TESTING SMART DEVICES WITH CLIO 12.5 QC PART II

by Daniele Ponteggia – dp@audiomatica.com

INTRODUCTION

In the first part of this document, Application Note 017, we presented the test of a seven microphone smart device miniDSP UMA-8 USB mic array – V2.0. This device has also an output channel which is visible between the ALSA playback devices, but it is not actually connected to any port.

Here we will complete the picture by testing the UMA-8-SP USB mic array which features a digital audio amplifier connected to the playback device. This is basically the same hardware as the miniDSP UMA-8 USB mic array – V2.0 with added a digital audio power amplifier.

DEVICE UNDER TEST

The device under test is a UMA-8-SP USB mic array, in this Part II we will test its playback capabilities.

https://www.minidsp.com/products/usb-audio-interface/uma-8-sp-detail

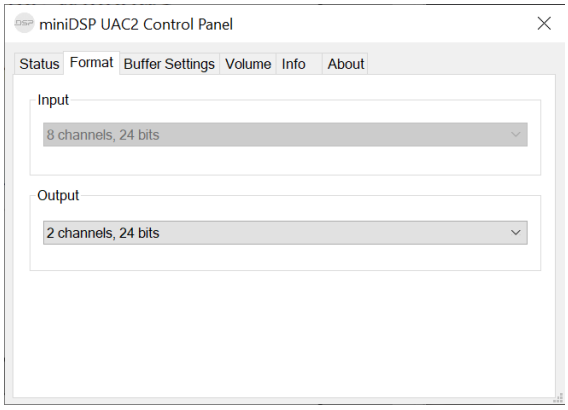
As illustrated in Part I, the device has a circular form factor but without case and has a 4 port Molex connector for digital power amplifier outputs.

DUT SETUP

We will use in this document the same setup we previously used, connecting the device to a RaspberryPi single board computer. Please read Part I for detailed instruction on how to set up the devices.

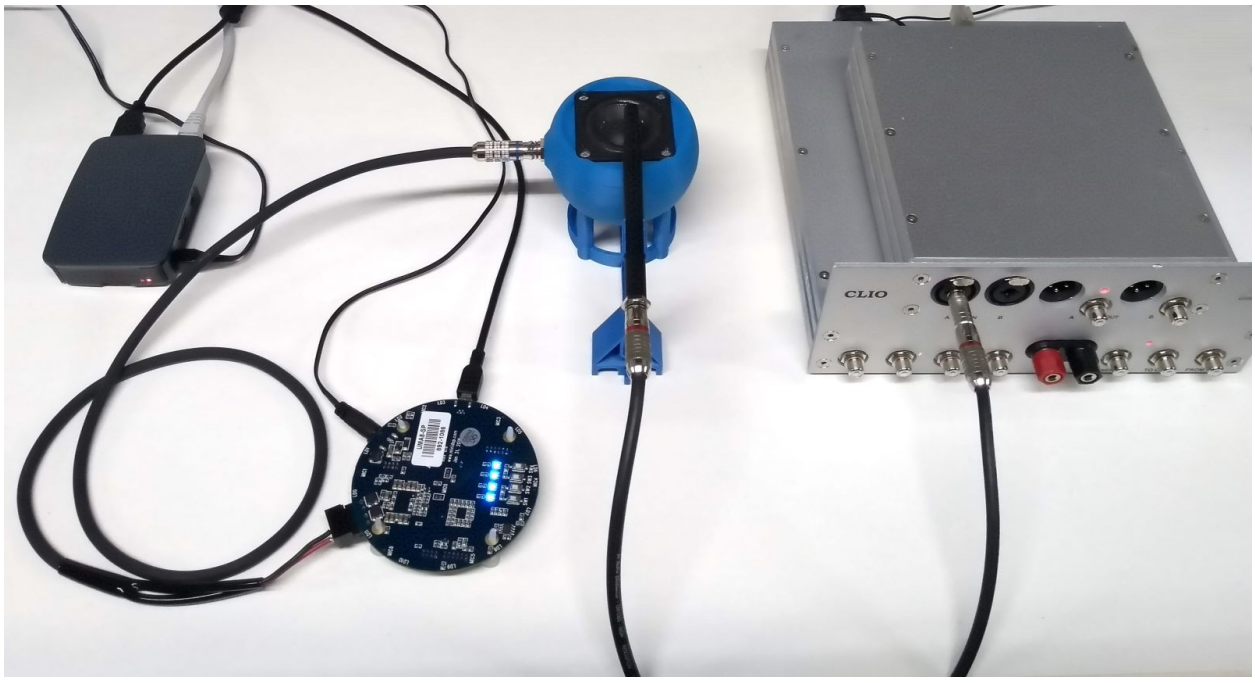
In Part I of this article we already tested the response of each individual microphone of the array, in order to achieve that we had to use a specific RAW firmware. In this application the RAW firmware is not required since output device is anyway seen as a two channel output.

In our example here the RAW firmware is present, upon connection to a Windows PC we should see the device as a 2 channels Output:



As illustrated in part I we are going to use a RaspberryPi single board computer running a light Linux distribution (Raspbian) to emulate a generic smart device.

The miniDSP UMA-8-SP device should be connected to one of the RaspberryPi free USB ports. We need also to connect a loudspeaker transducer to the miniDSP UMA-8-SPL digital power amplifier output. Without loss of generality here we will use a single loudspeaker connected to one of the two output channels of the miniDSP UMA-8-SPL.



The same speaker used in Part I can now be used as a transducer with an Audiomatica MIC-02 connected to CLIO fw-02 input and placed in the near field to collect the transducer's response.

Please follow ALSA setup instruction shown in Part I. Our midiDSP device output is “card 1”. We can access the RaspberryPi via ssh and control ALSA output volume using the command `amixer`. As a first step it is possible to check the available control channels:

```
pi@raspberrypi:~ $ amixer scontrols
Simple mixer control 'Mic',0
Simple mixer control 'Mic',1
Simple mixer control 'miniDSP VocalFusion Spk (UAC2.0)  Playback ',0
Simple mixer control 'miniDSP VocalFusion Spk (UAC2.0)  Playback ',1
Simple mixer control 'miniDSP VocalFusion Spk (UAC2.0)  Playback ',2
Simple mixer control 'miniDSP VocalFusion Spk (UAC2.0)  Playback ',3
```

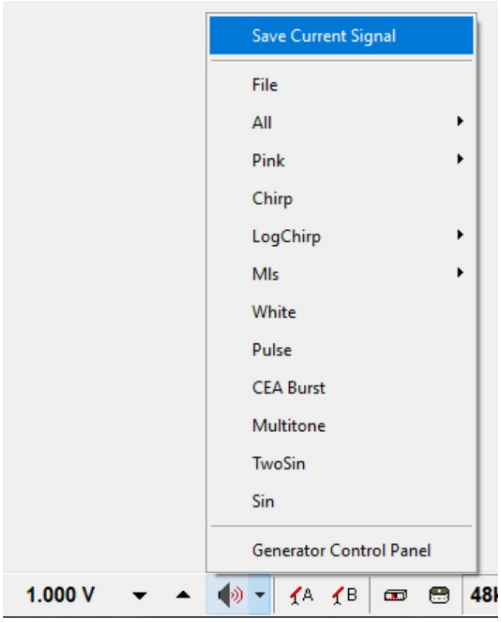
Our output channel connected to the audio digital power amplifier is the third playback device (Playback, 2). It is also possible to see the output channels mixer using the `alsamixer` command which opens a graphical interface.

From prompt we can read and set the output level using the commands `amixer sget` and `sset`:

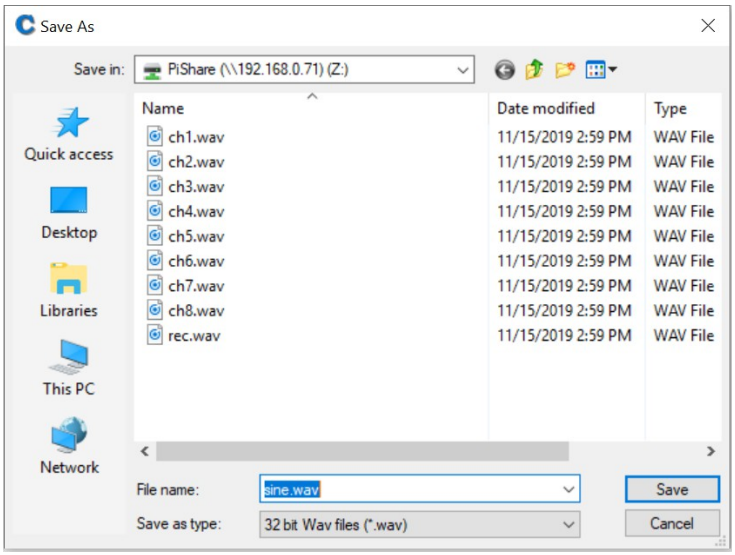
```
pi@raspberrypi:~ $ amixer sget 'miniDSP VocalFusion Spk (UAC2.0)  Playback ,
2'
Simple mixer control 'miniDSP VocalFusion Spk (UAC2.0)  Playback ',2
  Capabilities: volume
  Playback channels: Front Left - Front Right
  Capture channels: Front Left - Front Right
  Limits: 0 - 127
  Front Left: 127 [100%]
  Front Right: 127 [100%]

pi@raspberrypi:~ $ amixer sset 'miniDSP VocalFusion Spk (UAC2.0)  Playback ,
2' 95%
Simple mixer control 'miniDSP VocalFusion Spk (UAC2.0)  Playback ',2
  Capabilities: volume
  Playback channels: Front Left - Front Right
  Capture channels: Front Left - Front Right
  Limits: 0 - 127
  Front Left: 121 [95%]
  Front Right: 121 [95%]
```

We would like now reproduce a wave file using our RaspberryPi with miniDSP connected device. We can create a short sinusoidal wave audio clip using CLIO generator: generate a sinusoidal signal at default 1031.25 Hz frequency then select from the generator menu “Save Current Signal”.



A save file dialog will popup and we can save the sinusoidal stimulus on the shared resource on the RaspberryPi:



The sine.wav file is a short clip of a periodic sinusoidal wave with sampling rate 48 kHz and frequency 1031.25 Hz, which are the default values of the CLIO sinusoidal generator. The file is 16384 samples long, for a duration of about 341 ms.

The choice of this specific frequency and size is not causal, the 1031.25 Hz frequency is exactly a frequency bin for 48 kHz sampling rate and a 16384 samples FFT.

The file can be played from the RaspberryPi using the aplay command:

```
pi@raspberrypi:~ $ aplay Share/sine.wav
Playing WAVE 'Share/sine.wav' : Signed 32 bit Little Endian, Rate 48000 Hz,
Mono
```

A short beep should be heard from the loudspeaker connected to the audio power output of the miniDSP UMA-8-SP device. The output level can be adjusted using the amixer sset command.

In our case we found a proper value at about 83% of the maximum output level:

```
pi@raspberrypi:~ $ amixer sset 'miniDSP VocalFusion Spk (UAC2.0)  Playback ,
2' 88%
Simple mixer control 'miniDSP VocalFusion Spk (UAC2.0)  Playback ',2
  Capabilities: volume
  Playback channels: Front Left - Front Right
  Capture channels: Front Left - Front Right
  Limits: 0 - 127
  Front Left: 105 [83%]
  Front Right: 105 [83%]
```

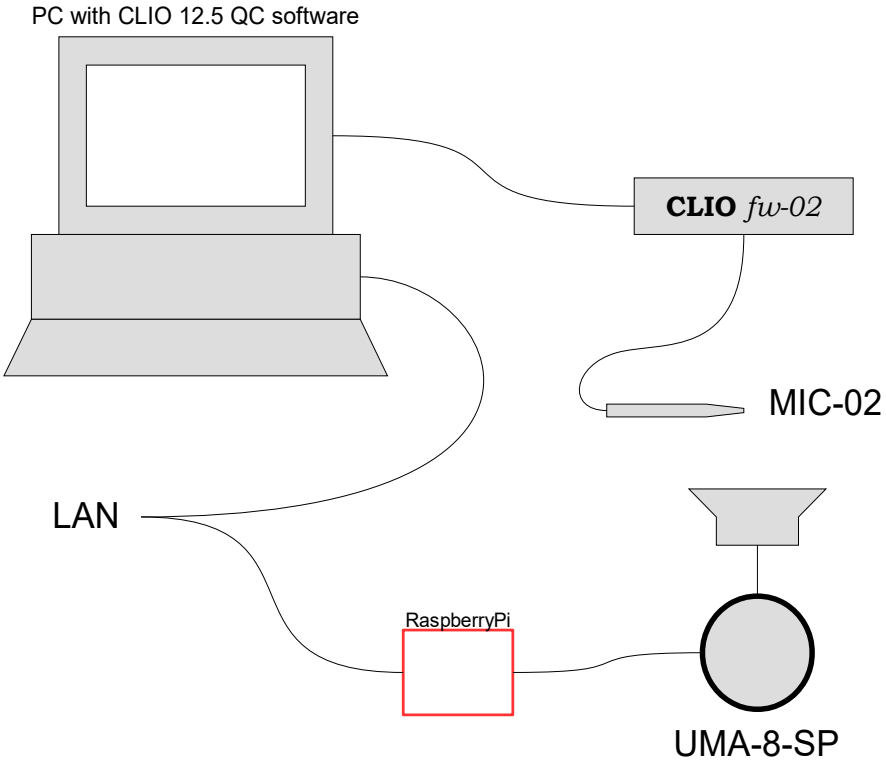
At this point we are able to send commands to our RaspberryPi to reproduce a wave file at a given level using the miniDSP UMA-8-SP device output.

MEASUREMENT SETUP

The test of the device output requires the following setup:

- PC with CLIO 12.5 QC software connected to LAN
- CLIO fw-02 connected via USB
- MIC-02 microphone connected to CLIO fw-02 input A
- RaspberryPi with Raspbian OS connected to LAN
- miniDSP UMA-8 (DUT) connected to RaspberryPi via USB
- Speaker connected to miniDSP UMA-8-SP output

This setup is schematically reported in the following figure:

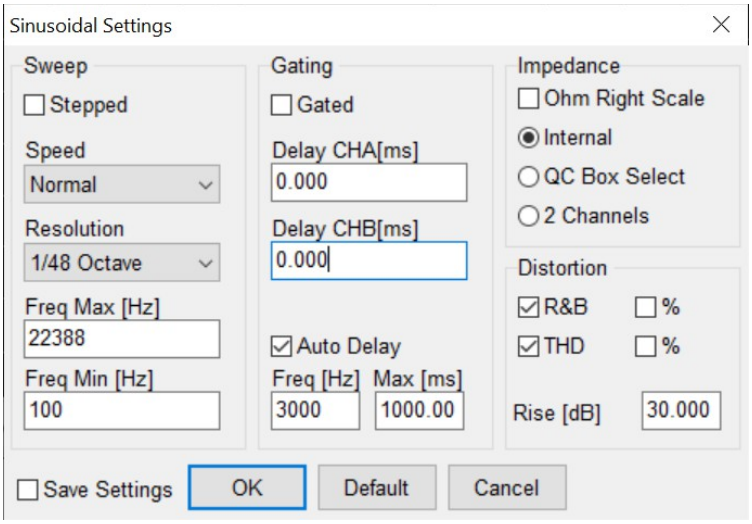


A suitable electro-acoustical transducer is required since the miniDSP UMA-8-SP has no loudspeaker. In this application we used a small 2-inches loudspeaker housed in an 3d printed box. In a typical application a smart device is usually equipped with one or more loudspeakers.

Please note that there are no connections to CLIO output. The stimulus will be transferred to the RaspberryPi as a file and reproduced by the miniDSP UMA-8-SP through the loudspeaker connected to the board's power amplifier output.

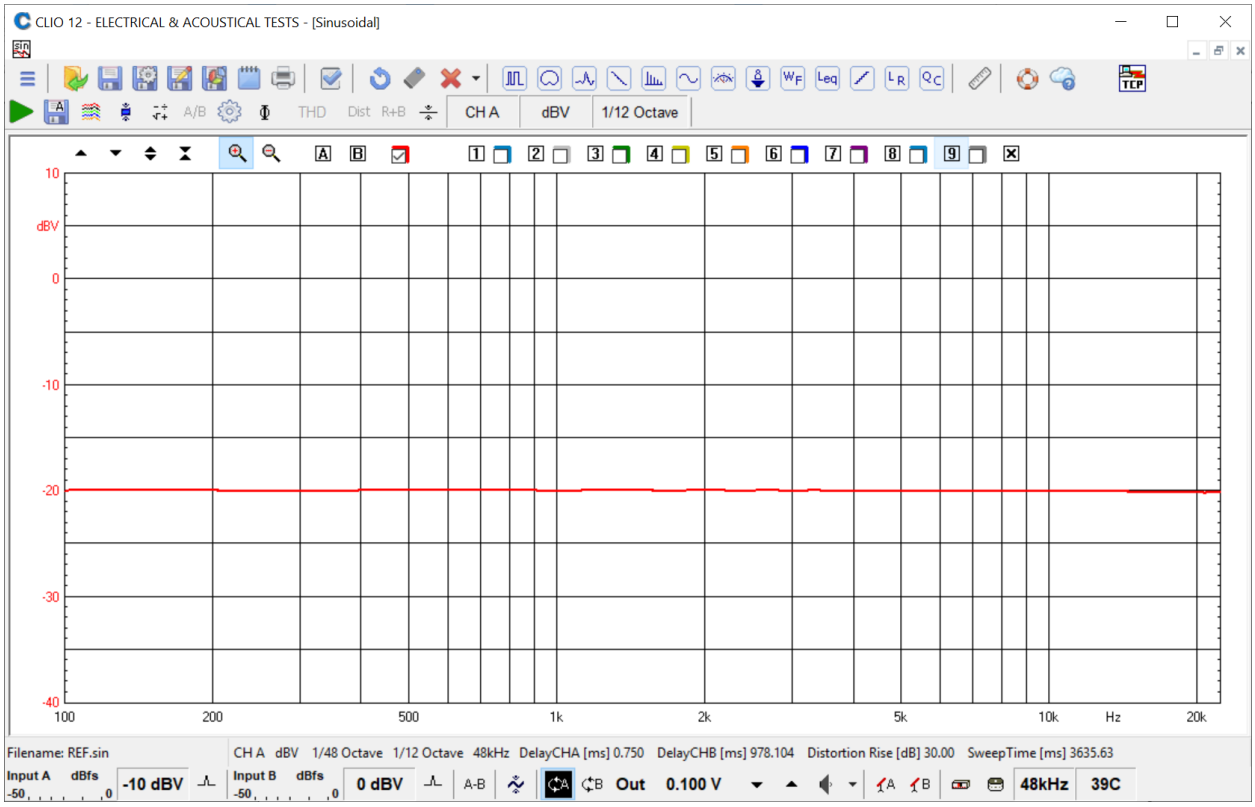
As a first step, like in Part I, a **reference file** which stores all the Sinusoidal measurement settings should be created. In order to create such Sinusoidal file it is advisable to set up the CLIO fw-02 channel A in loop.

In our example we are using settings similar to the one adopted for the measurement of the microphones in Part I:



Please not that here we also activated the THD and R&B Buzz readings, in this way it will be possible to include THD and R&B testing by editing the limit file.

Running the loop measurement we should get the following response:

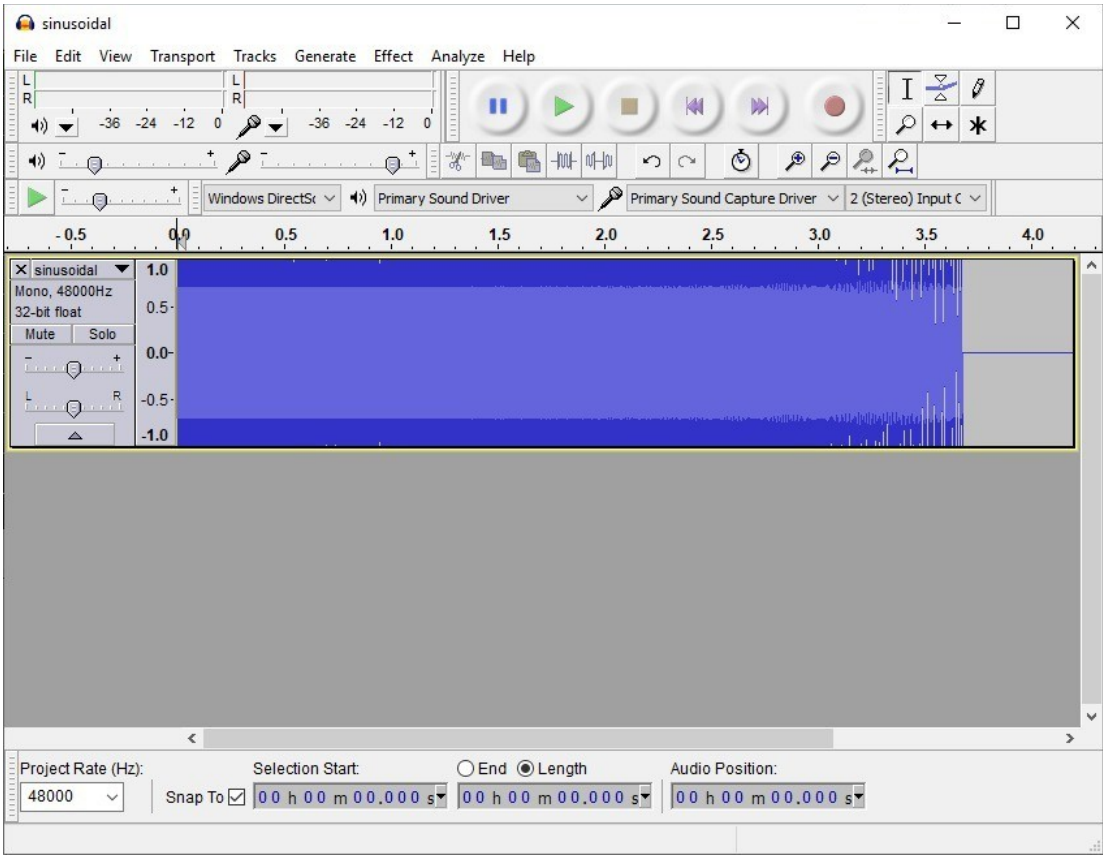


We will now save this file in a folder in the CLIO 12 host PC as a REF.sin and use for our Sinusoidal test in the QC script.

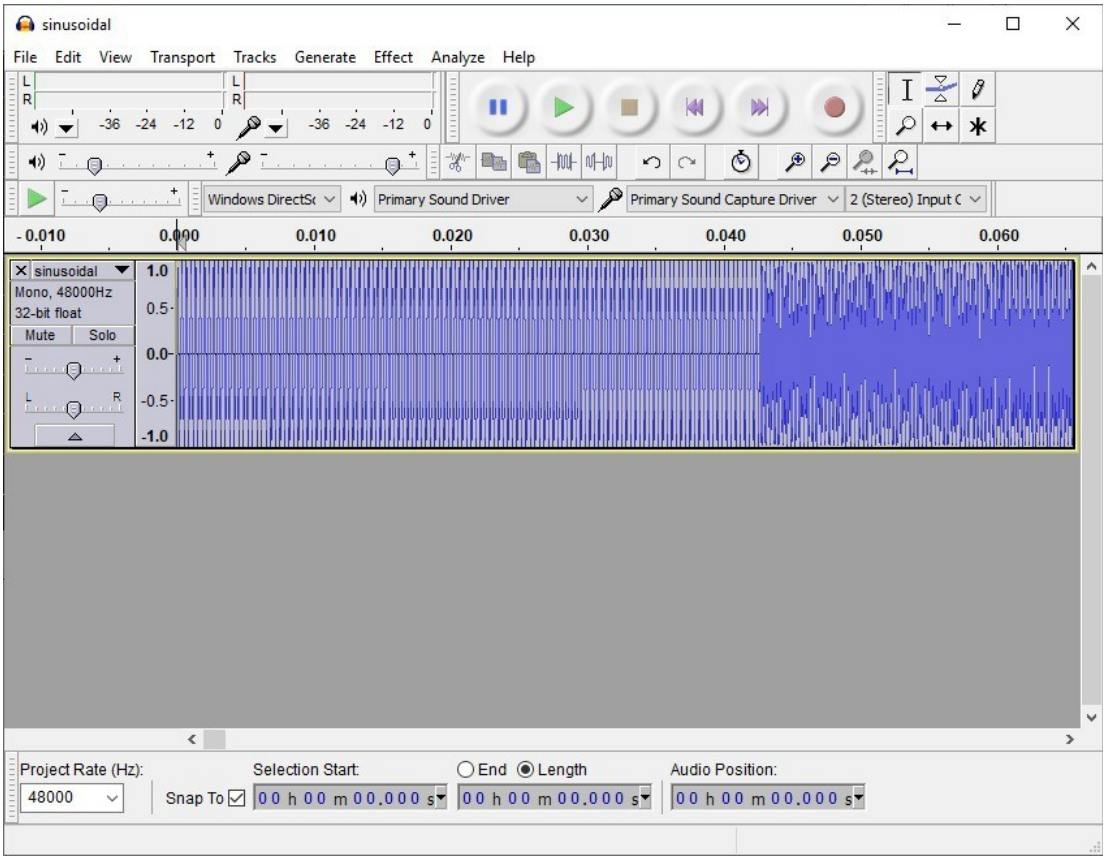
We need to save the sinusoidal stimulus by selecting the "Save Current Signal" item from the generator pop-up menu, the file can be saved[1] as "Sinusoidal.wav" in the RasperryPi shared resource.

Let's have a look at our stimulus signal, the wave file can be opened and inspected using an audio file editor such as Audacity:

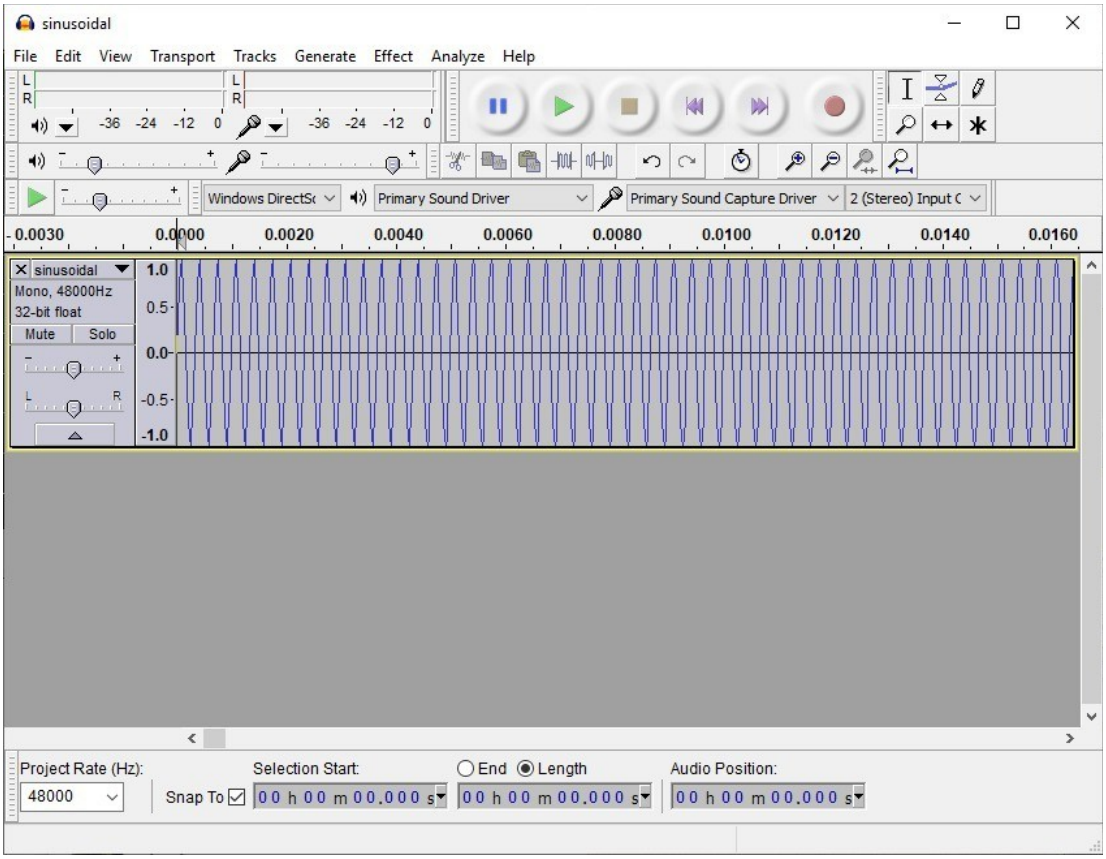
1 This is a feature of CLIO 12.5 generator, using "Save Current Signal" entry it is possible to save as a wave file the last signal played by the CLIO system, this can be also a signal generated at run time such as a Sinusoidal sweep.



It can be seen that the wave file is a 48 kHz sampling rate sinusoidal sweep. By zooming in the first part of the signal on the time axis the 3000 Hz pilot tone used by CLIO Sinusoidal tool to estimate the channel delay is clearly visible.



Zooming even further:



It has to be stressed that the saved wav file stimulus strictly reflects current sinusoidal settings and matches the reference REF.sin earlier defined which will then be used to analyze the result; if any of these settings change, for example the Auto Delay Frequency, or frequency extremes or resolution etc. etc., then the wav file has to be redone and delivered to DUT.

The sinusoidal stimulus can now be played back from the RaspberryPi:

```
pi@raspberrypi:~ $ aplay Share/sinusoidal.wav
Playing WAVE 'Share/sinusoidal.wav' : Signed 32 bit Little Endian, Rate 48000
Hz, Mono
```

The output level can be adjusted, if needed, using previously introduced `amixer sset` command. During level output check it is also convenient to monitor the input level and choose accordingly the right input sensitivity for CLIO fw-02 input A. This is the sensitivity we should then use in out QC script in the INA field.

We suppose here that the plink.exe utility is already installed, as described in Part I.

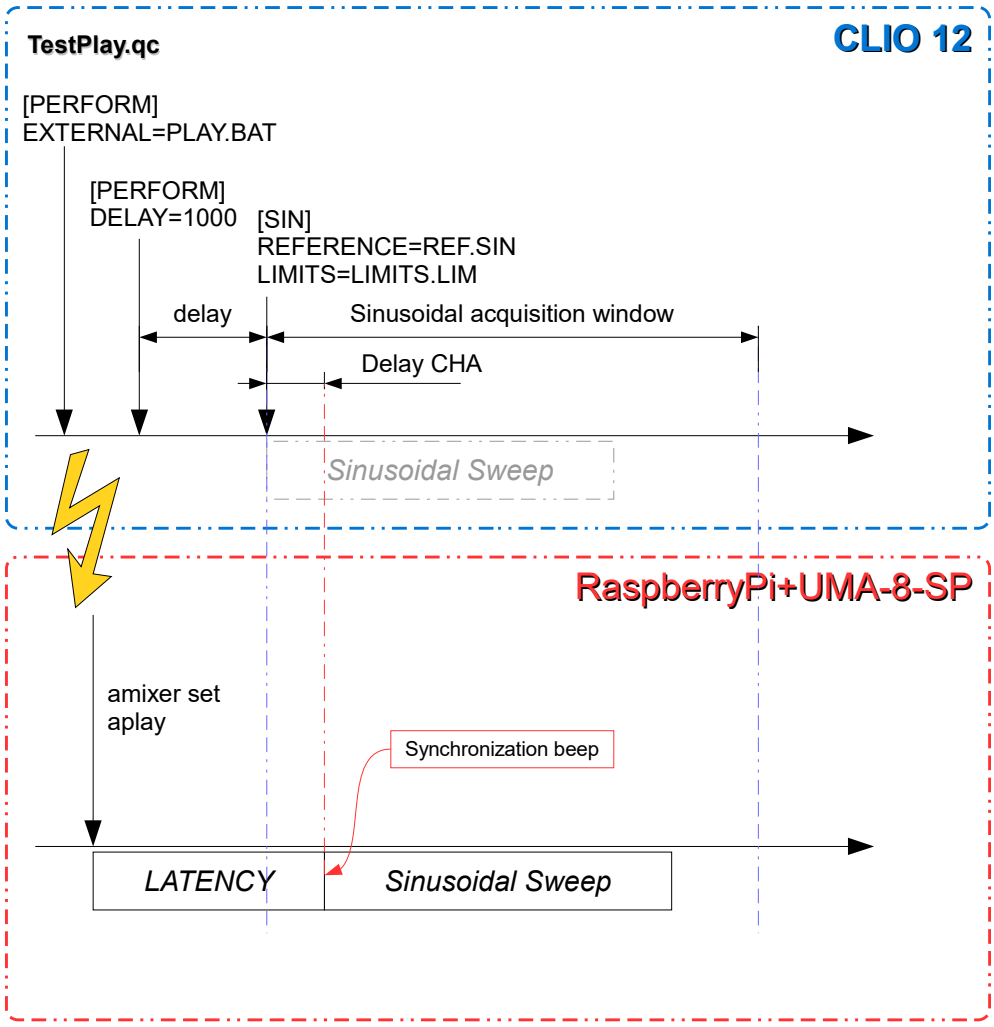
At this point remember to connect MIC-02 to CLIO fw-02 input A and check that phantom power supply is activated.

A possible QC Script to test the miniDSP UMA-8-SP output is as follows:

```
[SETPHANTOMA]
[PERFORM]
EXTERNAL=PLAY.BAT
DELAY=1000
[SIN]
OUT=1 V
INA=-10
INB=0
REFERENCE=REF.SIN
```

```
LIMITS=none
```

A scheme can help to understand the QC script timing, which is critical in this application:



The script sets the phantom power for CHA, where the microphone is connected, then calls a batch file:

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

PLAY.BAT is a simple batch file to set the output level and play back the Sinusoidal stimulus:

```
@echo off
plink pi@192.168.0.71 -pw raspberrypi amixer sset "'miniDSP VocalFusion Spk
(UAC2.0)  Playback ,2'" 105
plink pi@192.168.0.71 -pw raspberrypi aplay Share/sinusoidal.wav
```

The PLAY.BAT sets the output level of the miniDSP to 112 over maximum value of 127. Then sends the play command. This will start the playback of the previously created sinusoidal sweep .wav file[2].

After that there is a delay followed by the [SIN] test.

```
DELAY=1000
```

2 There are also smart devices with output gating features, in these cases it might be needed to edit the signal by adding a very low frequency tone prior the stimulus to activate the gate.

```
[SIN]
OUT=0.050 V
INA=-10
INB=0
REFERENCE=REF.SIN
LIMITS=none
```

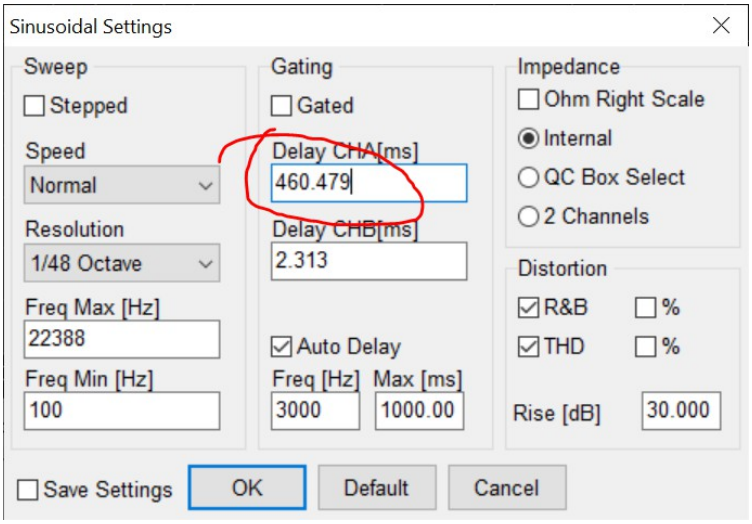
In this application the Sinusoidal test is carried out directly, without the need to resort to the WAV2SIN analysis as did in Part I. The trick here is that the stimulus is played by the RaspberryPi through our UMA-8-SP connected to the speaker. Please note that during the Sinusoidal test CLIO is also generating the sweep, but since there are no devices connected to CLIO outputs this signal is discarded.

In this situation the proper timing should be found depending the DUT latency and measurement setup. In CLIO Sinusoidal analysis the "delay CHA" and "delay CHB" fields are used to take into account the time-of-flight and/or latency of devices. This value can be set in the Sinusoidal measurement settings or CLIO can automatically find the value using a probe tone by selecting the "Auto Delay" checkbox.

There are limits though to the possible channel delay values, with a maximum of 1000 ms. In this application, as in similar cases, the latency of the RaspberryPi aplay is unknown, thus the auto-delay feature must be used. It is important to get this automatic process under control, and before creating a production line ready QC script, it is needed to check if the Sinusoidal test is properly carried out.

Once the above QC script has been run, there are two ways to check the synchronization and eventually change the value of `DELAY=1000` into QC script.

A first option is to inspect the value found (estimated) by the Auto Delay option to Delay CHA value inside Sinusoidal Settings:

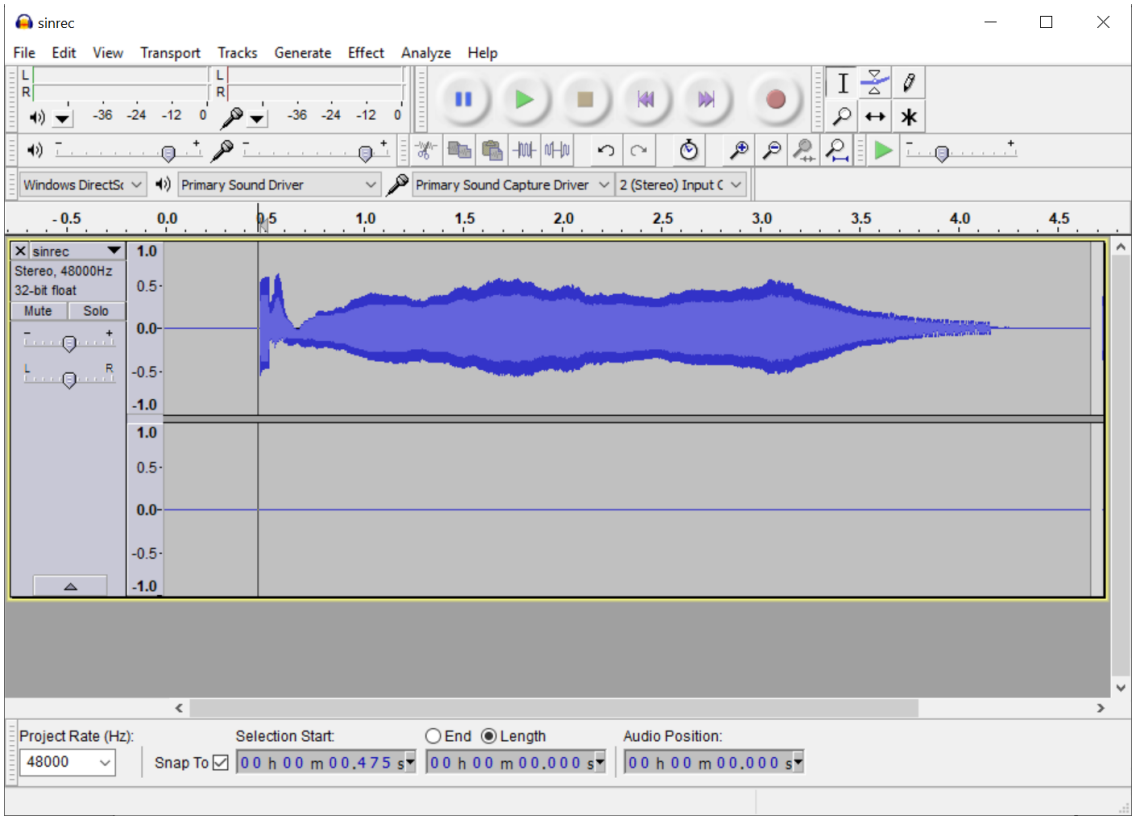


The Delay CHA value should be positive and less than 1000 ms. In this case the value is 460.479 ms which is in turn a good value because can handle latency variations of about ±450 ms.

Another option is to look at the waveform recorded by the Sinusoidal measurement menu, this is the waveform that is analyzed by the measurement menu and the one used to get the frequency response of the device. During QC operations both the stimuli and the recorded signal are saved in CLIO 12 TEMP folder (see the CLIO 12 User's Manual at 5.6 STARTUP OPTIONS AND GLOBAL SETTINGS) as temporary files with names: **sinplay.wav** and **sinrec.wav**.

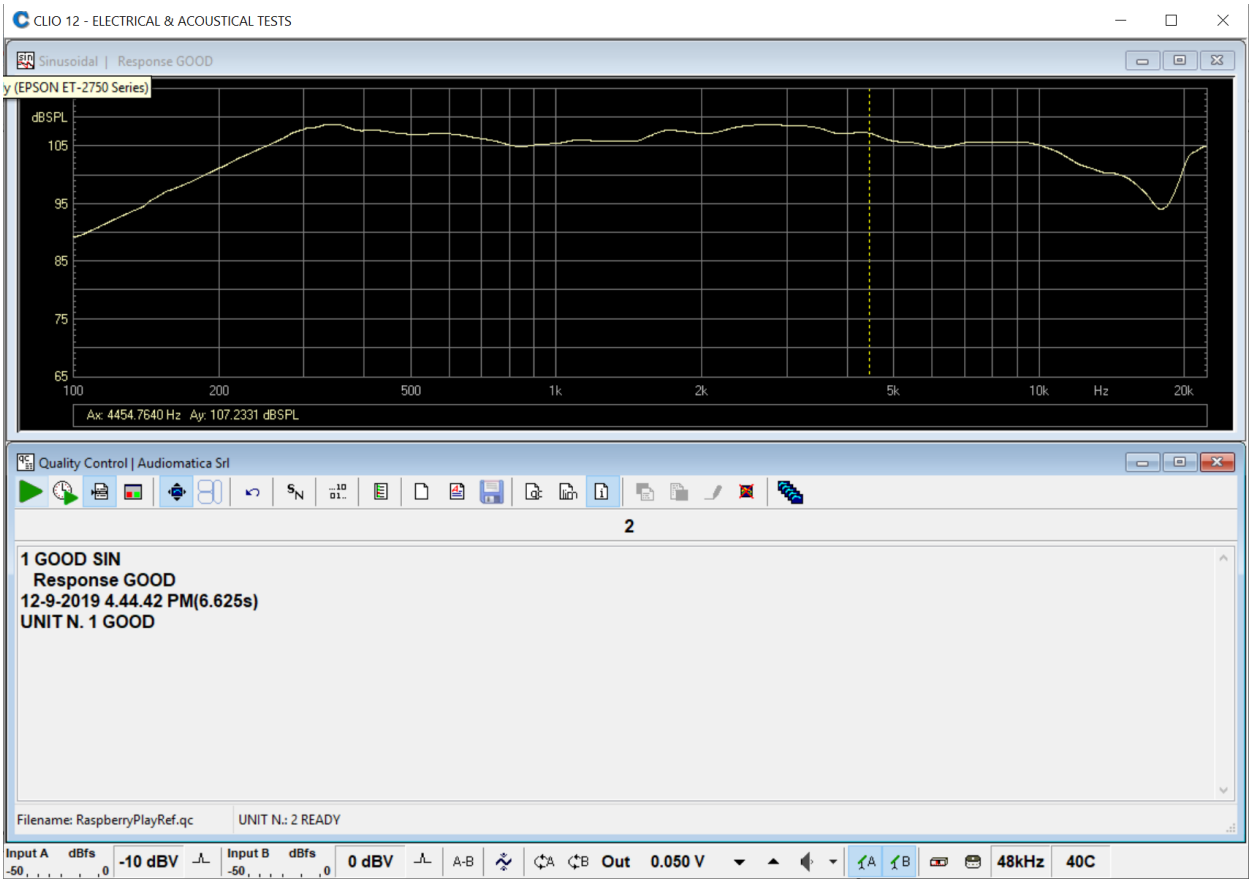
By opening the sinrec.wav file with a wave file editor such as Audacity it is possible to check the integrity of the acquired signal. This is the waveform of the acquisition

of the previous measurement:

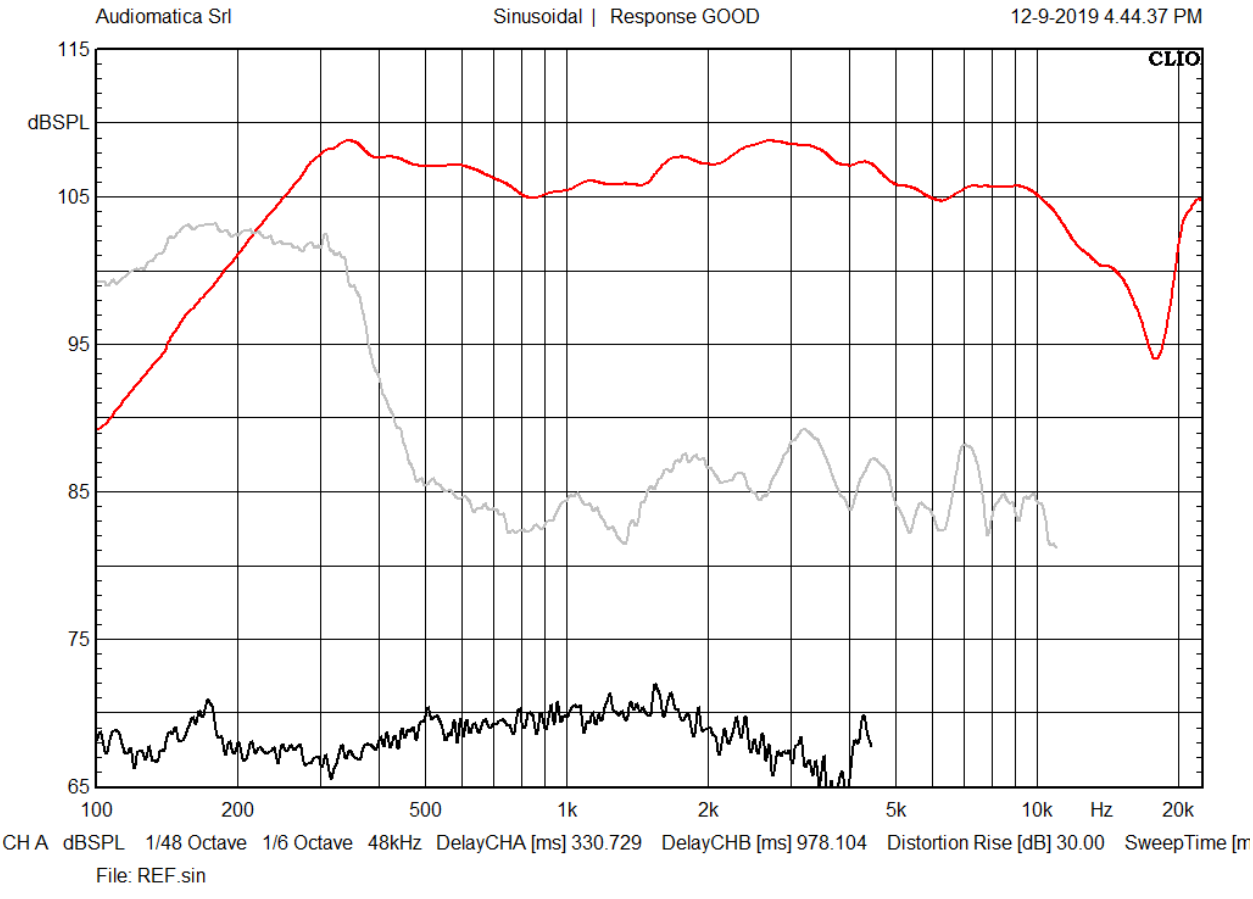


If the Delay CHA is outside the expected range and/or the acquired waveform is lacking head or tail, the `DELAY` setting in the QC script should be changed accordingly.

In this case we are satisfied with the `DELAY=1000` value, and after running the QC test we have the device frequency response.



Releasing the measurement it is possible to check the linear response, the distortion and rub'n'buzz curves:



The sinusoidal file can be now saved again as ref.sin. This file can be used as a reference file for the QC script, if the device is our reference sample relative limits can be used in the QC limit file.

COMPLETE QC TEST

Adding the following limit file to our QC script:

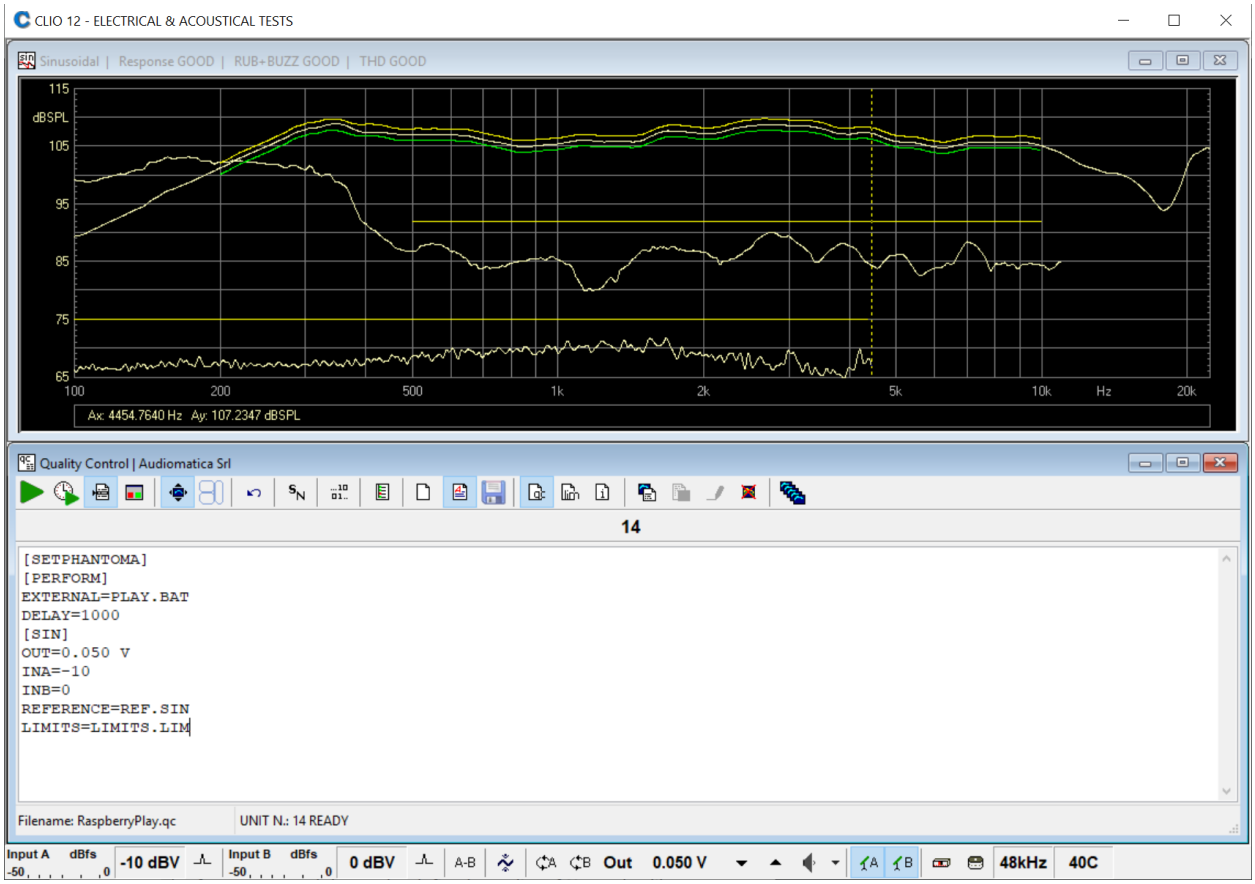
```
[RELATIVE]
[UPPER LIMIT DATA]
200    1
10000  1
[LOWER LIMIT DATA]
200    -1
10000  -1
[THD DISPLAY]
[THD UPPER LIMIT DATA]
500    62
10000  62
[RUB+BUZZ DISPLAY]
[RUB+BUZZ UPPER LIMIT DATA]
100    45
4400   45
```

The QC script will check against frequency response, distortion and rub'n'buzz limits.

This limit file should be referenced in the QC script as follows:

```
[SETPHANTOMA]
[PERFORM]
EXTERNAL=PLAY.BAT
DELAY=1000
[SIN]
OUT=0.050 V
INA=-10
INB=0
REFERENCE=REF.SIN
LIMITS=LIMITS.LIM
```

Launching the QC script we should finally find the following response:



CONCLUSIONS

In this series of application notes divided in Part I (microphones) and Part II (speakers) we presented the test of a smart device.

The techniques hereby used are very general and can be used to test devices which are not equipped with physical audio input and/or output line signal capabilities but can be accessed via ssh or Android Debug Bridge.

In our experience in order to successfully setup a quality control for these devices a complete knowledge of the device capabilities is mandatory: volume controls, automatic gain control, gating, compressors, etc...

In this two part document we have also shown a possible work path which starts from the simple playback or recording via remote control up to a complete QC test script using CLIO 12.5.