DEVELOP CUSTOM TESTS WITH CLIO QC TCP/IP REMOTE CONTROL THROUGH SCILAB

by Daniele Ponteggia - dp@audiomatica.com

INTRODUCTION

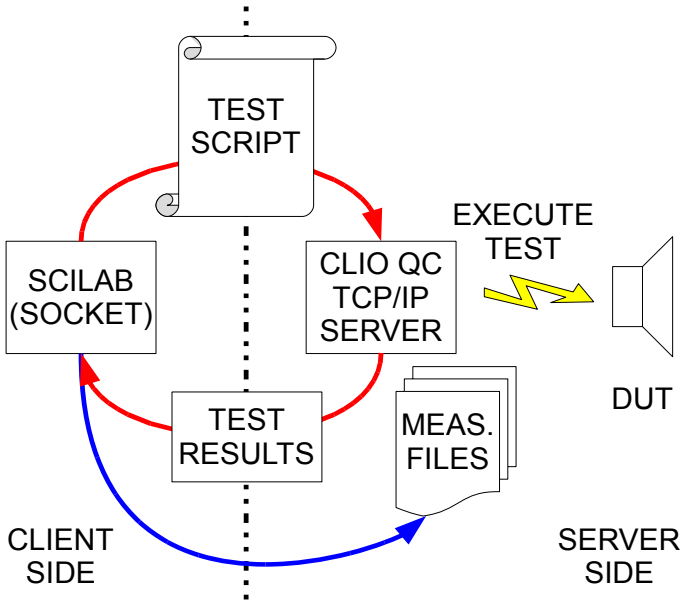
This document reports three examples of the powerful combination of the CLIO measurement system and the mathematical package SCILAB.

The CLIO software is already scriptable by means of the QC module. The QC module is specifically designed for the production line operations, nevertheless QC scripts can be used to perform R&D tests with some obvious limitations.

Within CLIO it is possible to invoke the QC module using TCP/IP, CLIO release 10 is greatly enhancing this functionality. The users manual already reports examples of the control the QC TCP/IP server using C++ programming. This approach is well suited in production applications but it can be cumbersome in the R&D usage.

The combination of the CLIO remote control through a SCILAB open endless possibilities to create custom test sequences for R&D use.

The operations are summarized in the following figure:

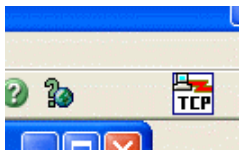


SCILAB sends a command script to CLIO QC using TCP/IP communication, CLIO executes the measurements and return the results via TCP/IP. Additional measurement binary or text files that are saved locally by CLIO can be read locally or via LAN by SCILAB.

In this document CLIO 10.03 QC and Scilab version 5.2.1 with Socket toolbox have been used. The installation of Scilab and the Socket toolbox is straightforward, please read the Scilab documentation.

SCILAB TO CLIO TCP/IP COMMUNICATION

The Socket toolbox adds to Scilab simple commands to manage the communication via TCP/IP.
The CLIO QC server must be started passing the the "TCP" parameter. Here is an example: "C:\Program Files\Audiomatica\CLIO 10\Clio.exe TCP". When running CLIO starts to listen on port defined in the CLIO Options>QC settings dialog (port 1234 default). The CLIO desktop shows this particular operating condition in the main toolbar with the TCP icon:



To open the communication with the CLIO QC TCP/IP server we will use the *SOCKET_open* command followed by *SOCKET_read*; note that in this case the server is *on the same machine* as the client and the address "localhost" is used.

```
-->SOCKET_open(1,"localhost",1234)

-->SOCKET_read(1)
 ans  =

  Welcome to Clio 8 QC Remote Server
```

To close the communication:

```
 -->SOCKET_close(1)
```

At this point, programming a Scilab script, it is possible to create CLIO QC scripts with an high complexity and flexibility.

EXAMPLE 1: MEASUREMENT OF THE THD VERSUS LEVEL

In this first example we measure the THD of a loudspeaker as a function of the SPL produced until the THD reaches a given limit.

The loudspeaker is feed with a 1kHz sinewave with increasing level. Using the CLIO Multimeter the THD and the SPL are measured using a microphone connected to the CLIO input.

A Multimeter measurement of the SPL with the correct settings, generator with a 1031.25 Hz sinewave included, must be carried out and saved as `multi.met`.

Then a limit file `limits.lim` must be supplied to let CLIO QC TCP/IP server to answer with the interesting measured data: THD and SPL.

```
[UPPER LIMIT DATA]
THD=100
PRESSURE=200
[LOWER LIMIT DATA]
THD=0.000001
PRESSURE=1
```

The QC test that must be invoked through the TCP/IP is:

```
[MET]
OUT=0.1 V
IN=0
REFERENCE=MULTI.MET
LIMITS=LIMITS.LIM
[]
```

The sequence of commands above can be sent to the QC Server as separate commands or in a single string.

In order to get a better understanding of the communication protocol, we report the communication between SCILAB and the QC Server in the case of separate commands, note that inside the *SOCKET_query* calls two backslash characters must be added at the beginning and the end of the command string (this is due to the TCL/TK interpreter of the SCILAB Socket Toolbox):

```
-->SOCKET_open(1,"localhost",1234)

-->SOCKET_read(1)
 ans  =

 Welcome to Clio 8 QC Remote Server

-->SOCKET_query(1,"\[MET]\")
 ans  =

 200 Start Command OK

-->SOCKET_query(1,"\OUT=0.1 V\")
 ans  =

 200 Additional Command OK

-->SOCKET_query(1,"\IN=0\")
 ans  =

 200 Additional Command OK

-->SOCKET_query(1,"\REFERENCE=MULTI.MET\")
 ans  =

 200 Additional Command OK

-->SOCKET_query(1,"\LIMITS=LIMITS.LIM\")
 ans  =

 200 Additional Command OK

-->MET=SOCKET_query(1,"\[]\")
 MET  =

!200 GOOD                        !
```

```
!                                !
!200 GOOD Pressure:112.40dBSPL   !
!                                !
!200 GOOD THD:1.495%             !
```

At the end of the script the CLIO QC server executes the measurement and return three strings with the test results. Using string manipulation commands in SCILAB is possible to extract the numerical values of the measurements:

```
-->[SPL,endstr]=strtod(part(MET(2),19:length(MET(2))))
 endstr  =

 dBSPL
 SPL  =

    121.97

-->[THD,endstr]=strtod(part(MET(3),14:length(MET(3))))
 endstr  =

 %
 THD  =

    5.171
```

The communication protocol implemented by the Socket Toolbox is not fast, and sending the commands separately using a sequence of *SOCKET_query* calls is not efficient. Grouping the commands in a single string is much more faster. To separate the commands we need to define the variable CRLF which contains the Carriage Return and Line Feed characters.

```
-->CRLF=char(13)+char(10);
```

Here is an example of a single string with all the commands needed, where we also added the `OVERLOADDETECT=1` command that returns input overload (this is currently an undocumented QC feature):

```
-->QCstring="\[MET]\"+CRLF+"\OUT="+string(Vout)+
"V\"+CRLF+"\IN=0\"+CRLF+"\REFERENCE=MULTI.MET\"+CRLF+"\LIMITS=LIMITS.LIM\
"+CRLF+"\OVERLOADDETECT=1\";

-->SOCKET_query(1,QCstring);
```

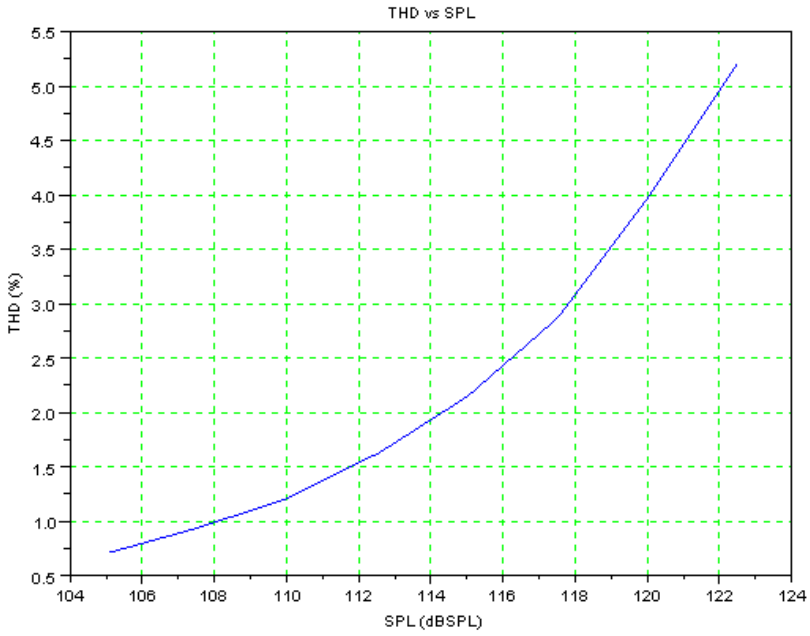
Now that the communication between SCILAB and CLIO QC TCP/IP Server is understood, we can create a complete SCILAB script that runs a series of tests increasing the CLIO output level.
The script takes care also of the input gain, with a simple autorange algorithm.

```
//TEST MOL SINGLE FREQUENCY
clear

//distorsion limit
THDlimit=5; //5%
```

```
//Vout starting value
Vout=0.05;

//boolean variable that indicates if the distorsion limit is reached
THDunderlimit=1;

//Open TCP communication
SOCKET_open(1,"localhost",1234);
SOCKET_read(1);

//Char CR+LF
CRLF=char(13)+char(10);

//measurement count index
i=0;

//start input range
IN=-40;

while THDunderlimit
  i=i+1;
  QCstring="\[MET]\"+CRLF+"\OUT="+string(Vout)+" V\"+CRLF;
  QCstring=QCstring+"\IN="+string(IN)+"\"+CRLF;
  QCstring=QCstring+"\REFERENCE=MULTI.MET\"+CRLF;
  QCstring=QCstring+"\LIMITS=LIMITS.LIM\"+CRLF;
  QCstring=QCstring+"\OVERLOADDETECT=1\";
  SOCKET_query(1,QCstring);
  ANS=SOCKET_query(1,"\[]\");
  [OVL,endstr]=strtod(part(ANS(4),14:length(ANS(4))));
  if OVL then
    //if overload then raise input gain and discard measurement misura
    i=i-1;
    IN=IN+10;
    if IN>40 then
      IN=40;
    end
  else
    //if not overload save measurement data
    [THD,endstr]=strtod(part(ANS(3),14:length(ANS(3))));
    [SPL,endstr]=strtod(part(ANS(2),19:length(ANS(2))));
    if THD>THDlimit then
      //if over the distorsion limit is reached set the flag
      THDunderlimit=0
    else
      //raise output level
      Vout=Vout.*1.33;
    end
    THDi(i)=THD;
    SPLi(i)=SPL;
  end
end
```

```
//close communication
SOCKET_close(1)

//draw graph
scf();
plot2d(SPLi,THDi,2);
title('THD vs SPL');
xlabel('SPL (dBSPL)');
ylabel('THD (%)');
a=gca();
a.box="on";
a.grid=[3,3];
```

At the end of the Scilab procedure, when the maximum distortion limit is reached, a chart is created:



The previous test is limited by the fact that in the QC operations the generator is linked with the measurement, and thus the frequency is fixed to the 1031.25 Hz.

EXAMPLE 2: MEASUREMENT OF THE MAXIMUM OUTPUT LEVEL (MOL) AS FUNCTION OF FREQUENCY

In this example we circumvent the previous limitation by creating a .wav file stimulus on the fly.

First of all a proper Multimeter test must be carried out and saved.

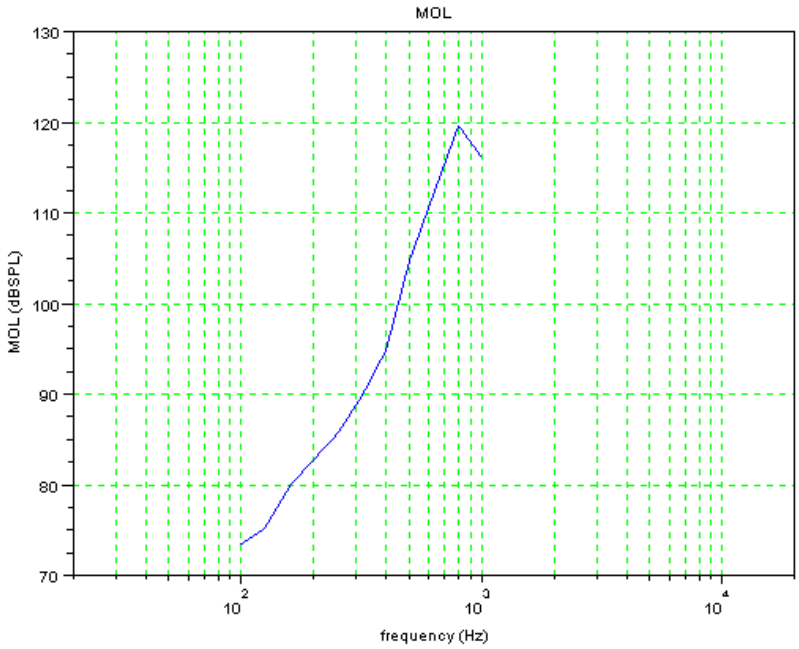
As a stimulus a proper wave file must be used. To create the .wav file, generate a 1031.25 Hz sine wave, then save the file as sine.wav (using the generator menu) and select the saved file as stimulus.

Here is the Scilab script code:

```
//TEST MOL
clear

//distosion limit
THDlimit=1.5;

//starting voltage output
Vstart=0.05;

//input range starting value
INstart=-20;

//open TCP communication
SOCKET_open(1,"localhost",1234);
SOCKET_read(1);

//char CR+LF definition
CRLF=char(13)+char(10);

//measurement frequencies nominal values
freqz=[100 125 160 250 315 400 500 630 800 1000];

for f=1:size(freqz,2) do
  //generate a sine.wav at the nearest frequency bin
  Fs=48000;
  METSize=16384;
  t=0:1/Fs:(METSize-1)/Fs;
  Fbin=round(freqz(f)/(Fs/METSize))*(Fs/METSize); //round to bin
  y=sin(2.*%pi.*Fbin.*t);
  wavwrite(y, Fs, 16, "sine.wav")
  //measurement index
  i=0;
  //output voltage
  Vout=Vstart;
  //input range
  IN=INstart;
  //distorsion limit flag
  THDunderlimit=1;
  while THDunderlimit
    //inc counter
    i=i+1;
    //test sequence
    QCstring="\[MET]\"+CRLF+"\OUT="+string(Vout)+" V\"+CRLF;
    QCstring=QCstring+"\IN="+string(IN)+"\"+CRLF;
    QCstring=QCstring+"\REFERENCE=MULTI.MET\"+CRLF;
    QCstring=QCstring+"\LIMITS=LIMITS.LIM\"+CRLF;
    QCstring=QCstring+"\OVERLOADDETECT=1\";
    SOCKET_query(1,QCstring);
    ANS=SOCKET_query(1,"\[]\");
    if isempty(ANS) then
      i=i-1;
    else
```

```
      [OVL,endstr]=strtod(part(ANS(4),14:length(ANS(4))));
      if OVL then
        //if overload raise input gain and discard measurement
        i=i-1;
        IN=IN+10;
        if IN>40 then
          IN=40;
        end
      else
        //if not overload save measurement data
        [THD,endstr]=strtod(part(ANS(3),14:length(ANS(3))));
        [SPL,endstr]=strtod(part(ANS(2),19:length(ANS(2))));
        if THD>THDlimit then
          //distorsion limit reached
          THDunderlimit=0
        else
          //inc out level
          Vout=Vout.*1.2;
        end
        THDi(i,f)=THD;
        SPLi(i,f)=SPL;
      end
    end
  end
end

//close communication
SOCKET_close(1)

//draw figure
scf();
plot2d(freqz,max(SPLi,'r'),2)
title('MOL');
xlabel('frequency (Hz)');
ylabel('MOL (dBSPL)');
a=gca();
a.box="on";
a.grid=[3,3];
a.log_flags="lnn";
a.data_bounds=[20,90;20000,140];
a.tight_limits="on";
```

At the end of this script we got two matrices `THDi(i,f)` and `SPLi(i,f)`, where we have the THD and the SPL as function of the output level and the frequency.



The above picture shows the MOL of a loudspeaker with THD>1.5%.

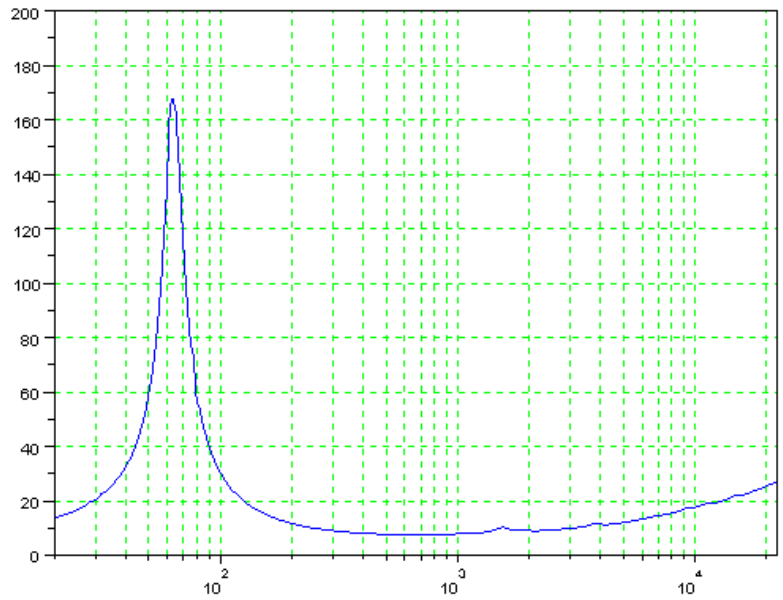
The algorithm used in the previous script is quite rough, but it is sufficient to illustrate the potential of the SCILAB plus CLIO QC TCP/IP approach.

EXAMPLE 3: MEASUREMENT OF LOUDSPEAKER IMPEDANCE

In this last example we will show the possibility to run a sinusoidal measurement using a QC script and then load the resulting measurement data into Scilab. The function `loadsin.sce` is described in the Application Note AN-001.

```
//TEST SINUSOIDAL
clear

//load the loadsin function
exec('loadsin.sce');

//open communication
SOCKET_open(1,"localhost",1234);
SOCKET_read(1);

//char CR+LF
CRLF=char(13)+char(10);

//Test
QCstring="\[SIN]\"+CRLF+"\OUT=-24 dBu\"+CRLF;
QCstring=QCstring+"\IN=-40\"+CRLF;
QCstring=QCstring+"\REFERENCE=SINUSOIDAL.SIN\"+CRLF;
QCstring=QCstring+"\LIMITS=NONE\"+CRLF;
QCstring=QCstring+"\SAVEONGOOD=1\";
SOCKET_query(1,QCstring);
```

```
ANS=SOCKET_query(1,"\[]\");

//wait for response
while SOCKET_read(1)==[]
end

//load sinusoidal file
[Ch,CHAUnit,CHBUnit,SinFreq,SinData,RNBFlag,RNBFreq,RNBData,THDFlag,THDFr
eq,THDData]=loadsin('tcpresult.sin');

//close communication
SOCKET_close(1)

//create figure
scf();
plot2d(SinFreq(1,:),abs(SinData(1,:)),2);
a=gca();
a.box="on";
a.grid=[3,3];
a.log_flags="lnn";
a.data_bounds=[20,0;22338,200];
a.tight_limits="on";
```



CONCLUSIONS

In this application note we showed the potential of controlling CLIO QC using the TCP/IP server functionality from SCILAB. Custom tests can be created with a minimal programming effort, using the high level and signal processing power of SCILAB and the CLIO measurement reliability.