TESTING SMART DEVICES WITH CLIO 12.5 QC

by Daniele Ponteggia – dp@audiomatica.com

INTRODUCTION

Here we present a proof-of-concept CLIO 12.5 QC application to test a seven microphone array which is typically used in smart devices such as Amazon Echo or Google Assistant.

This QC application exploits the new **WAV2SIN** measurement capability and **GRAPHREPORT** reporting mode of CLIO 12.5 QC.

DEVICE UNDER TEST

The device under test is a miniDSP UMA-8 USB mic array – V2.0[1].

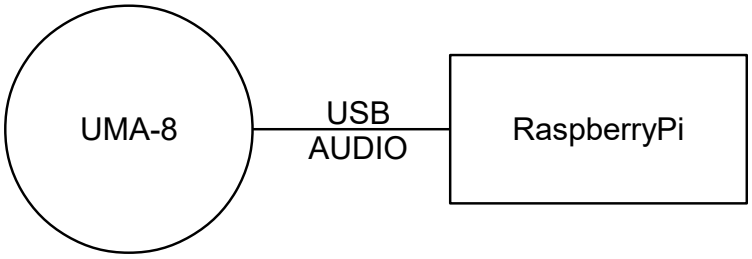
https://www.minidsp.com/products/usb-audio-interface/uma-8-microphone-array

The device has a circular form factor, with 7 MEMS microphones soldered to its device printed circuit board. The layout sees a microphone (MIC0) at the center of the board and 6 microphones on the outer circular edge (MIC1 to MIC6). The microphone array is connected to a microcontroller board which comes with a preloaded firmware with a beamforming and echo cancellation implementation. If connected to an USB port of a computer the device is seen as an USB audio device: a stereo microphone which is the result of the beamforming of the sound captured by the 7 microphone capsules.

1 The miniDSP UMA-8 microphone array is marketed in two variants: miniDSP UMA-8 and miniDSP UMA-8-SP. The difference lies in the latter being equipped with a digital power amplifier output. In this Part I the miniDSP UMA-8 is shown with a case. In Part II the miniDSP UMA-8-SP will be tested. If one is interested in both Part I and Part II only a miniDSP UMA-8-SP can be used.

DUT SETUP

In a typical consumer application this device is used as an input to a smart device: the mic array is connected to a single board computer such as a RaspberryPi which runs a light OS with installed the Amazon Alexa or Google Assitant application.



https://www.minidsp.com/applications/usb-mic-array/uma-8-rpi-diy-amazon-echo

https://www.minidsp.com/applications/usb-mic-array/uma-8-google-assistant

We would like to test the response of each individual microphone of the array, getting access to the output of each channel without the beamforming and echo canceling digital signal processing.

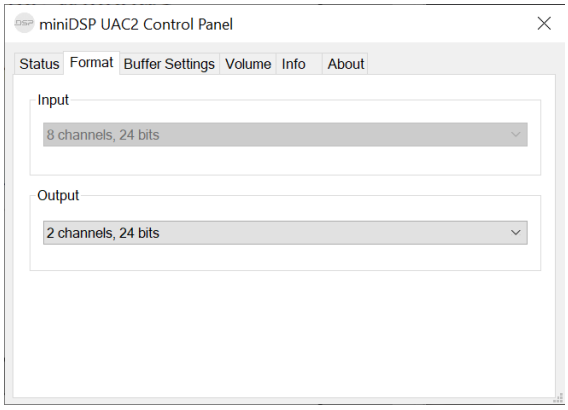
In the miniDSP UMA-8 device this is possible using a specific firmware. According to miniDSP documentation the available RAW mode firmware disables the DSP and allow the recording of each microphone separately.

Instruction on how to install the RAW mode firmware are available here:

https://www.minidsp.com/userdownloads/usb-mic-array-series/uma-8-drivers

Please be aware that in order to install the micArray_vf_raw_v1.0_up.bin firmware image the Windows UAC2 driver must be installed first on a Windows PC host. Then it is possible to run the miniDSP_UAC2_DFU_Win/miniDSPUAC2Dfu.exe flashing utility. Please follow instructions as reported in MiniDSP documentation.

At this point the MiniDSP UMA-8 should be seen from the Windows PC as 8 channels, 24 bit input device (microphones) with a 11025 Hz sample rate.



In this application we simulate the actual application where the microphone array is integrated on a device where it is typically connected to a single board computer.

Without loss of generality we use a RaspberryPi running a light Linux distribution (Raspbian) as a single board computer and simulate our entire application.

We now continue describing our actual implementation where we used a RaspberryPi 3 Model B, with a stock Raspbian image, with the following default credentials:

user: pi

pass: raspberrypi

The RaspberryPi should be set with ssh and Samba sharing enabled. Please refer to RaspberryPi documentation:

https://www.raspberrypi.org/documentation/remote-access/ssh/

https://www.raspberrypi.org/documentation/remote-access/samba.md

In our case we shared a subfolder of the pi user home/pi/Share/ as a shared resource named PiShare.

While not generally strictly necessary, we also installed SoX (Sound eXchange), in the RaspberryPi, in order to ease the audio splitting process as we will show later. To download SoX: http://sox.sourceforge.net/

The Raspberry IP has also been set as static with address[2] 192.168.0.71 and can be accessed via ssh shell[3]:

```
PS C:\Users\ponte> ssh pi@192.168.0.71
pi@192.168.0.71's password:
Linux raspberrypi 4.19.66-v7+ #1253 SMP Thu Aug 15 11:49:46 BST 2019 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Fri Sep 20 11:17:12 2019
Linux raspberrypi 4.19.66-v7+ #1253 SMP Thu Aug 15 11:49:46 BST 2019 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Fri Sep 20 11:17:12 2019
```

Now it is possible to connect the MiniDSP UMA-8 to one of the RaspberryPi free USB ports.

2 IP address should be set according to specific cases, in our case the local network we used is the 192.168.0.x range. It is also possible to directly connect the RaspberryPi to the PC by properly assigning IP addresses to both the Raspberry and PC.

3 During the first connection between RaspberryPi and the PC SSH a prompt requesting to accept a private key might appear.

Following instructions available in the MiniDSP site:

https://www.minidsp.com/applications/usb-mic-array/uma-8-google-assistant

It is possible to check from the RaspberryPi shell that the UMA-8 USB device is seen as ALSA recording device:

```
pi@raspberrypi:~ $ arecord -l
**** List of CAPTURE Hardware Devices ****
card 1: SpkUAC20 [miniDSP VocalFusion Spk (UAC2.0], device 0: USB Audio [USB
Audio]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
```

And also as ALSA playback device:

```
pi@raspberrypi:~ $ aplay -l
**** List of PLAYBACK Hardware Devices ****
card 0: ALSA [bcm2835 ALSA], device 0: bcm2835 ALSA [bcm2835 ALSA]
  Subdevices: 7/7
  Subdevice #0: subdevice #0
  Subdevice #1: subdevice #1
  Subdevice #2: subdevice #2
  Subdevice #3: subdevice #3
  Subdevice #4: subdevice #4
  Subdevice #5: subdevice #5
  Subdevice #6: subdevice #6
card 0: ALSA [bcm2835 ALSA], device 1: bcm2835 IEC958/HDMI [bcm2835
IEC958/HDMI]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 0: ALSA [bcm2835 ALSA], device 2: bcm2835 IEC958/HDMI1 [bcm2835
IEC958/HDMI1]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 1: SpkUAC20 [miniDSP VocalFusion Spk (UAC2.0], device 0: USB Audio [USB
Audio]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
```

In this specific application we will not test playback capabilities of the device, but this is possible using the same procedure we will use to test the microphones. Please note that the playback “card 0” is the internal Raspberry audio device, while our midiDSP device is “card 1”.

Now the .asoundrc ALSA setting file should be created and edited. The .asoundrc should be in your home directory /home/pi:

```
pcm.!default {
  type asym
  capture.pcm "mic"
  playback.pcm "speaker"
}
pcm.mic {
  type plug
  slave {
    pcm "hw:1,0"
  }
}
pcm.speaker {
  type plug
  slave {
    pcm "hw:0,0"
  }
}
```

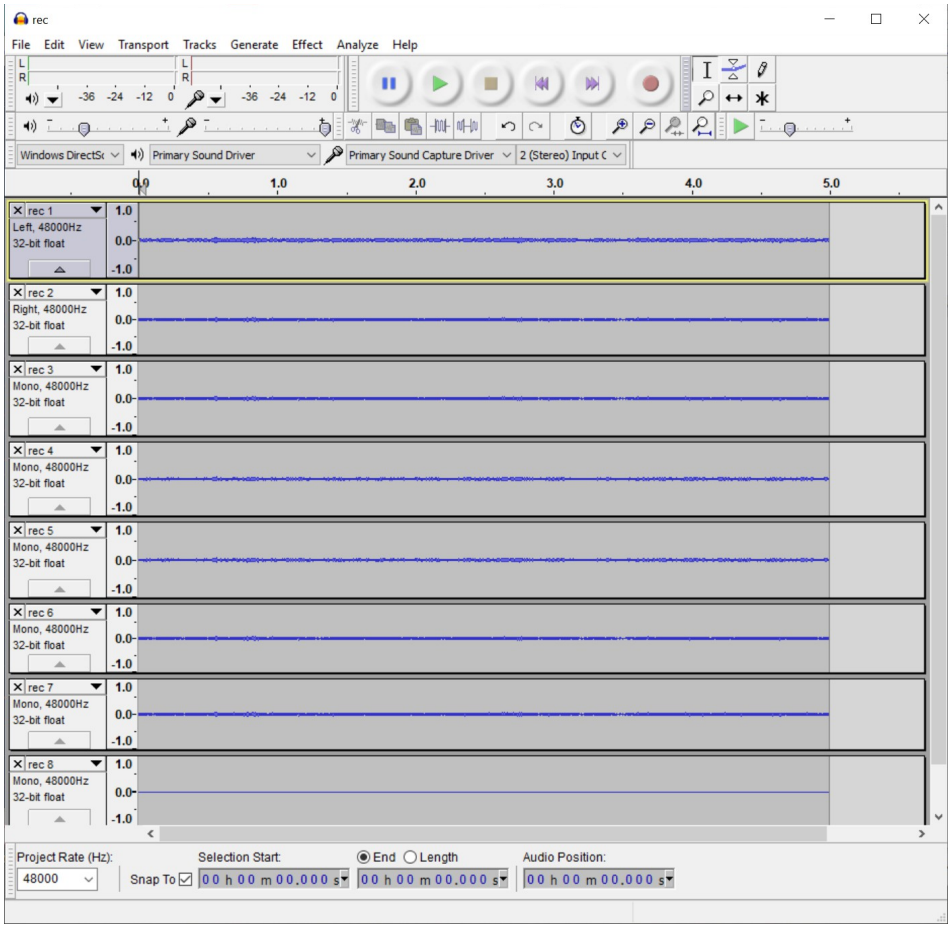
With a simple shell command like:

```
pi@raspberrypi:~ $ arecord --channels=8 --format=S16_LE --rate=48000 --file-
type=wav /home/pi/Share/recording.wav
```

It is possible to start the recording a 8 channel multichannel (--channels=8) 16 bit (--format=S16_LE) wave file recording.wav at 48 kHz sampling rate (--rate=48000):

The recording can be halted with CLTR+C or killing the arecord process.

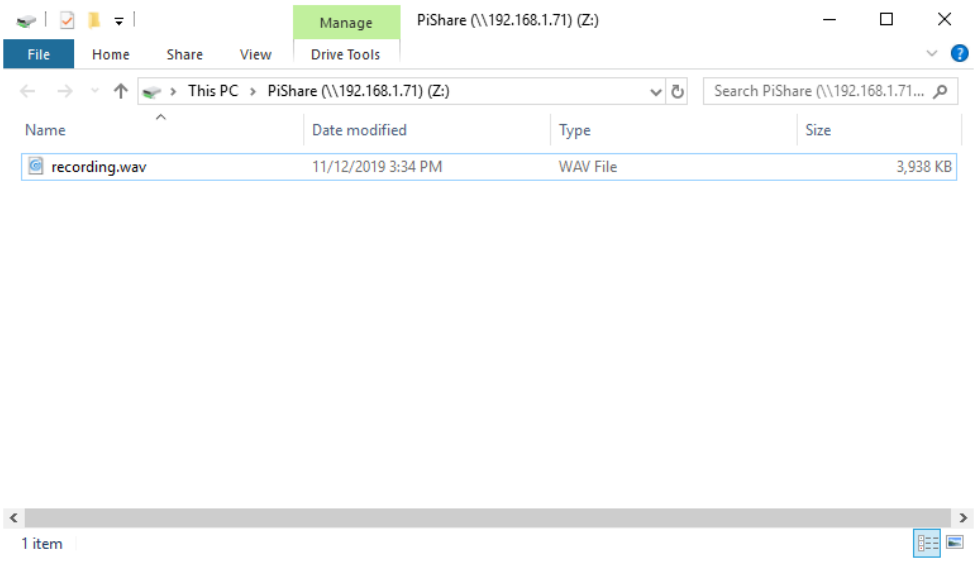
Accessing the shared resource the file can be opened, from a PC with access to the network, with an audio editor such as Audacity:



Please note that while the miniDSP UMA-8 microphone array has 7 MEMS channels in this example we recorded a total of 8 channels. The eighth channel is the output of a spare PDM port which has no transducer attached in the UMA-8 board.

We are now confident that the RaspberryPi+UMA-8 microphone is properly setup and able to record multichannel audio. We are also able to access the RaspberryPi shared resource from the PC where our CLIO 12 software runs.

Last thing to do is to map the shared resource as a network drive, in our example as Z:



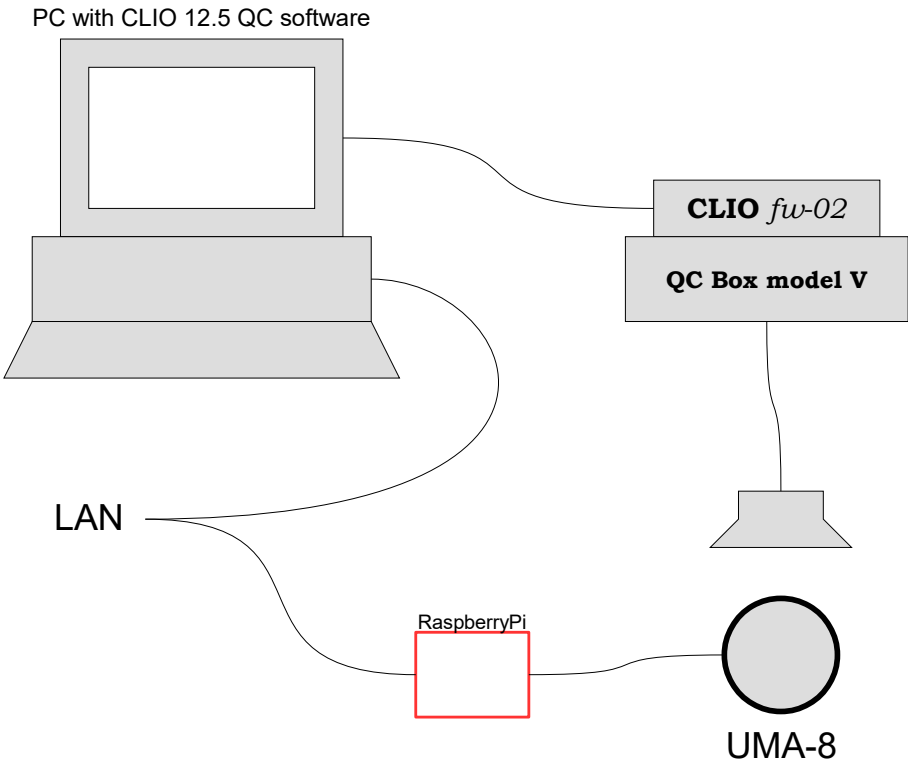
MEASUREMENT SETUP

In order to proceed with the test of our multichannel microphone device a proper setup is needed.

In our case we created a setup as follows:

- PC with CLIO 12.5 QC software connected to LAN
- CLIO fw-02 and QC Box Model V connected via USB
- CLIO fw-02 output A connected to QC Box Model V input
- Speaker connected to the QC Box Model V output
- RaspberryPi with Raspbian OS connected to LAN
- miniDSP UMA-8 (DUT) connected to RaspberryPi via USB

This setup is schematically reported in the following figure:



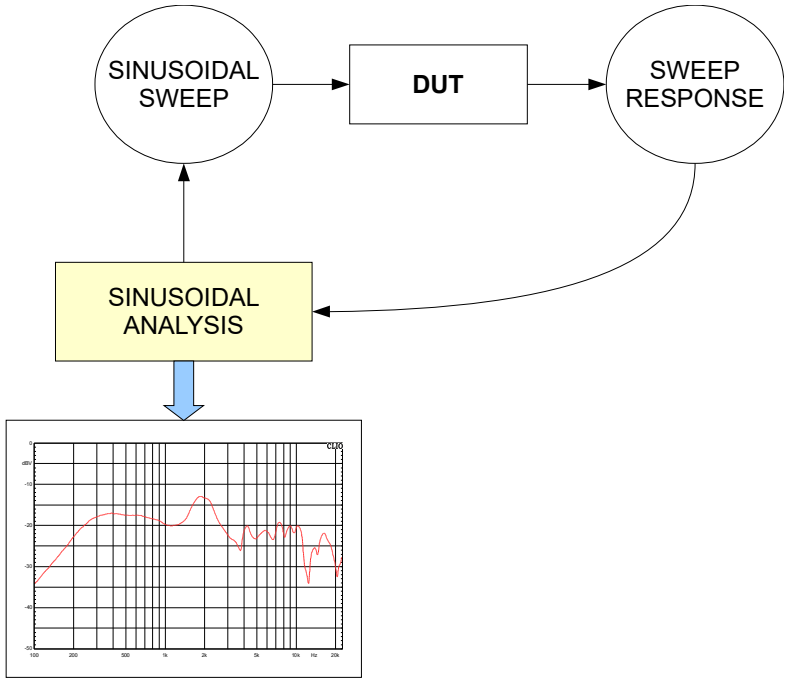
A suitable electro-acoustical transducer is required to reproduce the stimulus signal. In this application we used a small 2-inch loudspeaker housed in an 3d printed box connected to our Audiomatica QC Box Model V amplifier and CLIO fw-02 box.

As it can be seen from the previous picture there are no cables connected to CLIO fw-02 inputs.

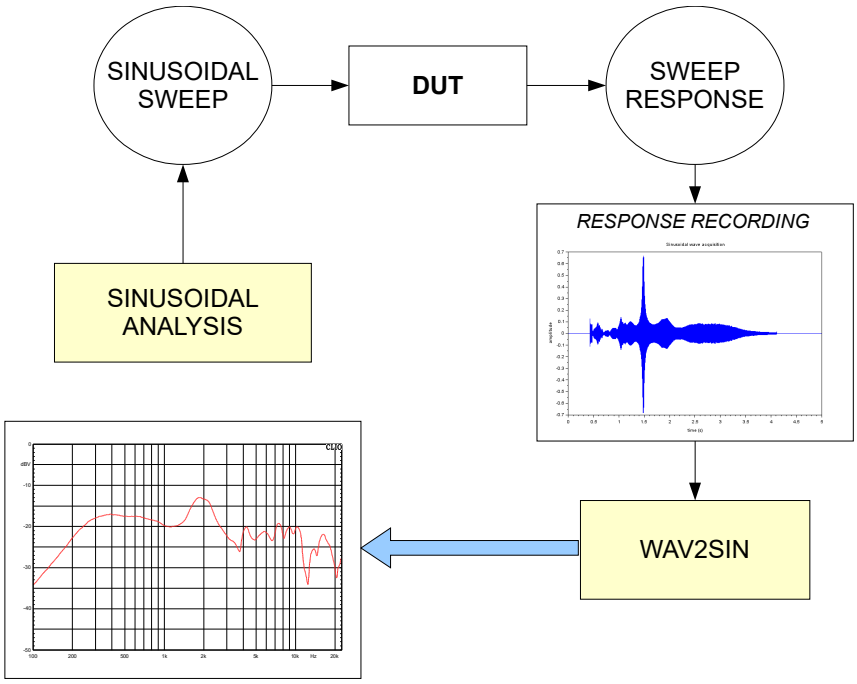
USING WAV2SIN

The WAV2SIN mode, introduced with CLIO 12.5, allows to perform **non real-time Sinusoidal analysis**. Instead of the simultaneous generation of stimulus signal and acquisition of DUT response. The DUT response to the sweep is saved in a .wav file and then analyzed when the .wav file recording is available.

Here is a scheme of the Sinusoidal analysis as implemented in CLIO. The measurement system generates a sinusoidal sweep which is acquired and analyzed in real time to get the frequency response of the DUT:



Following is a similar representation of the WAV2SIN. The sweep is generated and sent through the DUT, but instead of sent back to software for analysis is recorded as a wave file and analyzed at a later stage by WAV2SIN:



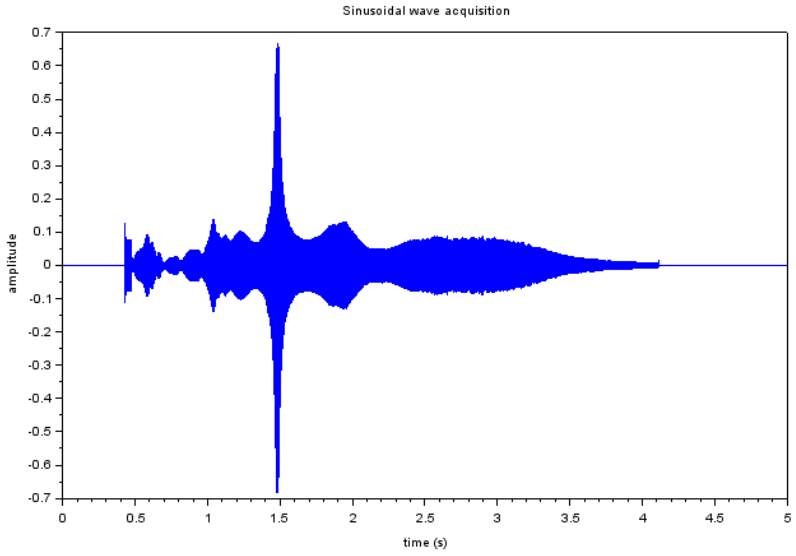
This a typical scenario in devices which are not featuring direct access to analog audio input and/or outputs but are able to provide recording and eventually playback functions through wave files. Our RaspberryPi smart device with UMA-8 microphone is one of these devices.

As we saw before we were able to record the multichannel audio from the microphones in a wave file. The idea now is to record the response to the Sinusoidal stimulus in a wave file, put it in a shared folder with the CLIO12 host PC, and then access the wave file and use WAV2SIN to get the frequency response.

The recording can be done manually by:

- starting the recording process on the RaspberryPi,
- launching a Sinusoidal Test with CLIO
- manually halting the recording.

As a result we should have a recording similar to this:



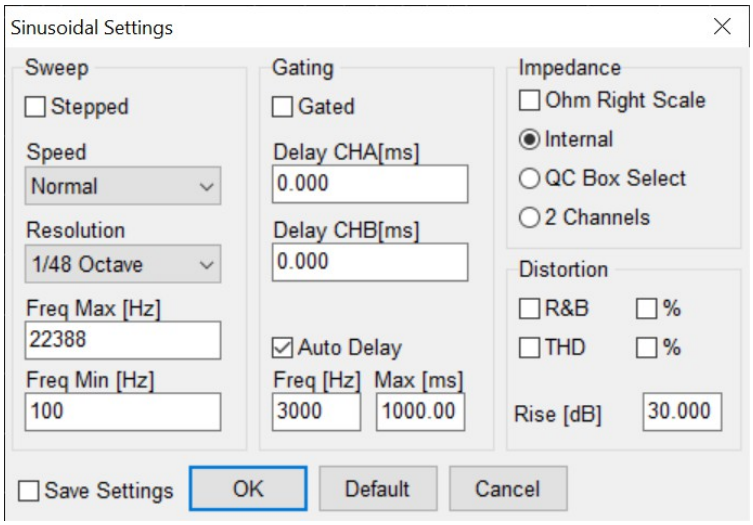
We also would like a more simple and streamlined process which does not require manual user intervention.

We will use here a QC script which implements all the necessary steps: starts the recording on the remote device, runs the Sinusoidal measurement, halts the recording, copies the wave file in a local folder in the CLIO 12 host PC and finally runs the WAV2SIN analysis.

At the end of this process we will have a non real-time sinusoidal frequency response test of our remote DUT.

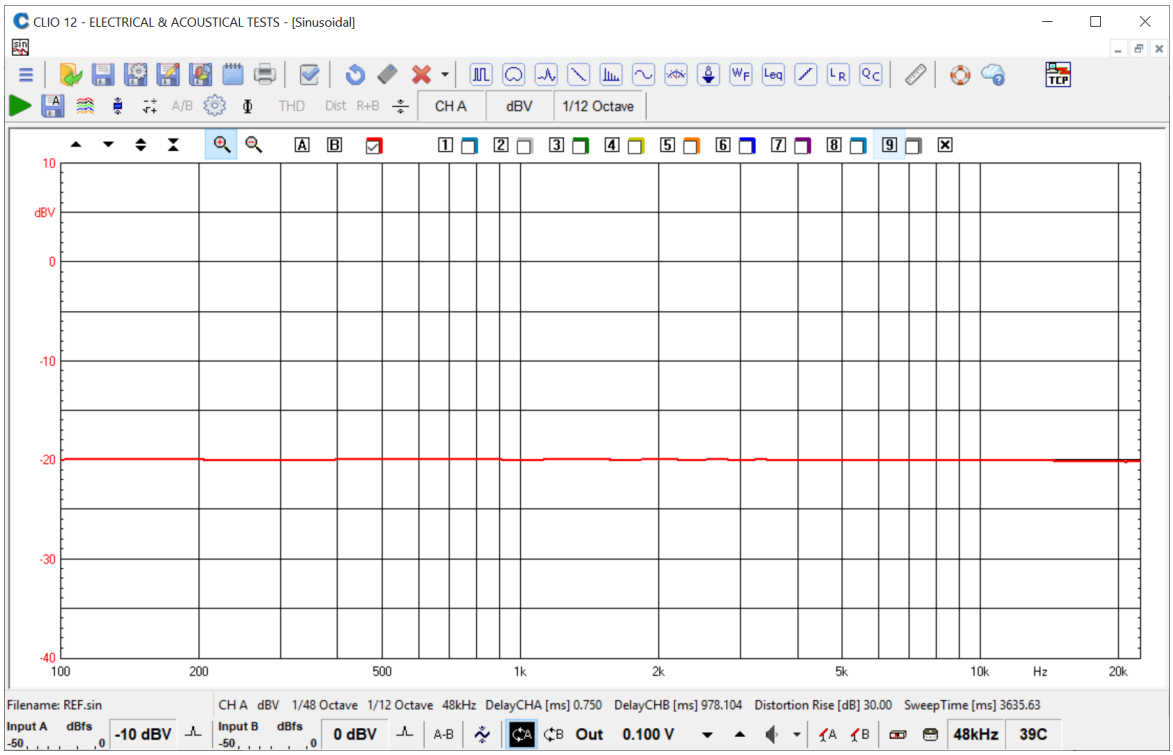
First of all a **reference file** which stores all the Sinusoidal measurement settings should be created. In order to create such Sinusoidal file it is advisable to set up the CLIO fw-02 channel A in loop.

In our example we are using the following settings:



where the **Auto Delay must be selected**: the WAV2SIN mode uses the auto-delay pilot tone to find the time reference during the post-processing of the wave file. The Auto Delay frequency should be set to be well inside the linear frequency response range of the DUT response and Auto Delay Max [ms] time should be set to its maximum value 1000. Please note that also the smoothing settings and frequency limits of the visualization which will be shown during the QC procedures will be the same of this reference file.

Running the loop measurement we should get the following response:



Note the smoothing factor of 1/12 octave which is one of the interactive settings which have to be set outside the Sinusoidal settings dialog.

We will now save this file in a folder as a REF.sin and use for our next non real-time WAV2SIN Sinusoidal test.

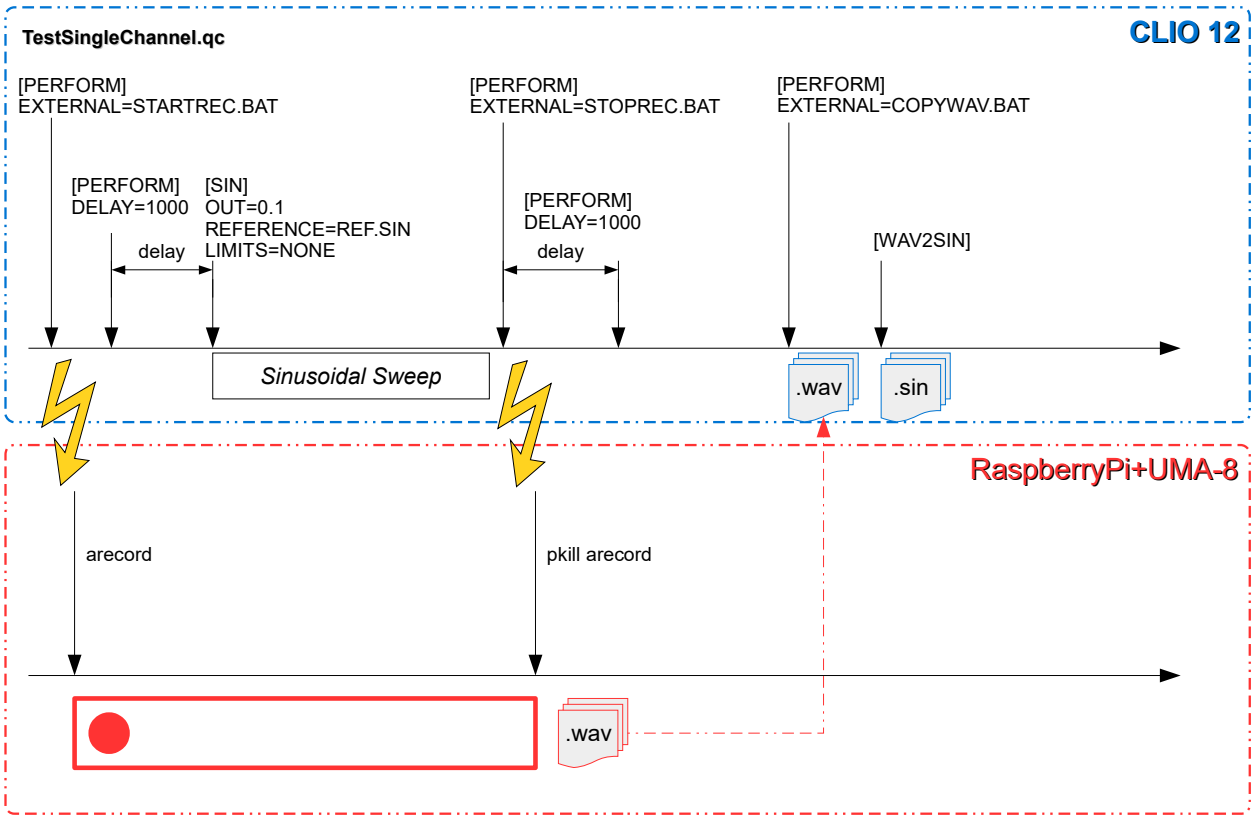
Before proceeding further we need to install in our CLIO 12 host PC the plink.exe utility which greatly eases the communication via SSH with the RaspberryPi. The plink command is a command line utility which is freely available within the PuTTY application suite, this command allows to send an SSH script without the need to manually insert a password.

https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html

A possible QC Script to implement the WAV2SIN measurement is as follows:

```
[PERFORM]
EXTERNAL=STARTREC.BAT
[PERFORM]
DELAY=1000
[SIN]
OUT=0.1 V
REFERENCE=REF.SIN
LIMITS=NONE
[PERFORM]
EXTERNAL=STOPREC.BAT
[PERFORM]
DELAY=1000
[PERFORM]
EXTERNAL=COPYWAV.BAT
WAITCOMPLETION=1
[WAV2SIN]
WAVFILEINPUT=ch1.wav
REFERENCE=REF.sin
LIMITS=NONE
```

Let's follow the script with the aid of a scheme:



The script starts with a first section which calls a batch file:

```
[PERFORM]
EXTERNAL=STARTREC.BAT
```

This is the content of STARTREC.BAT:

```
@echo off
plink pi@192.168.0.71 -pw raspberrypi -batch arecord --channels=1
--format=S16_LE --rate=48000 --file-type=wav /home/pi/Share/ch1.wav
```

which will remotely start the arecord command on the RaspberryPi.

Then there is a delay section to allow the recording on the remote device to start:

```
[PERFORM]
DELAY=1000
```

At this point the Sinusoidal test is launched using the REF.sin as a reference file:

```
[SIN]
OUT=0.1 V
REFERENCE=REF.sin
LIMITS=NONE
```

this Sinusoidal test is only to be meant as a stimulus for the DUT, the DUT response is not sent to CLIO inputs, but instead remotely recorded on a wave file.

The results of this Sinusoidal test are in fact meaningless since there is no signal routed to CLIO input.

As soon as the Sinusoidal test ends, the recording should be stopped by killing the arecord process:

```
[PERFORM]
```

```
EXTERNAL=STOPREC.BAT
```

The STOPREC.BAT content is:

```
@echo off
plink pi@192.168.0.71 -pw raspberrypi -batch pkill arecord
```

where the arecord process is killed.

A small delay is introduced to allow the recording to stop and the .wav file to be available in the shared folder.

```
[PERFORM]
DELAY=1000
```

Then the wave file is copied to the CLIO 12 host PC through the COPYWAV.BAT:

```
[PERFORM]
EXTERNAL=COPYWAV.BAT
WAITCOMPLETION=1
```

Content of the COPYWAV.BAT file is a simple file copy command:

```
copy z:\ch1.wav ch1.wav
```

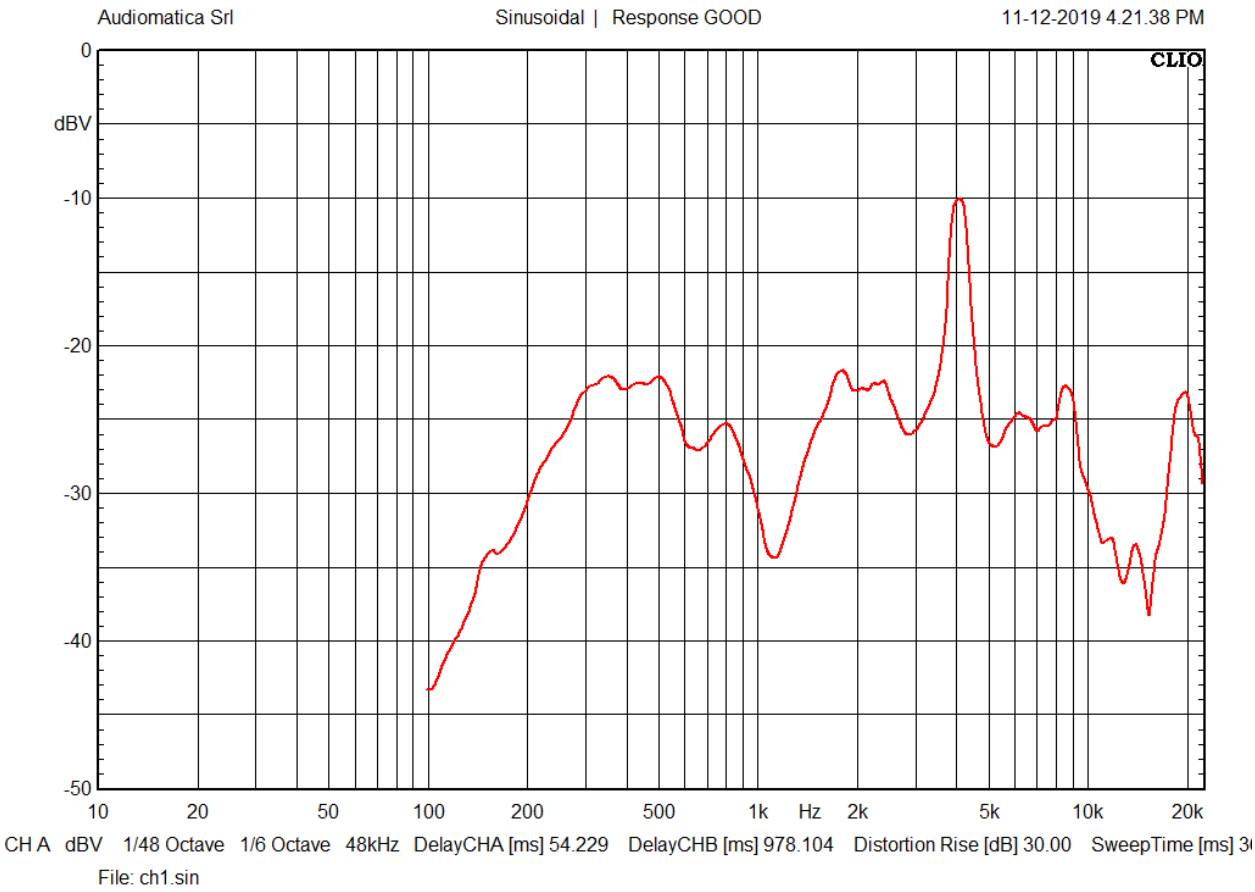
Now the ch1.wav file, which is the response of the DUT to the Sinusoidal sweep test signal is available in the CLIO QC working directory.

All is set for the WAV2SIN test:

```
[WAV2SIN]
WAVFILEINPUT=ch1.wav
REFERENCE=REF.sin
LIMITS=NONE
```

The response of the device under test, in this case the first channel of our miniDSP UMA-8 microphone, is shown in the QC window. The resulting frequency response sinusoidal file is saved in the ch1.sin file, with the same name as the WAVEINPUT file but with .sin extension.

A detailed inspection of the measurement results as in ch1.sin file is possible by pressing the “release measurements” button from the CLIO QC interface.



This ch1.sin file can be used for any purpose, such as simply inspecting the remote device response or using the file as a reference for a QC script to test the same device.

COMPLETE TEST OF AN 8 CH MICROPHONE

Now that the WAV2SIN usage has been introduced it is simple to extend the same concept to a multichannel approach.

In a similar fashion to what we have done for the single channel we can remotely send to the RaspberryPi device a multiple channel recording command, execute a sinusoidal test, separate the recording in a set of single channel wave files and finally set up a series of WAV2SIN tests; one for each channel.

It is possible to combine the previous commands into a script record.sh which can be stored in a file into the Pi user's home folder which implements the above sequence of operations:

```
cd /home/pi/Share/
arecord --channels=8 --format=S16_LE --rate=48000 --file-type=wav rec.wav
```

This shell script will be run by CLIO 12.5 QC through a batch file when needed.

The following QC script has been used:

```
[GLOBALS]
GRAPHREPORT=1
TILEWINDOWS=0
[PERFORM]
EXTERNAL=STARTRECMC.BAT
[PERFORM]
```

```
DELAY=1000
[SIN]
OUT=0.050 V
INA=0
INB=0
REFERENCE=REF.SIN
LIMITS=none
NOGRAPHREPORT=1
[PERFORM]
EXTERNAL=STOPRECMC.BAT
[PERFORM]
DELAY=1000
[PERFORM]
EXTERNAL=COPYWAVMC.BAT
WAITCOMPLETION=1
[WAV2SIN]
GRAPH TITLE=MICROPHONE 1
WAVFILEINPUT=ch1.wav
REFERENCE=REF.SIN
LIMITS=ch1.LIM
[WAV2SIN]
GRAPH TITLE=MICROPHONE 2
WAVFILEINPUT=ch2.wav
REFERENCE=REF.SIN
LIMITS=ch1.LIM
[WAV2SIN]
GRAPH TITLE=MICROPHONE 3
WAVFILEINPUT=ch3.wav
REFERENCE=REF.SIN
LIMITS=ch1.LIM
[WAV2SIN]
GRAPH TITLE=MICROPHONE 4
WAVFILEINPUT=ch4.wav
REFERENCE=REF.SIN
LIMITS=ch1.LIM
[WAV2SIN]
GRAPH TITLE=MICROPHONE 5
WAVFILEINPUT=ch5.wav
REFERENCE=REF.SIN
LIMITS=ch1.LIM
[WAV2SIN]
GRAPH TITLE=MICROPHONE 6
WAVFILEINPUT=ch6.wav
REFERENCE=REF.SIN
LIMITS=ch1.LIM
[WAV2SIN]
GRAPH TITLE=MICROPHONE 7
WAVFILEINPUT=ch7.wav
REFERENCE=REF.SIN
LIMITS=ch1.LIM
[WAV2SIN]
GRAPH TITLE=MICROPHONE 8
WAVFILEINPUT=ch8.wav
REFERENCE=REF.SIN
LIMITS=none
```

The STARTRECORDMC.BAT is a batch file in the QC working directory which is used to start a multichannel audio recording in the smart device:

```
@echo off
plink pi@192.168.0.71 -pw raspberrypi -batch arecord --channels=8
--format=S16_LE --rate=48000 --file-type=wav /home/pi/Share/rec.wav
```

Let's analyze in detail the QC script, at the top of the script in the [GLOBALS] section we find:

```
[GLOBALS]
GRAPHREPORT=1
TILEWINDOWS=0
```

where the new GRAPHREPORT=1 mode is selected. This is a new graphical reporting mode of CLIO QC which allows to use multiple graphics in a single measurement window. For details on GRAPHREPORT usage please check CLIO QC user's manual.

Then the recording is started on the remote device:

```
[PERFORM]
EXTERNAL=STARTRECMC.BAT
[PERFORM]
DELAY=1000
```

And the Sinusoidal test is run:

```
[SIN]
OUT=0.100 V
INA=0
INB=0
REFERENCE=REF.SIN
LIMITS=none
NOGRAPHREPORT=1
```

Once the sinusoidal sweep is done the recording is stopped:

```
[PERFORM]
EXTERNAL=STOPRECMC.BAT
[PERFORM]
DELAY=1000
```

Now the recorded rec.wav should be separated in eight mono wave files, one for each channel. Thanks to the fact that we are running a simple but complete Linux distribution on our RaspberryPi it is possible to use the SoX utility to separate the multichannel wave file into 8 separated mono wave files[4].

This can be done using the SoX remix command, as an example the first channel can be extracted from the multichannel wave recording using the following syntax:

```
sox rec.wav ch1.wav remix 1
```

Once the files are separated in a set of mono wave files then they can be copied to the CLIO QC working directory on the CLIO host PC. This can be accomplished by another batch file which does the trick:

```
[PERFORM]
EXTERNAL=COPYWAVMC.BAT
WAITCOMPLETION=1
```

The COPYWAVMC.BAT should be:

```
@echo off
plink pi@192.168.0.71 -pw raspberrypi -batch sox rec.wav ch1.wav remix 1
```

4 This is not a critical constraint as the conversion of the multichannel wave file can be also carried out in the CLIO QC host machine.

```
plink pi@192.168.0.71 -pw raspberrypi -batch sox rec.wav ch2.wav remix 2
plink pi@192.168.0.71 -pw raspberrypi -batch sox rec.wav ch3.wav remix 3
plink pi@192.168.0.71 -pw raspberrypi -batch sox rec.wav ch4.wav remix 4
plink pi@192.168.0.71 -pw raspberrypi -batch sox rec.wav ch5.wav remix 5
plink pi@192.168.0.71 -pw raspberrypi -batch sox rec.wav ch6.wav remix 6
plink pi@192.168.0.71 -pw raspberrypi -batch sox rec.wav ch7.wav remix 7
plink pi@192.168.0.71 -pw raspberrypi -batch sox rec.wav ch8.wav remix 8
copy z:\ch1.wav ch1.wav
copy z:\ch2.wav ch2.wav
copy z:\ch3.wav ch3.wav
copy z:\ch4.wav ch4.wav
copy z:\ch5.wav ch5.wav
copy z:\ch6.wav ch6.wav
copy z:\ch7.wav ch7.wav
copy z:\ch8.wav ch8.wav
```

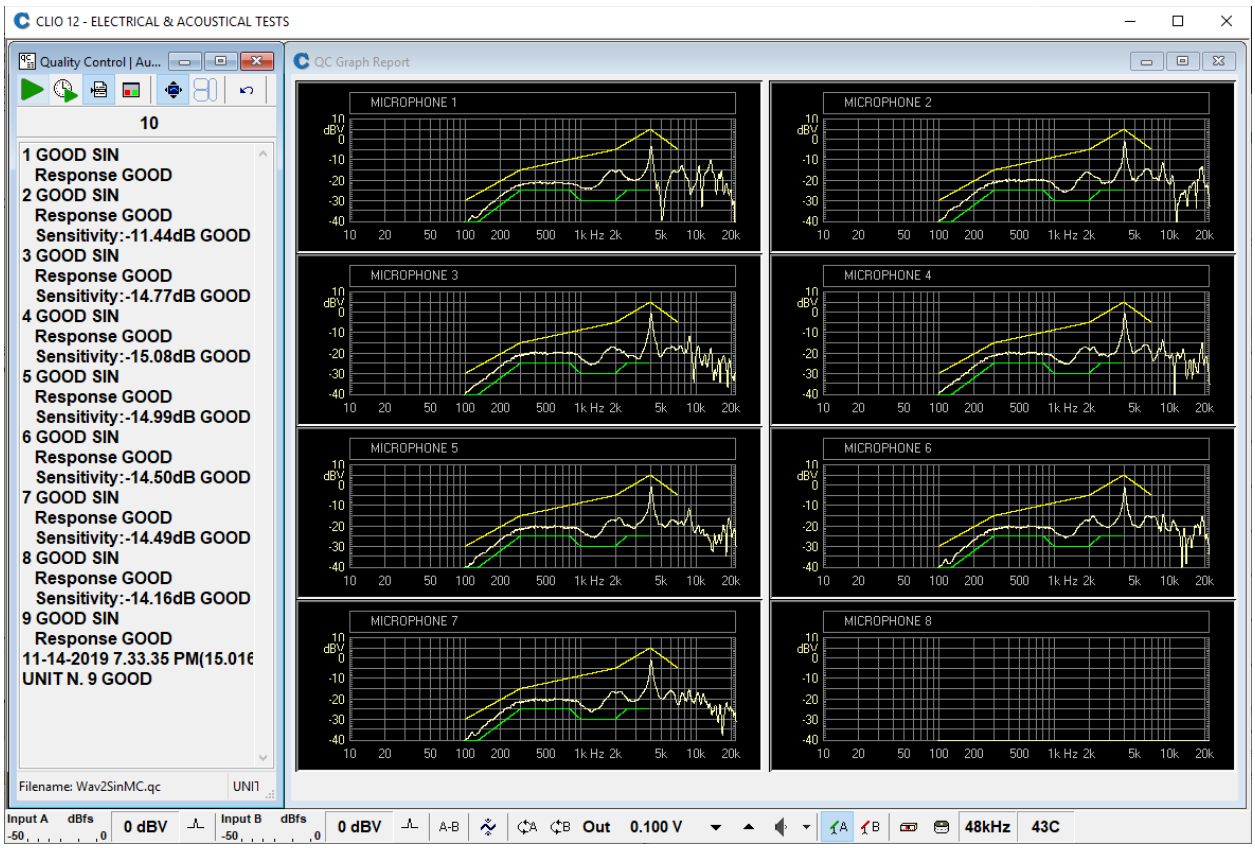
At this point the [WAV2SIN] sections will test in series the response of each microphone channel alongside specified limits:

```
[WAV2SIN]
GRAPH TITLE=MICROPHONE 1
WAVFILEINPUT=ch1.wav
REFERENCE=REF.SIN
LIMITS=ch1.LIM
…
```

Where the limit file has a sensitivity and an absolute frequency response mask:

```
[ABSOLUTE]
[SENSITIVITY]
UPPER=-10
LOWER=-20
[UPPER LIMIT DATA]
100     -30
300     -15
2000    -5
4000    5
7000    -5
[LOWER LIMIT DATA]
100     -45
300     -25
800     -25
1000    -30
2000    -30
2500    -25
4000    -25
```

At the end of the test you should get a complete report of the QC test of the 8 microphone channels of the remote device.



CONCLUSIONS

CLIO 12.5 QC new WAV2SIN functionality allows to test smart devices such as the multichannel microphone shown in this application note.

The scenario hereby presented shows the flexibility of the non real time and remote Sinusoidal testing and can be applied to many practical cases.