

TECH NOTE :: ClipX API für Linux

Version: 2019-09-09

Autor: Michael Guckes, Florian Schopp, Roland Siepmann

Status: HBM: Public

Kurzbeschreibung

Dies ist eine Anleitung zur Verwendung der ClipXLinuxApi.so Datei zur Einbindung von ClipX in eigene Programme in Linux-basierte Systeme. Die Bibliothek verwendet eine TCP/IP Verbindung zum Port 55000, weswegen eine weitere Verbindung zu diesem Port, während einer Messung mithilfe der .so-Datei nicht möglich ist. Die Verwendung des Webserver ist weiterhin ohne Einschränkungen möglich, da diese Kommunikation nicht den Port 55000 verwendet. Die ClipX API basiert auf einer Kommunikation mit dem Objektverzeichnis von ClipX und verwendet den internen Fifo Speicher dessen. Daher sind Messungen mit dieser Methode an eine maximale Messrate von 1kHz gebunden (minimale Rate 1Hz).

Hinweis: Stellen Sie sicher, dass Sie die aktuellsten Treiber verwenden.

<https://www.hbm.com/de/7077/clipx-praeziser-leicht-integrierbarer-messverstaerker/>



Einbindung

Die ClipX API kann prinzipiell in jedes Projekt eingebunden werden. Da die Programmierung in C++ vorgenommen wurde, ist die Einbindung in gleichsprachige Umgebungen am unkompliziertesten. Sowohl für die Einbindung in C, also auch für die Einbindung in C#, liegt dieser Technote ein Beispiel bei.

Funktionen

ClipX ist in Form einer C++ Klasse implementiert. Somit kann jedes einzelne Gerät in der folgenden Programmierung als separate Instanz dieser Klasse realisiert werden. Folgende Methoden sind zugänglich:

```
class ClipX
{
public:
    ClipX_API ClipX();
    ClipX_API int Connect(const char* IP);
    ClipX_API int Disconnect();
    ClipX_API int SDORead(int idx, int subidx, char* res, int size);
    ClipX_API int SDOWrite(int idx, int subidx, const char* val);
    ClipX_API int startMeasurement();
    ClipX_API int AvailableLines();
    ClipX_API int ReadNextLine(double* MVLine);
    ClipX_API int ReadNextBlock(int maxreads, double* time, double* value1, double* value2, double* value3, double* value4, double* value5, double* value6);
    ClipX_API int stopMeasurement();
    ClipX_API bool GetOverflowFlag();
    ClipX_API bool isConnected();
    ClipX_API void ClearBuffer();
    ClipX_API ~ClipX();
    bool KeepAlive = true;
};
```

Connect

Die „Connect“-Methode erstellt einen Socket und verbindet zu diesem. Als Eingabeparameter wird die IP-Adresse des Gerätes eingegeben. Bei erfolgreicher Verbindung liefert diese Methode den Integer 0 zurück, andernfalls einen Wert kleiner 0. Bei erfolgreicher Verbindung wird ebenfalls ein „Keep-Alive“-Thread gestartet (mehr hierzu im Kapitel „Keep Alive“).

Disconnect

Die „Disconnect“-Methode trennt die Verbindung zu ClipX und schließt den Socket. Bei erfolgreicher Ausführung dieser Methode wird 0 zurückgeliefert.

SDORead

Die „SDORead“-Methode ermöglicht ein Lesen aus dem Objektverzeichnis von ClipX. Die Eingabeparameter sind hierbei der Index und Subindex, ein Chararray, in den das Ergebnis gespeichert wird, und die Größe, die gelesen werden soll (sizeof(res)). Als Rückgabeparameter wird die Anzahl an gelesenen Bytes geliefert.

Hinweis: Siehe ClipX Bedienungsanleitung für Informationen bzgl. des Objektverzeichnisses und der verfügbaren Indizes.

SDOWrite

Die „SDOWrite“-Methode ermöglicht das Schreiben in das Objektverzeichnis von ClipX. Die Eingabeparameter sind hierbei Index, Subindex und der zu schreibende Wert als Char/String. Bei erfolgreichem Schreiben gibt diese Methode 0, bei fehlgeschlagenem Schreiben -1 zurück.

Hinweis: Siehe ClipX Bedienungsanleitung für Informationen bzgl. des Objektverzeichnisses und der verfügbaren Indizes.

startMeasurement

Die „startMeasurement“-Methode initialisiert die Messung. Hierbei wird der Keep Alive Thread beendet, das Füllen des Fifo's gestartet und anschließend ein Thread gestartet, der alle 100 Millisekunden die Einträge aus dem Fifo liest und diese in einem Buffer/Struct speichert.

AvailableLines

Die „AvailableLines“-Methode gibt die Anzahl der verfügbaren ungelesenen Messwertzeilen zurück. Dies ist die Anzahl, der bereits aus dem Fifo abgeholten, aber noch nicht mit der Methode „ReadNextLine“ gelesenen Wertezeilen.

ReadNextLine

Die „ReadNextLine“-Methode liest die nächst verfügbare Messwertzeile aus dem internen Buffer/Struct des Programms aus. Als Eingabeparameter wird ein Doublearray der Größe 7 erwartet, in das zuerst der Zeitstempel und anschließend die sechs Signale gespeichert werden. Als Rückgabewert liefert diese Methode die Anzahl noch verfügbarer Zeilen.

Hinweis: Bei Aufruf der Methode „startMeasurement“ wird die aktuelle Zeit aus ClipX ausgelesen. Die Zeitstempel der einzelnen Messwerte ergeben sich dann aus der Addition dieser Zeit mit den Ticks, die der Fifo liefert.

ReadNextBlock

Die „ReadNextBlock“-Methode liest eine vom Nutzer durch die Variable „maxreads“ vorgegebene, maximale Anzahl an Werten aus dem internen Buffer/Struct des Programms aus. Als Eingabeparameter werden für jedes der 7 Signale (6 Signale + Zeit) ein Doublepointer erwartet, in die die Werte gespeichert werden. Als Rückgabewert liefert diese Methode die Anzahl noch verfügbarer Zeilen. Wenn weniger Werte verfügbar sind, als durch „maxreads“ vorgegeben, so werden lediglich die verfügbaren Werte ausgelesen.

Hinweis: Bei Aufruf der Methode „startMeasurement“ wird die aktuelle Zeit aus ClipX ausgelesen. Die Zeitstempel der einzelnen Messwerte ergeben sich dann aus der Addition dieser Zeit mit den Ticks, die der Fifo liefert.

stopMeasurement

Die „stopMeasurement“-Methode deinitialisiert die Messung, beendet also den Messthread und startet anschließend wieder den Keep Alive Thread. Wenn die Messung erfolgreich beendet wurde, gibt diese Methode 0 zurück. Falls kein Gerät verbunden war oder keine Messung gestartet wurde, bevor die Methode aufgerufen wird, liefert die Methode -1 zurück.

GetOverflowFlag

Die „GetOverflowFlag“-Methode liefert einen Boolean, der anzeigt, ob bereits Messwerte, durch „Nicht-Abholen“, verloren gegangen sind. Das Overflowflag lässt nur auf einen Overflow im internen Buffer/Struct des Programms schließen und nicht auf einen Overflow des Fifos.

Hinweis: Der Overflow des Fifos lässt sich aus den Fifo-Kontrollflags durch eine Objektverzeichnisabfrage ermitteln.

isConnected

Die „isConnected“-Methode liefert einen Boolean, der anzeigt, ob aktuell eine Verbindung zu einem ClipX besteht.

ClearBuffer

Die „ClearBuffer“-Methode setzt den Readpointer des Programms auf die Stelle des Writepointers. Durch diese Methode werden die Werte, die gelesen wurden, nicht gelöscht, sondern nur bei nächstem Aufruf der „ReadNextLine“-Methode, die neusten Werte ausgelesen.

KeepAlive

Die „KeepAlive“-Methode ist eine private Methode, die in diesem Kontext nicht zugänglich ist. Sie wird intern verwendet, um bei Ausbleiben einer Kommunikation, ein Timeout der Verbindung zu unterdrücken. Hierzu wird alle 10 Sekunden eine KeepAliveMessage an ClipX gesendet. Der Thread wird bei Start der Messung (startMeasurement) und bei Schließen der Verbindung (Disconnect) automatisch beendet.

ClipX_Interface für C Implementierungen

Das ClipX Interface bietet eine Schnittstelle für die Einbindung in C. Hierbei wird, statt eine Klasse zu erstellen, ein Handle geliefert, dass den vollen Funktionalitätsumfang bietet, ohne eine Klasse zu erstellen.

```
typedef struct sClipX * MHandle;
ClipX_API MHandle __stdcall ClipX_Connect(const char *);
ClipX_API void __stdcall ClipX_SDORRead(MHandle m, int idx, int subidx, char* val, int size);
ClipX_API void __stdcall ClipX_SDOWrite(MHandle m, int idx, int subidx, char* val);
ClipX_API int __stdcall ClipX_startMeasurement(MHandle m);
ClipX_API int __stdcall ClipX_AvailableLines(MHandle m);
ClipX_API int __stdcall ClipX_ReadNextLine(MHandle m, double* MVLine);
ClipX_API int __stdcall ClipX_stopMeasurement(MHandle m);
ClipX_API void __stdcall ClipX_Disconnect(MHandle m);
```

Der Aufruf erfolgt genauso, wie bei der normalen Klasse, außer dass der Methode immer das entsprechende Handle als erstes Eingabeargument mitgeliefert werden muss.

Einbindungsbeispiel

Die C++-Dateien „testmain.cpp“ und „testmainint.cpp“ stellen Beispiele für eine einfache Einbindung der API in ein Programm dar. Erstere verwendet hierzu direkt die Klasse ClipX, wohingegen zweiteere das C-Interface zur Einbindung verwendet.

```
#include "ClipX.h"
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <time.h>

int main() {
    ClipX dev = ClipX();
    char buf[100];
    dev.Connect("172.21.64.181");
    dev.SDOWrite(0x4428, 8, "100");
    dev.SDORRead(0x4428, 8, buf, 100);
    printf("Fifo Fill Rate: %s\n", buf);
    dev.startMeasurement();
    printf("Measurement Started");
    for (int i = 0; i < 2; i++) {
        printf("Lines: %i", dev.AvailableLines());
        //std::cout << dev.AvailableLines() ;
        double values[7];
        do {
            if (dev.ReadNextLine(values) > 0)
                printf("Time: %t %f %t %f %t %f %t %f %t %f %t %f %t %f %t %f\n", values[0], values[1], values[2], values[3], values[4], values[5], values[6]);
        } while (dev.AvailableLines() > 0);
        sleep(1);
    }
    dev.stopMeasurement();
    dev.Disconnect();
}
```

Rechtlicher Hinweis

Diese Beispiele dienen lediglich der Veranschaulichung. Sie unterliegen keinen Gewährleistung oder Haftungsansprüchen.