TECH NOTE :: ClipX Matlab Einbindung

Version: 2019-08-27
Autor: Michael Guckes, Roland Siepmann
Status: HBM: Public

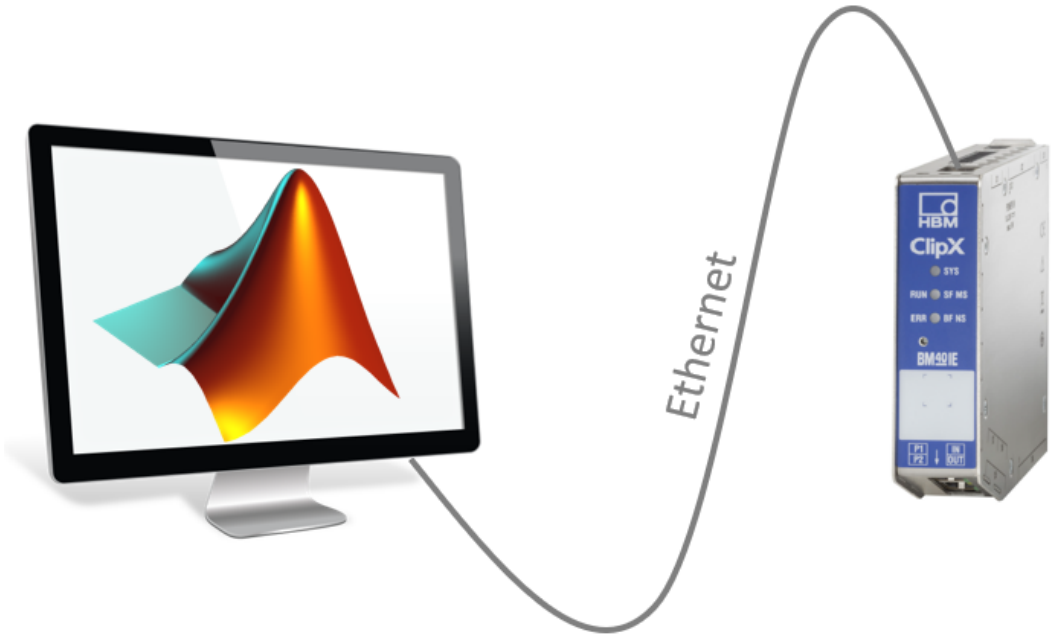
Kurzbeschreibung

Dies ist eine Anleitung zur Einbindung von ClipX in Matlab. Zur Kommunikation mit ClipX wird die HBM ClipX API verwendet, welche mit dem Objektverzeichnis von ClipX kommuniziert und für Messungen den Fifospeicher von ClipX nutzt. ClipX verwendet eine feste Messwert-Abtastrate von 19,2 kHz. Die Übertragungsrate der Werte von ClipX zu Matlab ist von 0,1Hz bis 1kHz einstellbar.

Da in diesem Verfahren eine TCP/IP Verbindung zu ClipX hergestellt wird, kann während der Messung mit Matlab, kein anderes Gerät auf den Port 55000 zugreifen. Die Verwendung des Webservers ist weiterhin ohne Einschränkungen möglich, da diese Kommunikation nicht den Port 55000 verwendet.
Im Folgenden wird vorausgesetzt, dass Matlab bereits installiert wurde.

Wichtig: Für eine korrekte Darstellung der Signale von ClipX müssen diese, wie in diesem Beispiel, über dessen NTP-Zeitkanal geplottet werden.

Hinweis: Stellen Sie sicher, dass Sie die aktuellste Version der ClipX API verwenden.
https://www.hbm.com/de/7077/clipx-praeziser-leicht-integrierbarer-messverstaerker/



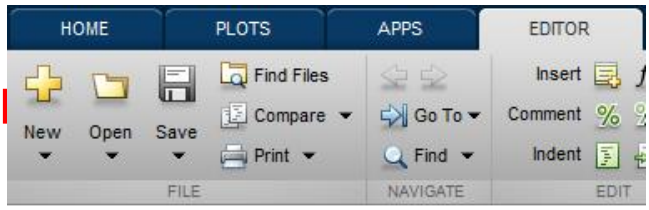
Inhalt

Das zur Verfügung gestellte Beispiel besteht aus den folgenden Dateien:

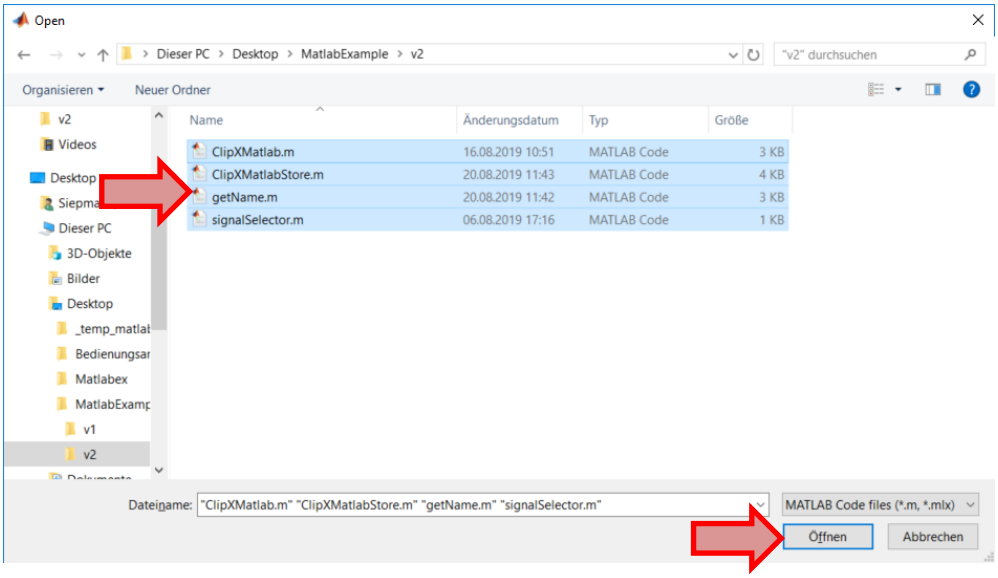
- Matlabskript „ClipXMatlab.m“ zur Messung mit einem Livegraph
- Matlabskript „ClipXMatlabStore.m“ zur Messung mit Datenspeicherung und anschließendem Plot
- Matlabfunktion „signalSelector.m“ zur Signalauswahl im Fifo
- Matlabfunktion „getName.m“ zur Ermittlung der Namen der ausgewählten Signale

Zur Durchführung einer Messung wird Matlab gestartet.

- In der Menüleiste „Öffnen“ wählen



- Anschließend die beiden oben genannten Dateien öffnen



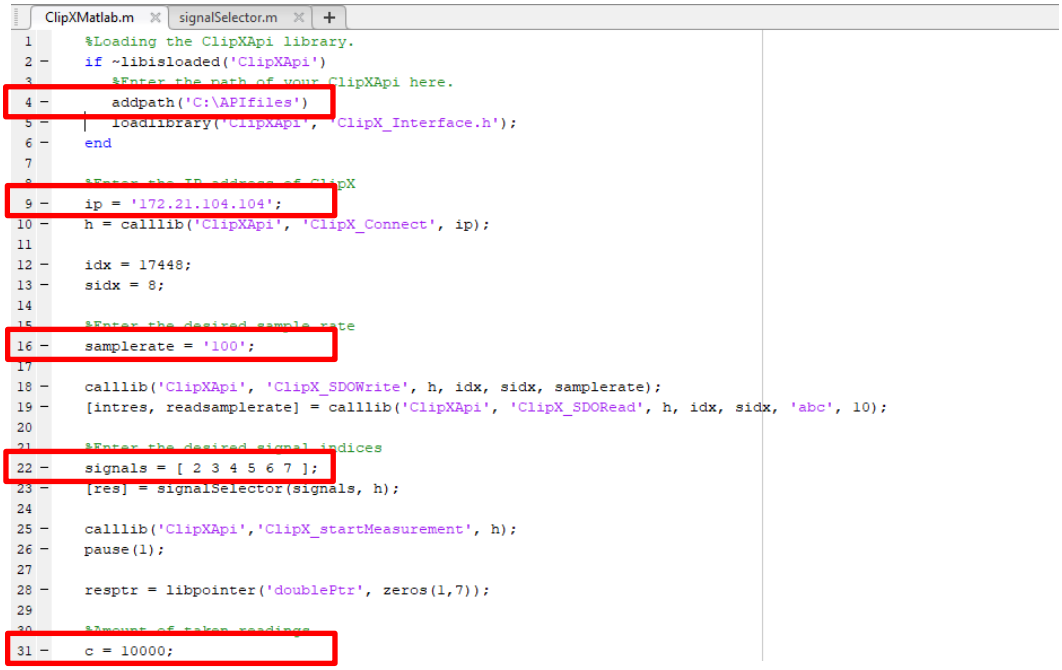
ClipXMatlab.m

Mit diesem Matlabskript wird eine Messung mit einem Livegraph gestartet. Hierzu werden die in der .dll-Datei enthaltenen Funktionen verwendet.

Parametereinstellungen

Die folgenden Parameter sind in der Datei ClipXMatlab.m anzupassen:

- addpath: Falls nicht im gleichen Ordner wie das Skript: Pfad der .dll und .h Datei
- ip: Die IP-Adresse des gewünschten ClipX Gerätes
- samplerate: Die gewünschte Samplerate (max. 1kHz, da der Fifo verwendet wird, siehe Korrespondenztabelle)
- signals: Die Signalindizes der gewünschten Signale die von ClipX übertragen werden sollen
- c: Anzahl der aufgenommenen Messwerte



Die Funktion „signalSelector“ stellt die gewünschten Signale für die Übertragung mit dem Fifo ein. Sofern diese Funktion nicht benötigt wird, kann sie einfach auskommentiert werden („%“).

Visualisierungseinstellungen

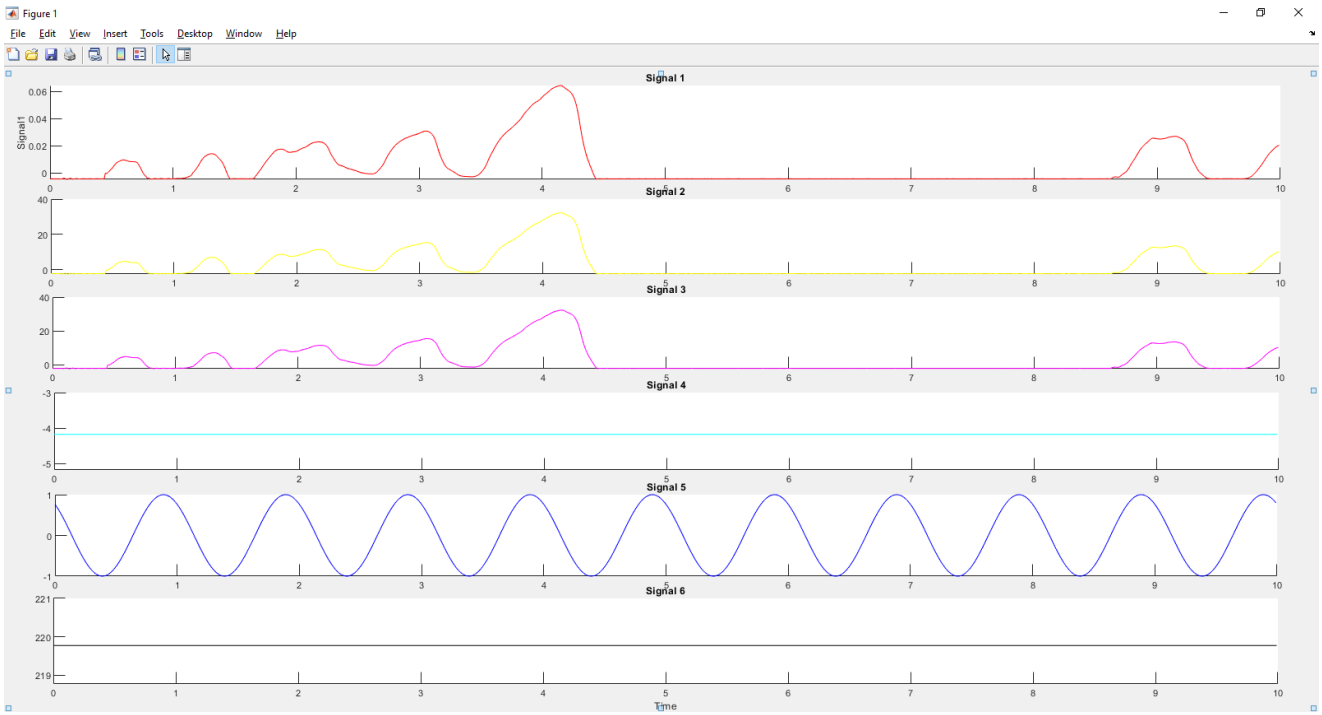
Als Visualisierung wurden in diesem Beispiel animated lines verwendet. Diese werden während des Messvorgangs laufend geupdatet und fungieren somit als Liveplot. Hier kann natürlich jede beliebige Visualisierung, wie z.B. die klassische plot() Funktion von Matlab, verwendet werden.

```
36 -    figure
37 -    grid on
38 -    hold on
39
40 -    subplot(6,2,1)
41 -    an1 = animatedline('Color','r');
42 -    title('Signal 1')
43 -    ylabel('Signal1')
44 -    subplot(6,2,2)
45 -    an2 = animatedline('Color','y');
46 -    title('Signal 2')
47 -    subplot(6,2,3)
48 -    an3 = animatedline('Color','m');
49 -    title('Signal 3')
50 -    subplot(6,2,4)
51 -    an4 = animatedline('Color','c');
52 -    title('Signal 4')
53 -    subplot(6,2,5)
54 -    an5 = animatedline('Color','b');
55 -    title('Signal 5')
56 -    subplot(6,2,6)
57 -    an6 = animatedline('Color','k');
58 -    title('Signal 6')
59 -    xlabel('Time')
```

Messwertspeicherung

Eine Speicherung der Messwerte ist in dieser Beispieldatei nicht vorgesehen, da die Dynamik der Liveplots hierdurch eingeschränkt wird. Falls diese gewünscht ist, kann sie aus der Datei ClipXMatlabStore.m übernommen werden.

Zur Durchführung der Messung wird einfach das Skript ausgeführt. Eine Messung mit der vorliegenden Visualisierung sieht beispielsweise aus, wie folgt:



ClipXMatlabStore.m

Mit diesem Matlabskript wird eine Messung mit Messwertspeicherung und anschließendem Plot gestartet. Hierzu werden die in der .dll-Datei enthaltenen Funktionen verwendet.

Parametereinstellungen

Die folgenden Parameter sind in der Datei ClipXMatlabStore.m anzupassen:

- addpath: Falls nicht im gleichen Ordner wie das Skript: Pfad der .dll und .h Datei
- ip: Die IP-Adresse des gewünschten ClipX Gerätes
- samplerate: Die gewünschte Samplerate (max. 1kHz, da der Fifo verwendet wird, siehe Korrespondenztabelle)
- signals: Die Signalindizes der gewünschten Signale die von ClipX übertragen werden sollen
- c: Anzahl der aufgenommenen Messwerte

```
%Loading the ClipXApi library.
if ~libisloaded('ClipXApi')
    %Enter the path of your ClipXApi here.
    %addpath('C:\Users\Siepmann\Desktop\Matlab')
    loadlibrary('ClipXApi.dll', 'ClipX_Interface.h');
end

%Create file to log data
file = fopen('MeasurementData.csv', 'w');

%Enter the IP address of ClipX
ip = '172.21.104.125';
h = calllib('ClipXApi', 'ClipX_Connect', ip);

idx = 17448;
sidx = 8;

%Enter the desired sample rate
samplerate = '1000';

calllib('ClipXApi', 'ClipX_SDOWrite', h, idx, sidx, samplerate);
[intres, readsamplerate] = calllib('ClipXApi', 'ClipX_SDORead', h, idx, sidx, 'abc', 10);

%Enter the desired signal indices
signals = [ 2 3 4 5 21 7 ];
[res] = signalSelector(signals, h);

calllib('ClipXApi','ClipX_startMeasurement', h);
pause(1);

%resptr = libpointer('doublePtr', zeros(1, 200));

%Amount of taken readings
c = 1000;

%Get signal names
name1 = getName(signals(1));
name2 = getName(signals(2));
name3 = getName(signals(3));
name4 = getName(signals(4));
name5 = getName(signals(5));
name6 = getName(signals(6));

```

Die Funktion „signalSelector“ stellt die gewünschten Signale für die Übertragung mit dem Fifo ein. Sofern diese Funktion nicht benötigt wird, kann sie einfach auskommentiert werden („%“).
In die Variablen name1 bis name6 werden die Signalnamen geschrieben. Diese können dann als Achsentitel in den Plots verwendet werden.

Visualisierungseinstellungen

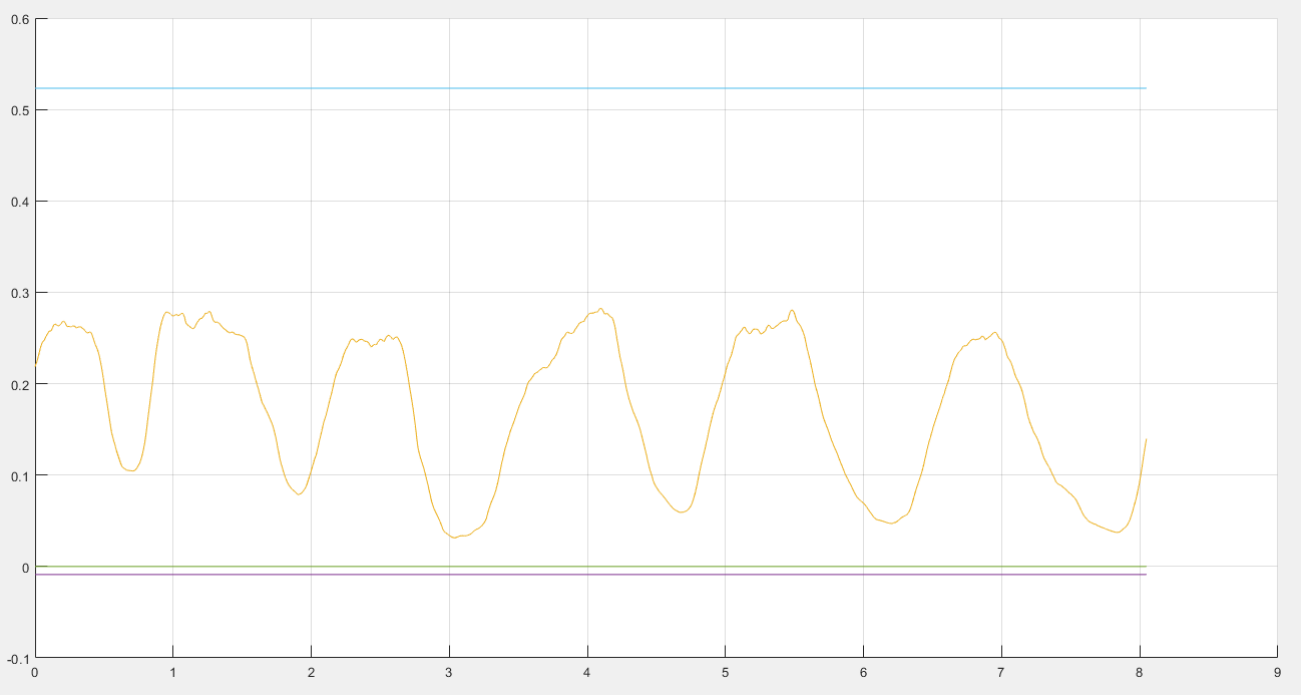
Als Visualisierung wurden in diesem Beispiel Plots verwendet. Der Graph zur Messung wird erst nach Abschluss der Messung erstellt.

```
figure
hold on
plot(timeplot, value1s)
plot(timeplot, value2s)
plot(timeplot, value3s)
plot(timeplot, value4s)
plot(timeplot, value5s)
plot(timeplot, value6s)
xlabel('Values')
ylabel('Time in s')
title('Measurement')
grid on
```

Messwertspeicherung

Die Messwerte werden in diesem Beispiel vollständig in den Vektoren times, value1s, value2s, value3s, value4s, value5s und value6s gespeichert. Zusätzlich werden alle Messwerte in eine .csv Datei und der komplette Workspace in der measurement.mat Datei gespeichert.

Eine Messung sieht beispielsweise wie folgt aus:



Korrespondenztabelle für Signalindizes

Index	Signal	Index	Signal
0	ADC Value	24	Calculated Value 4
1	Filtered ADC Value	25	Calculated Value 5
2	Field Value	26	Calculated Value 6
3	Gross Value	27	Ethernet API 1
4	Net Value	28	Ethernet API 2
5	Minimum Value	29	Fieldbus Value 1
6	Maximum Value	30	Fieldbus Value 2
7	Peak to Peak Value	31	Analog Out Value
8	Captured Value 1	32	Constant: -1
9	Captured Value 2	33	Constant: 0
10	ClipX Bus Value 1	34	Constant: 1
11	ClipX Bus Value 2	35	Constant: PI/2
12	ClipX Bus Value 3	36	Constant: PI
13	ClipX Bus Value 4	37	Constant: 2*PI
14	ClipX Bus Value 5	38	User Constant 1
15	ClipX Bus Value 6	39	User Constant 2
16		40	User Constant 3
17		41	User Constant 4
18		42	User Constant 5
19		43	User Constant 6
20		44	User Constant 7
21	Calculated Value 1	45	User Constant 8
22	Calculated Value 2	46	User Constant 9
23	Calculated Value 3	47	User Constant 10