스트림수신 메시지 핸들러 내에서 호출하여 스트림데이터 확보한다. 함수 GetStreamData 호출하면 , 함수 OpenDevice 호출시 인자로 전달한 구조체 멤버 변수에 신규 수집된 데이터들로 갱신.

인자

인자	가능한 값.	설명
unsigned int message_wparam	(unsigned int) wParam.	스트림 수신 메시지 (MSGTYPEID0_DEVICE_LXDAPI) 로 전달된 인자중 wParam 을 unsigned int 로 형변환하여 입력한다.

반환값

반환값	의미	설명
1	성공	메시지 설정성공.
-3	장치없음	device_handling_id 로 연결된 장치없음.
-10	함수호출순서 오류	OpenApi 호출이후에 OpenDevice 호출되어야 함.

Code Example

```
afx_msg LRESULT CLXDeviceAPI_Sample1View::OnStreamData(WPARAM wParam, LPARAM lParam)
{
	unsigned char msgtype_subid = (unsigned int)wParam; // wParam 을 unsigned int 로 형변환하고, 최하위 1바이트만을 받았다.
	switch (msgtype_subid)
	{
	case 0: //
		if ((unsigned int)lParam == 0) // 스트림 데이터 메시지인 경우.
		{
			// 이 함수 호출하고 나면 구조체 변수 stStreamData에 스트림 데이터들이 들어와 있음.
			GetStreamData_LXDeviceAPI((unsigned int)wParam);
			/// 시험용 파형 그려보기. QEEG-32FX 는 EEG-32채널, EOG-2채널, ECG-1채널 총 35채널.
			///  double 형 파형 데이터를 float형으로 변경하여 챠팅해보는것.
			for (int i = 0; i < NumSampleReturn * NumChannel; i++)
			{
				testfloat_Wave[i] = (float)stStreamData.Wave_StreamData_CS[i];
			}
		}
		//인자의미. 1번 :float형 파형 데이터배열의 포인터, 2번인자 : 총 채널수, 1채널당 샘플링 수량.
		ACQPLOT_DLL_Array_Datain_Strip(testfloat_Wave, NumChannel, NumSampleReturn);
	break;
	}
	return 0;
}
```

EventMarkingOnStream

int EventMarkingOnStream_LXDeviceAPI(int device_handling_id, unsigned int event_id);

설명

API에서 스트림전송 중에 응용프로그램 측의 특정 이벤트(키보드 누름, 화면제시시점등) 정보를 event_id로 입력하면 수신된 이벤트 스트림 데이터 구조체의 멤버 Event_StreamData_CS[] event_id 기록된다.

인자

인자	가능한 값.	설명
int device_handling_id	OpenDevice 함수 반환값.	OpenDevice 가 성공적으로 수행된 경우 반환된 값을 인자로 전달한다.
unsigned int event_id	1~65535	응용프로그램에서 여러 이벤트를 식별할 수 있도록 값을 정의해서 입력함. 예. 키보드 왼쪽화살표 키 누름 : 2 키보드의 오른쪽 화살표 키 누름 : 3

반환값

반환값	의미	설명
1	성공	메시지 설정성공.
-3	장치없음	device_handling_id 로 연결된 장치없음.
-10	함수호출순서 오류	OpenApi 호출이후에 OpenDevice 호출되어야 함.

Code Example 1

```
// when menu clicked, let's set the event_id as 20000
void CLXDeviceAPI_Sample1View::OnMenuEventmarkingonstream()
{
	EventMarkingOnStream_LXDeviceAPI(m_iDeviceHandlingID, 20000);
}
```

Code Example 2

```
// when keyboard pressed. let's set the event_id as arrow ky down. up 30000, down 40000,left 50000, right
60000

void CLXDeviceAPI_Sample1View::OnKeyDown(UINT nChar, UINT nRepCnt, UINT nFlags)
{

	switch (nChar)
	{
	case VK_UP:

		EventMarkingOnStream_LXDeviceAPI(m_iDeviceHandlingID, 30000);

		break;

	case VK_DOWN:

		EventMarkingOnStream_LXDeviceAPI(m_iDeviceHandlingID, 40000);

		break;

	case VK_LEFT:

		EventMarkingOnStream_LXDeviceAPI(m_iDeviceHandlingID, 50000);

		break;

	case VK_RIGHT:

		EventMarkingOnStream_LXDeviceAPI(m_iDeviceHandlingID, 60000);

		break;

	}

	// TODO: Add your message handler code here and/or call default

	CView::OnKeyDown(nChar, nRepCnt, nFlags);
}
```

SetMessageDevice

int SetMessageDevice_LXDeviceAPI(int device_handling_id,int msgtype_id,HWND hwnd_msgrcv,int msg_id,int onoff);

설명

OpenDevice 로 통신개설된 장치에서 수신받을 메시지 설정한다.

인자

인자	가능한 값.	설명
int device_handling_id	OpenDevice 함수 반환값.	OpenDevice 가 성공적으로 수행된 경우 반환된 값을 인자로 전달한다.
int msgtype_id	MSGTYPEID0_DEVICE_LXDAPI ~ MSGTYPEID9_DEVICE_LXDAPI	메시지타입아이디 . LXdeviceAPI.h 에서 정의된 값.
HWND hwnd_msgrcv	윈도우핸들	API 에서 전송되는 메시지(msgtype_id 에 해당하는)를 수신할 응용프로그램측의 윈도우 핸들을 전달한다.
int msg_id	WM_USER ~ WM_USER+31643 WM_APP ~ WM_APP+16383	응용프로그램에서 다른 메시지와 중복되지 않도록 정의하여 입력한다.
int onoff	1: 메시지 전송 켜기 0 : 메시지 전송끄기	msgtype_id 에 해당하는 메시지 발생유무.

반환값

반환값	의미	설명
1	성공	메시지 설정성공.
-3	장치없음	device_handling_id 에 해당하는 장치없음.
-10	함수호출순서 오류	OpenApi 호출이후에 SetMessageDevice 호출되어야 함.

Code Example

```
#define WM_STREAM_DEVICE WM_USER+203  // Define Message to get message from LXDeviceAPI.

void CLXDeviceAPI_Sample1View::OnMenuSetmessagedevice()
{
        SetMessageDevice_LXDeviceAPI(m_iDeviceHandlingID, MSGTYPEID0_DEVICE_LXDAPI ,this->m_hWnd,
WM_STREAM_DEVICE,1); //
}
```

SetSampleFrequency

int SetSampleFrequency_LXDeviceAPI(int device_handling_id, unsigned int sample_frequency)

설명

OpenDevice 로 통신 개설된 장치의 샘플링 주파수 설정.

인자

인자	가능한 값.	설명
int device_handling_id	OpenDevice 함수 반환값.	OpenDevice 가 성공적으로 수행된 경우 반환된 값을 인자 전달한다.
unsigned int sample_frequency	Hz 단위의 샘플링 주파수.	기기에 따라 설정 가능한 값이 다름. Appendix1. 참조.

반환값

반환값	의미	설명
1	성공	메시지 설정성공.
-3	장치없음	device_handling_id 로 연결된 장치없음.
-10	함수호출순서 오류	OpenApi 호출이후에 OpenDevice 호출되어야 함.
-20	인자값 오류	기기에서 지원되지 않은 샘플링 주파수. 기기별 지원되는 샘플링 주파수 Appendix.1 참조.

Code Example

```
/*
기기의 샘플링 주파수 2000Hz로 설정.
*/
void CLXDeviceAPI_Sample1View::OnSetsamplefrequency2000hz()
{
		SetSampleFrequency_LXDeviceAPI(m_iDeviceHandlingID, 2000);
}
```

SetDeviceControlPanel

int SetDeviceControlPanel_LXDeviceAPI(int device_handling_id,int para0, int para1);

설명

LXDeviceAPI 의 기기제어판 제어요소 활성화/비활성화.

인자

인자	가능한 값.	설명
int device_handling_id	OpenDevice 함수 반환값.	OpenDevice 가 성공적으로 수행된 경우 반환된 값을 인자로 전달한다.
int para0	0	0 : 기기제어판 요소 전체선택(측정치에 영향을 미치는 모든 제어요소)
int para1	0, 1	0 : 비활성화. 1 : 활성화.

반환값

반환값	의미	설명
1 이상	성공	인자 eeg_refelectrode 에 현재 기기에 설정 되어있는 기준전극 정보 반환됨.
-3	장치없음	LXDeviceID 에 해당하는 장치가 발견되지 않음. PC 에 연결되어있지 않음.
-10	함수호출순서 오류	OpenApi 호출이후에 OpenDevice 호출되어야 함.

Code Example

```
void CLXDeviceAPI_Sample1View::OnMenuSetdevicecontrolpanel()
{
		static int para1=0;
		SetDeviceControlPanel_LXDeviceAPI(m_iDeviceHandlingID, 0, para1);
		// toggling para1.
		if (para1) para1 = 0;
		else para1 = 1;
}
```

GetFilterFrequency

int GetFilterFrequency_LXDeviceAPI(int device_handling_id,int signal_source, float * freq_hpf, float * freq_lpf, float * freq_notch);

설명.

기기의 각 신호소스별 적용중인 필터 정보 받기.

인자

인자	가능한 값.	설명
int device_handling_id	OpenDevice 함수 반환값.	OpenDevice 가 성공적으로 수행된 경우 반환된 값을 인자로 전달한다.
int signal_source	0,1,2	0=EEG, 1=EOG, 2=ECG
float * freq_hpf	-100., 필터 주파수	High Pass Filter 컷오프 주파수. 단위 : Hz, 유효자리 : 소수점 1 자리. -100. : HPF 적용되지 않은 경우.
float * freq_lpf	-100., 필터 주파수	Low Pass Filter 컷오프 주파수. 단위 : Hz, 유효자리 : 소수점 1 자리. -100. : LPF 적용되지 않은 경우.
float * freq_notch	-100. , 50.0, 60.0	Notch Filter 주파수. 단위 : Hz, 유효자리 : 소수점 1 자리. -100. : Notch 필터 적용되지 않은 경우.

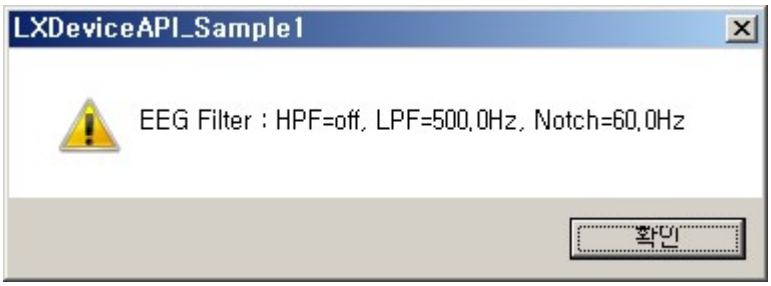
반환값

반환값	의미	설명
1 이상	성공	인자 freq_hpf, freq_lpf, freq_notch 에 현재 기기에 설정 되어있는 필터 주파수 값이 반환됨.
-3	장치없음	device_handling_id 인 장치가 없음.
-10	함수호출순서 오류	OpenApi 호출이후에 OpenDevice 호출되어야 함.

Code Example

```
void CLXDeviceAPI_Sample1View::OnMenuGetfilterfrequency()
{
	float f_hpf, f_lpf, f_notch;
	CString cst_hpf, cst_lpf, cst_notch;
	// Get EEG Filter info
	if (GetFilterFrequency_LXDeviceAPI(m_iDeviceHandlingID,0, &f_hpf, &f_lpf, &f_notch) == 1)
	{
		if (f_hpf < -1.f)		cst_hpf = _T("HPF=off, ");
		else				cst_hpf.Format(_T("HPF=%.1fHz, "), f_hpf);

		if (f_lpf < -1.f)		cst_lpf = _T("LPF=off, ");
		else				cst_lpf.Format(_T("LPF=%.1fHz, "), f_lpf);

		if (f_notch < -1.f)		cst_notch = _T("Notch=off");
		else				cst_notch.Format(_T("Notch=%.1fHz"), f_notch);

		AfxMessageBox(_T("EEG Filter : ") + cst_hpf + cst_lpf + cst_notch);
	}
}
```

Result



GetSampleFrequency

int GetSampleFrequency_LXDeviceAPI(int device_handling_id, int * sample_frequency);

설명.

기기에서 설정된 샘플링 주파수 받기.

인자

인자	가능한 값.	설명
int device_handling_id	OpenDevice 함수 반환값.	OpenDevice 가 성공적으로 수행된 경우 반환된 값을 인자로 전달한다.
int * sample_frequency	-	기기에 설정된 샘플링 주파수. 단위 : Hz

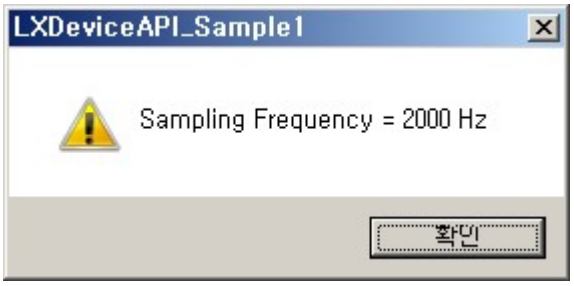
반환값

반환값	의미	설명
1 이상	성공	인자 eeg_refelectrode 에 현재 기기에 설정 되어있는 기준전극 정보 반환됨.
-3	장치없음	LXDeviceID 에 해당하는 장치가 발견되지 않음. PC 에 연결되어있지 않음.
-10	함수호출순서 오류	OpenApi 호출이후에 OpenDevice 호출되어야 함.

Code Example

```
void CLXDeviceAPI_Sample1View::OnMenuGetsamplefrequency()
{
	int sample_frequency;
	CString cst;

	if (GetSampleFrequency_LXDeviceAPI(m_iDeviceHandlingID, &sample_frequency) == 1)
	{
		cst.Format(_T("Sampling Frequency = %d Hz"), sample_frequency);

		AfxMessageBox(cst);
	}
}
```

Result



GetEEGRefElectrode

int GetEEGRefElectrode_LXDeviceAPI(int device_handling_id , int * eeg_refelectrode);

설명.

EEG 기준전극 선택정보 받기.

인자

인자	가능한 값.	설명
int device_handling_id	OpenDevice 함수 반환값.	OpenDevice 가 성공적으로 수행된 경우 반환된 값을 인자로 전달한다.
int * eeg_refelectrode	0, 1, 2, 3	0: A1 1: A2 2 : A1+A2 3 : Cz(ch18) .EEG Cap 타입 전극 사용시 Cz 에 해당하며, 채널번호 18 에 해당.

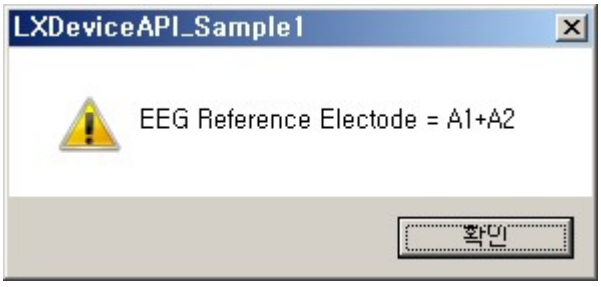
반환값

반환값	의미	설명
1 이상	성공	인자 eeg_refelectrode 에 현재 기기에 설정 되어있는 기준전극 정보 반환됨.
-3	장치없음	LXDeviceID 에 해당하는 장치가 발견되지 않음. PC 에 연결되어있지 않음.
-10	함수호출순서 오류	OpenApi 호출이후에 OpenDevice 호출되어야 함.

Code Example

```
void CLXDeviceAPI_Sample1View::OnMenuGeteegrefelectrode()
{
	int eeg_refelectrode;
	CString cst_eegref;

	if (GetEEGRefElectrode_LXDeviceAPI(m_iDeviceHandlingID,&eeg_refelectrode) == 1)
	{
		if (eeg_refelectrode == 0)		cst_eegref = _T("A1");
		else if(eeg_refelectrode == 1)		cst_eegref = _T("A2");
		else if (eeg_refelectrode == 2)		cst_eegref = _T("A1+A2");
		else if (eeg_refelectrode == 3)		cst_eegref = _T("Cz(ch18)");
		AfxMessageBox(_T("EEG Reference Electode = ") + cst_eegref);
	}
}
```

Result



CheckForUpdate

int CheckForUpdate_LXDeviceAPI(int closeifnoupdate);

설명.

API self update 실행.

인자

인자	가능한 값.	설명
int closeifnoupdate	0, 1	1 : 업데이트 할 사항없는 업데이트윈도우 자동 닫기. 0 : 업데이트 윈도우 자동닫기 하지 않음.

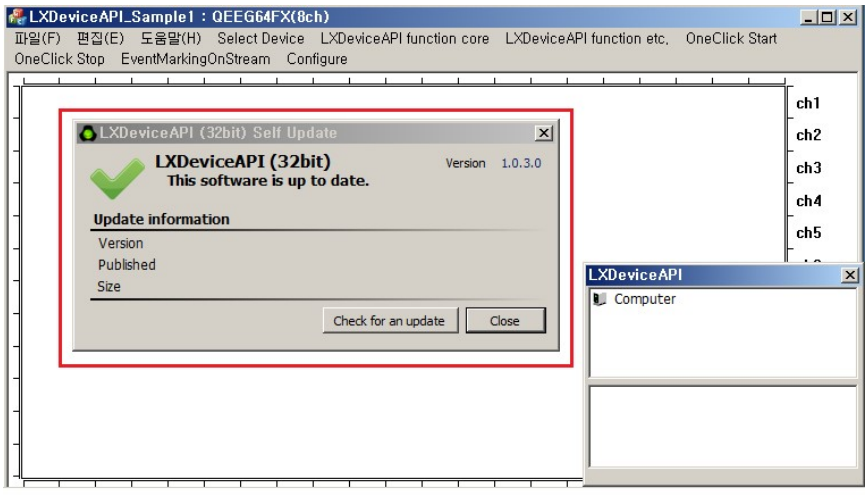
반환값

반환값	의미	설명
1 이상	성공	인자 eeg_refelectrode 에 현재 기기에 설정 되어있는 기준전극 정보 반환됨.
-10	함수호출순서 오류	OpenApi 호출이후에 OpenDevice 호출되어야 함.

Code Example

```
void CLXDeviceAPI_Sample1View::OnMenuCheckforupdate()
{
	CheckForUpdate_LXDeviceAPI(0); // parameter 0 : No Auto Close
	//CheckForUpdate_LXDeviceAPI(1); // parameter 1 : Auto close check program if there is no update.
}
```

Result



Appendix 1. 기기 모델별 정보.

기기 모델별 LXDeviceID, 샘플링 주파수, 채널인덱스별 측정치.

기기모델	LXDeviceID	샘플링 주파수.	채널인덱스별 측정치.
QEEG-32FX	300	250Hz, 500Hz, 1000Hz, 2000Hz	0~31 : EEG 채널 1~32, 32:EOG1, 33:EOG2, 34:ECG
QEEG-64FX(8ch)	16408	250Hz, 500Hz, 1000Hz, 2000Hz	0~7 : EEG 채널 1~8, 8:EOG1, 9:EOG2, 10:ECG
QEEG-64FX(16ch)	16416	250Hz, 500Hz, 1000Hz, 2000Hz	0~15 : EEG 채널 1~16, 16:EOG1, 17:EOG2, 18:ECG
QEEG-64FX(24ch)	16424	250Hz, 500Hz, 1000Hz, 2000Hz	0~23 : EEG 채널 1~24, 24:EOG1, 25:EOG2, 26:ECG
QEEG-64FX(32ch)	16432	250Hz, 500Hz, 1000Hz, 2000Hz	0~31 : EEG 채널 1~32, 32:EOG1, 33:EOG2, 34:ECG
QEEG-64FX(40ch)	16440	250Hz, 500Hz, 1000Hz,	0~39 : EEG 채널 1~40, 40:EOG1, 41:EOG2, 42:ECG
QEEG-64FX(48ch)	16448	250Hz, 500Hz, 1000Hz,	0~47 : EEG 채널 1~48, 48:EOG1, 49:EOG2, 50:ECG
QEEG-64FX(56ch)	16456	250Hz, 500Hz, 1000Hz,	0~55 : EEG 채널 1~56, 56:EOG1, 57:EOG2, 58:ECG
QEEG-64FX(64ch)	16464	250Hz, 500Hz, 1000Hz,	0~63 : EEG 채널 1~64, 64:EOG1, 65:EOG2, 66:ECG

비고 : QEEG-64FX(8ch), ..., QEEG-64FX(64ch) 은 모두 동일 모델 QEEG-64FX 이며 괄호속 숫자는 최대채널수옵션 .

Revision History

Release Date	Doc. ID	Description of Change
2017-04-25	LXD64 V1	First release