TECH NOTE :: PMX with Matlab

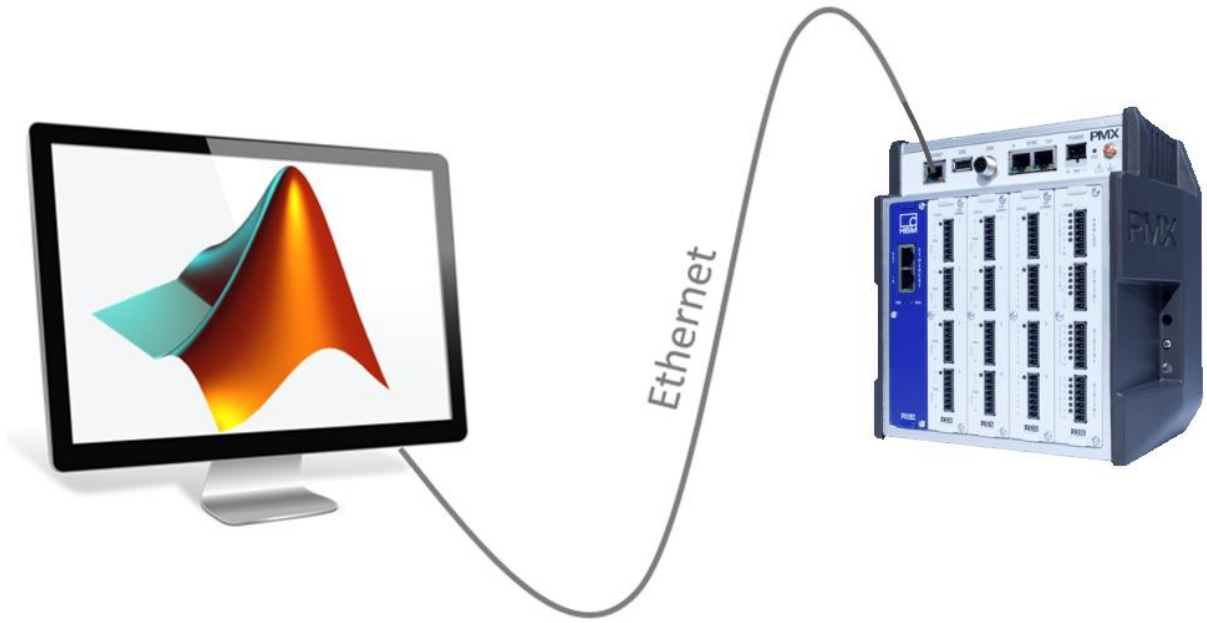
Version: 2019-09-12
Author: Michael Guckes, Roland Siepmann
Status: HBM: Public

Short description

This in an instruction for using PMX with the software environment Matlab. For the communication with PMX the TCP command interface of PMX is used. The commands are end manually to PMX via a socket connection and need to be adapted in Matlab.
The following assumes that Matlab is already installed.

Important: For a correct representation of the signals of PMX these must be plotted over its NTP time channel. Theoretically a measurement rate of 19.2kHz is possible. High measurement rates can cause performance problems depending from the system.

Note: Please make sure to use the latest example version of ClipX:
https://www.hbm.com/en/2981/pmx-modular-measuring-amplifier-system-for-the-iot/



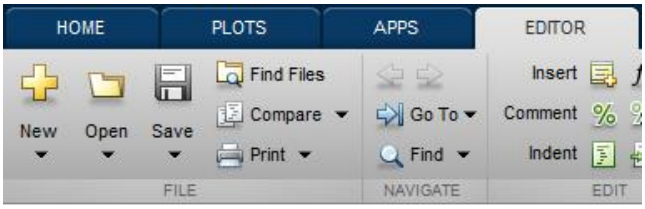
Content

This example includes the following files:

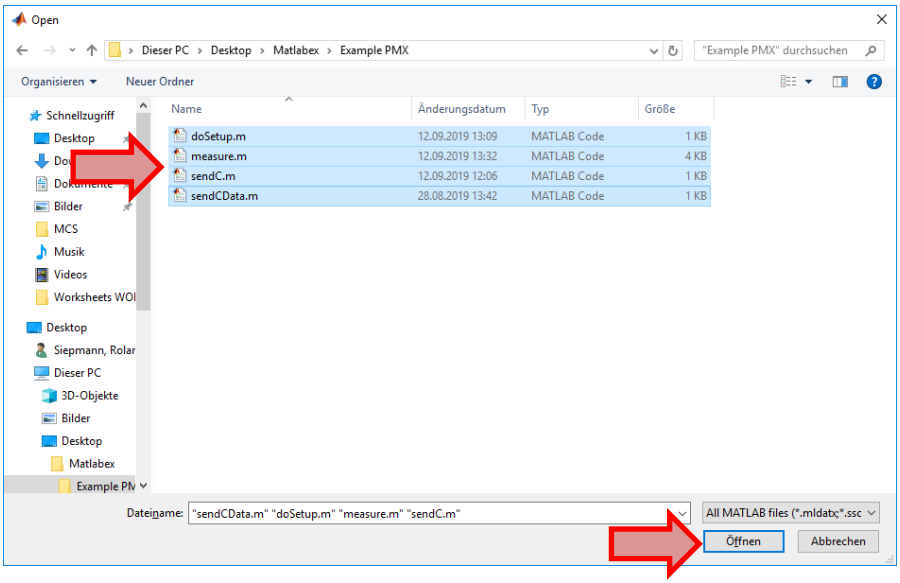
- Matlab script ‘measure.m’ which carries out the measurement
- Matlab function ‘doSetup.m’ which initializes the measurement
- Matlab function ‘sendC.m’ which sends commands to PMX
- Matlab function ‘sendCData.m’ reads the calculated amount of measurement data from the socket

To carry out this example start Matlab.

- Select ‘Open’ in the menu bar



- Browse for the two Matlab files and open them



doSetup.m

This function initializes the parameter for the measurement. For adapting the measured channels, rates and groups this file needs to be adapted.
In this example the calculated channel 9.1 is measured with a rate of 19.2kHz.

Hint: For the corresponding parameter please see the PMX manual (PMX command chapter).

```
function [sock] = doSetup(ip)
sock = tcpip(ip, 55000, 'NetworkRole', 'client');
sock.InputBufferSize = 90000000;
fopen(sock);

%%Set up measurement group 0
%Select measuring card
sendC(sock, 'PCS9');
%Select measuring channel
sendC(sock, 'SPS1');
%Select desired signal (gross, net,...)
sendC(sock, 'MSS214');
%Assign to measuring group (0,1,2)
sendC(sock, 'MRG0');

%Set up a card for measuring
sendC(sock, 'PCS9');
%Select channel
sendC(sock, 'SMS1');


%Select measuring channel
sendC(sock, 'MCS9,17');

%Set buffer format
sendC(sock, 'MBF1257,0');

%Set measuring rate for group 0
sendC(sock, 'ICR6345,0');

%Set time format (2 = milliseconds + seconds)
sendC(sock, 'STF2');

end
```

measure.m

This script carries out a measurement. In this example the measurement values are recorded and stored in arrays. After the measurement the values are plotted in a graph.

Parameter setup

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

- ip: Select the ip address of your desired PMX device
- maxchannels: Enter the amount of recorded measurement channels (excl. time)
- timechannels: Enter the amount of recorded time channels
- Values arrays: Add as many arrays as you measure values

```
1   %Edit IP address here
2 - ip = '172.21.64.40';
3
4   %transmitted measuring channels
5 - maxchannels = 1;
6
7   %transmitted time channels
8 - timechannels = 1;
9
10  %Sets up the measurement
11 - sock = doSetup(ip);
12
13  %Start measurement for group 0
14 - data1 = sendC(sock, 'TSV0');
15
16  %clock for measurement - change endtime for measurement duration
17 - acttime = clock;
18 - endtime = acttime(6) + 10;
19
20  %Calculation of memory requirement of a line
21 - datalineb = maxchannels*4 + timechannels*8;
22
23  %Value and time arrays (need to be added, if more signals are transmitted)
24 - recValues1 = [];
25 - recValues2 = [];
26 - recNTP = [];
```

The while loop carries out the data acquisition. First, the available lines and the measuring mode are requested. If measuring is true and lines are available, the available amount of data is calculated and read from the socket. In the for loop the data is assigned to the value arrays. Depending on how many values are measured, the red framed block has to be copied and assigned to another array (this is shown below).

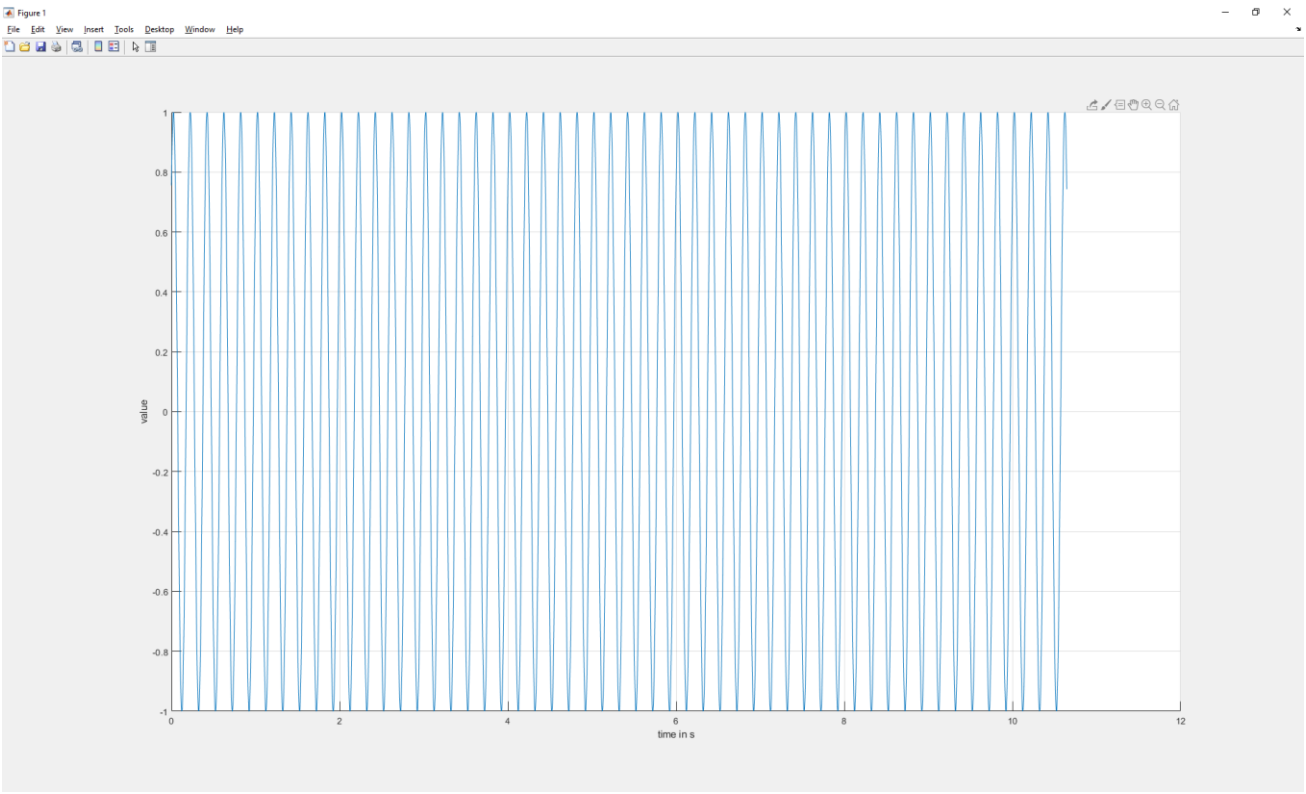
```
while acttime(6) < endtime
    datal = sendC(sock, 'OMP? 0');
    datal = split(datal, ',');
    lines = datal(1);
    msrmode = str2double(datal(2));
    if msrmode == 1
       if str2double(lines) > 0
           am = 4+(str2double(lines{1,1})*datalineb);
           data = sendCData(sock, char(strcat('RMB?', lines, ',6409,0')), am);
           noOfDataLines = (length(data)-4) / datalineb;
           bytePos = 3;

           for dataline = 1:1:(noOfDataLines)
                %{Read Measurement Values}%
                %Read Measurement Value1
                valueBuff = data(bytePos:(bytePos + 3));
                a = uint8(valueBuff);
                val = typecast(a, 'single');
                recValues1 = [recValues1 val];
                bytePos = bytePos + 4;

                %Read Measurement Value2
                %valueBuff = data(bytePos:(bytePos + 3));
                %a = uint8(valueBuff);
                %val = typecast(a, 'single');
                %recValues2 = [recValues2 val];
                %bytePos = bytePos + 4;

                %Add more measurement values here
```

Below the timestamp is put together. In this example the time format is chosen to seconds+microseconds. This can be adapted in the doSetup.m. After that the time is stored in the recNTP array.

```
58                  %{Read Time}%
59
60                  %%Optional Block to get the time in ticks (STF needs to be
61                  %%set to 0)
62                  %Get 64bit time
63                  %valueBuff = data(bytePos:(bytePos + 7));
64                  %a = uint8(valueBuff);
65                  %time = typecast(a, 'uint64');
66                  %time = double(time);
67                  %time = time*10^(-5);
68                  %bytePos = bytePos + 8;
69
70                  %Get ms time
71 -                valueBuff = data(bytePos:(bytePos + 3));
72 -                a = uint8(valueBuff);
73 -                timems = typecast(a, 'uint32');
74 -                bytePos = bytePos + 4;
75
76                  %Get s time
77 -                valueBuff = data(bytePos:(bytePos + 3));
78 -                a = uint8(valueBuff);
79 -                times = typecast(a, 'uint32');
80 -                bytePos = bytePos + 4;
81
82                  %Get doubles
83 -                timems = double(timems);
84 -                times = double(times);
85
86                  %Get exact time in s
87 -                timems = timems*10^(-6);
88 -                time = timems + times;
89
90                  %Add time to timearray
91 -                recNTP = [recNTP time];
```

Measurement

For measuring simly run the script. The measurement for a sine as calculated channel for example looks like this:



Disclaimer

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