TECH NOTE :: ClipX with Matlab

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

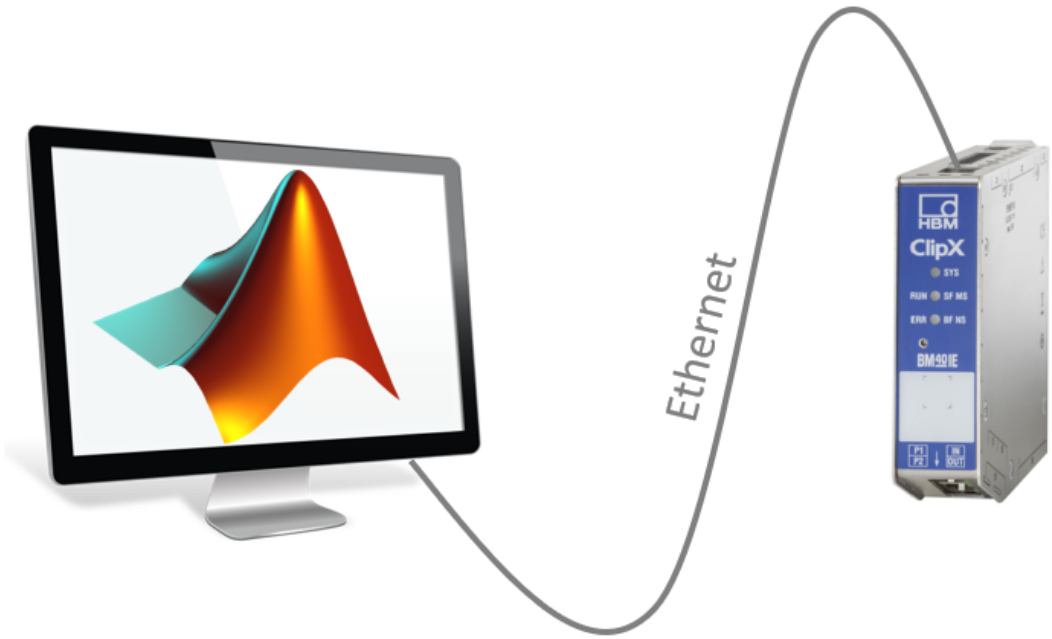
Short description

This in an instruction for using ClipX with the software environment Matlab. For the communication with ClipX the HBM ClipX API is used, which is based on an object dictionary communication and the ClipX fifo. ClipX uses a fixed sample rate of 19.2 kHz. The transfer rate of the values from ClipX to Matlab can be set from 0.1Hz to 1kHz (limited by the fifo).

This method uses a TCP/IP connection, so during a measurement with Matlab, no other device can connect to ClipX via port 55000. The use of the web server is still possible without restrictions, because this communication does not use the port 55000.
The following assumes that Matlab is already installed.

Important: For a correct representation of the signals of ClipX these must be plotted over its NTP time channel.

Note: Please make sure to use the latest API version of ClipX:
https://www.hbm.com/en/7077/clipx-precise-industrial-signal-conditioner/



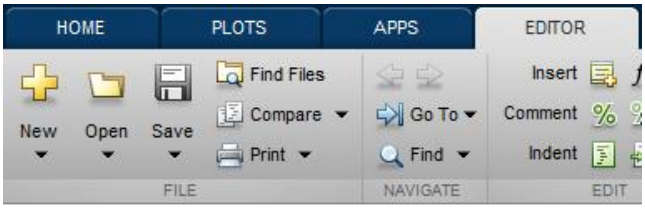
Content

This example includes the following files:

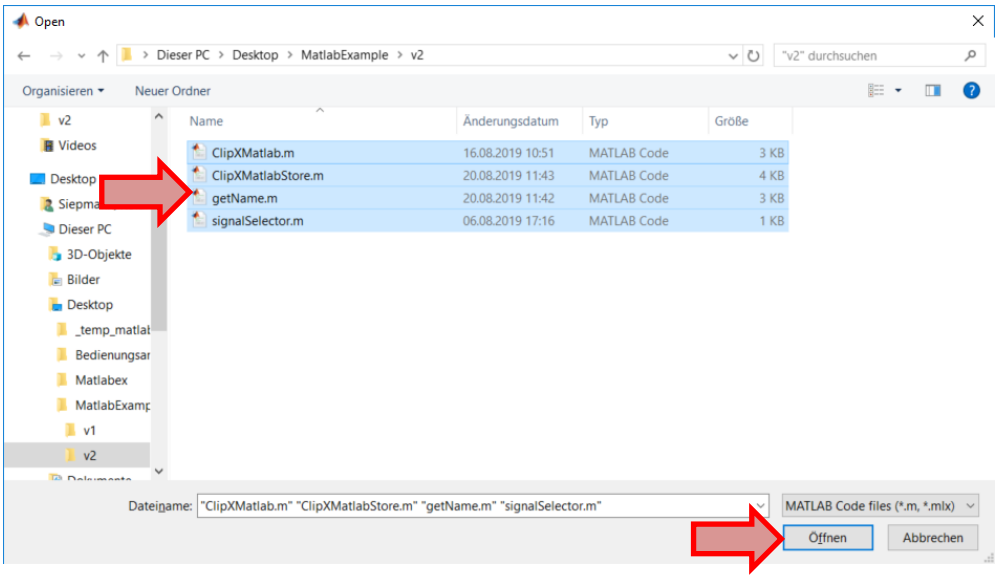
- Matlab script ‘ClipXMatlab.m’ which carries out the measurement
- Matlab script ‘ClipXMatlabStore.m’ for measurement which storing the data and a following plot
- Matlab function ‘signalSelector.m’ which sets up the signals that are stored in the fifo
- Matlab function ‘getName.m’ which provides the name for the corresponding signal indice

To carry out this example start Matlab.

- Select ‘Open’ in the menu bar



- Browse for the two Matlab files and open them



ClipXMatlab.m

This script carries out a measurement which a live plot. Therefore, the functions included in the .dll file are used.

Parameter setup

The following parameter must be adapted in the 'ClipXMatlab.m' file:

- addpath: If the API files are not located in the same folder as the Matlab files: API path
- ip: Select the ip address of your desired ClipX device
- samplerate: Select the desired sample rate
- signals: Select the signals you want to measure (see signal correspondences)
- c: Amount of taken readings

The function 'signalSelector' sets the desired signals that should be transmitted via the fifo storage. If this method is not needed, it can simply be out commended.

The function 'getName' returns the corresponding name for the given signal value.

Visualization settings

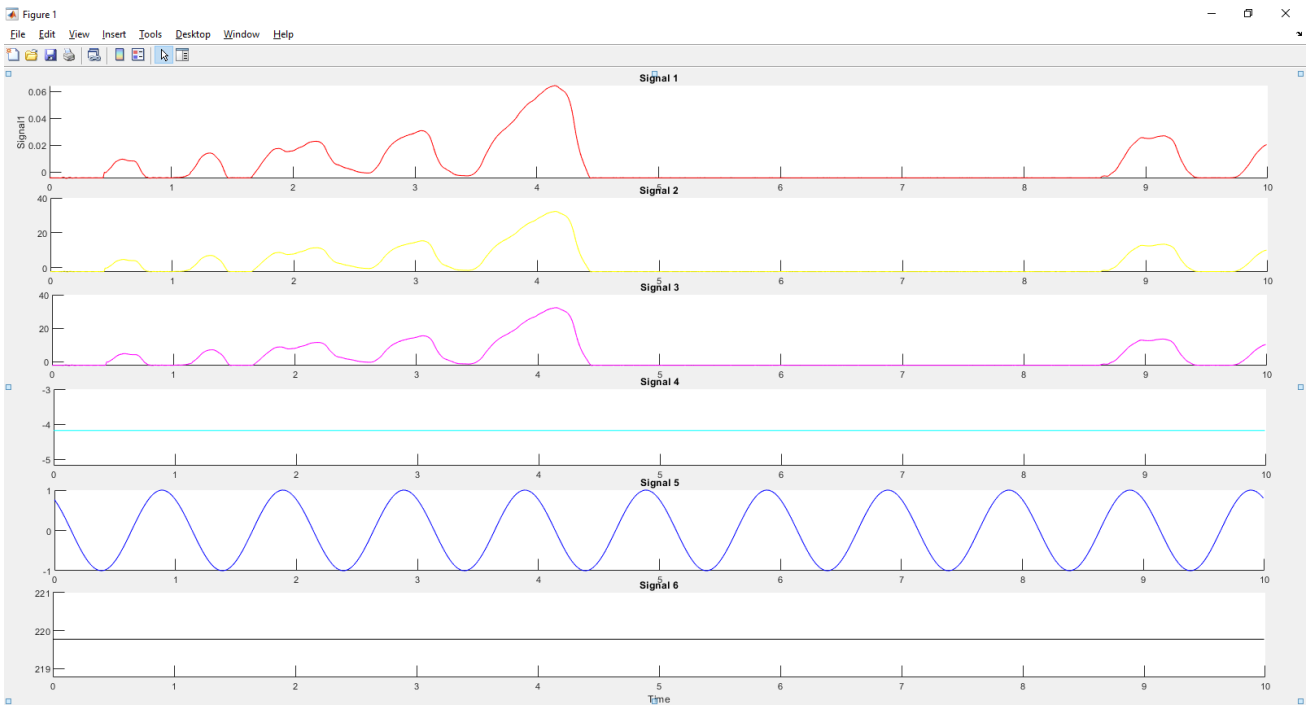
In this example, animated lines are chosen as the visualization object. These objects are continuously updated and so they work like a live graph. For sure, it can be taken any other visualization tool like e.g. the classic Matlab plot() function.

```
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')
```

Storing measurement values

In this example file, the measurement values are not stored cause this would influence the dynamic of the live plots negatively. If storing is desired, it can be implemented like it is done in ClipXMatlabStore.m

To carry out the measurement, simply run the script. For example, a measurement with the used visualization looks like this:



ClipXMatlabStore.m

This script carries out a measurement with storing the measurement values and plots them after the measurement. Therefore, the functions of the .dll file are used.

Parameter setup

The following parameter must be adapted in the 'ClipXMatlab.m' file:

- addpath: If the API files are not located in the same folder as the Matlab files: API path
- ip: Select the ip address of your desired ClipX device
- samplerate: Select the desired sample rate
- signals: Select the signals you want to measure (see signal correspondences)
- c: Amount of taken readings

```
%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));
```

The function 'signalSelector' sets the desired signals that should be transmitted via the fifo storage. If this method is not needed, it can simply be out commended.

The function 'getName' returns the corresponding name for the given signal value.

Visualization settings

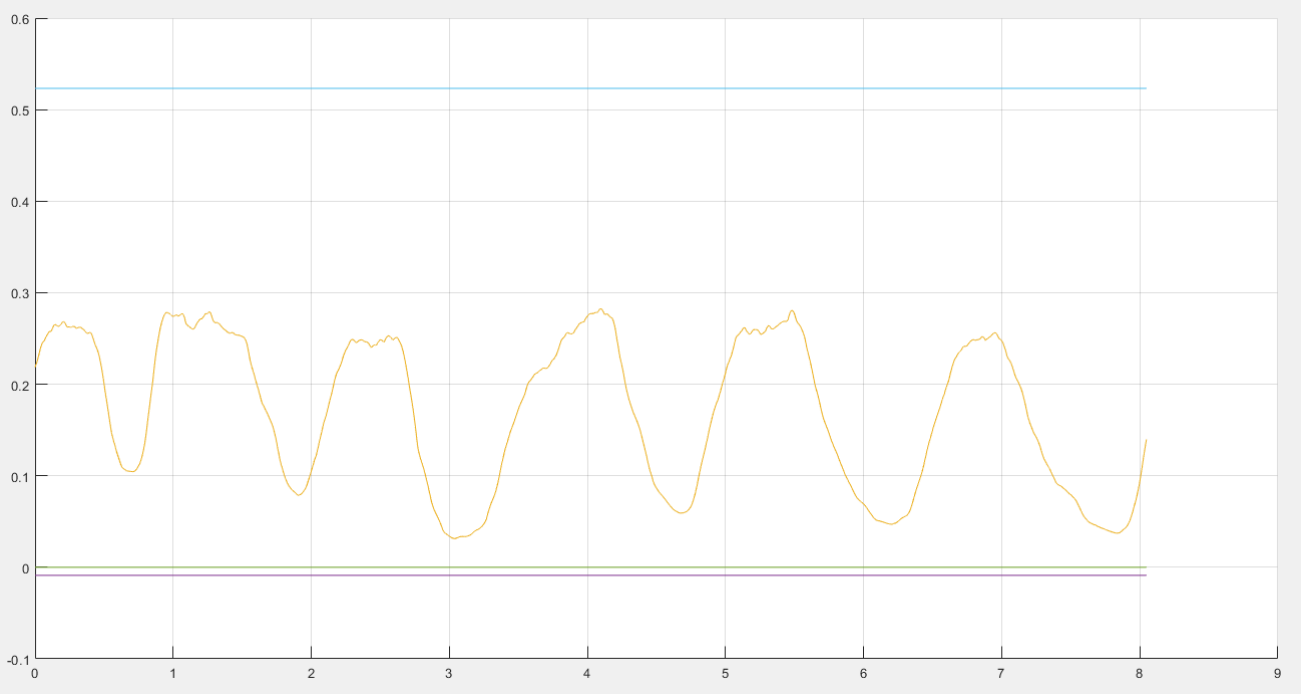
In this example, all six signals are visualized in one plot.

```
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
```

Storing measurement values

The measurement values are completely stored in the times, value1s, value2s, value3s, value4s, value5s and value6s vectors. Additionally, the values are stored in a .csv file and the whole workspace is stored in the measurement.mat file.

To carry out the measurement, simply run the script. For example, a measurement with the used visualization looks like this:



Signal correspondences

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

s

Disclaimer

These examples are for illustrative purposes only. They cannot be used as the basis for any warranty or liability claims.