ADCTest User Guide
Version 0.1 build 110
2018-02-05

Contact information

AVP | https://www.WeAreAVP.com | info@WeAreAvp.com

GitHub repository

https://github.com/WeAreAVP/ADCTest

Download pages for application

Windows

https://www.weareavp.com/wp-content/uploads/2018/02/ADCTestSetup.zip

Note

Please help refine ADCTest further by reporting all bugs to
https://github.com/WeAreAVP/ADCTest/issues

Background

In 2016 AVP created and submitted to the Library a performance testing guideline and a user guide for low cost testing of analog to digital converters (ADCs) used in the digitization of audio recordings for preservation. These documents and the work behind them were part of a larger effort and set of deliverables over the past few years focused on the performance testing of ADCs. The earlier work in this larger effort focused on a high level testing guideline and user guide, offering a stringent set of performance specifications and a series of sophisticated test methods requiring a high performing, highly calibrated and complex test system. The testing devices on the market which meet the requirements of the high level guideline are upwards of $25,000. It is the case that these devices have dropped significantly in value since the original work was performed, and can be found on the market today for upwards of $15,000, but even this more reasonable number is well beyond the means of many archives. The concern at the conclusion of the high level test guideline effort was that a majority of organizations would simply not be able to utilize it, failing to address a central aim of this effort, which was to provide a simple, low cost means of testing their ADCs.

With this realization, the Library was interested in exploring lower cost test options that may be used by organizations with less resources. It was clear from the start that the endeavor to lower the cost would entail a sacrifice in both sophistication and precision. In many ways, this presented challenges to another foundational goal, which was to be able to test ADCs used in the preservation of audio recordings to ensure that the ADC performance was sufficient for the task at hand and that the ADC was operating without failure. Considering these trade offs the following considerations became central in the conversation:

- Many organizations are currently not testing their devices at all. This was an impetus for the project.
- Testing, even if elementary, is better than not testing at all. A major finding of our field tests, performed as part of the larger effort over the years, was that even the most basic testing revealed significant errors in the ADC or signal path that had not been caught previously due to lack of performing any testing at all.
- The more frequently that testing is performed, the better the chance of identifying issues.
- The complexity and learning curve involved with being able to properly operate and interpret the results of sophisticated test devices is a significant deterrent for many organizations. Thus, one of the original aims was to bring simplicity to this process.
- The historic lack of standards, best practices, or guidelines providing both the test methods and the performance metrics in one succinct document has created ambiguity on the need and protocols for testing.
- The significant cost of test devices has been a major deterrent for organizations. In a survey done as part of this work, an overwhelming majority of organizations reported that they would not spend more than $1,500 for a test device. In the author’s experience

working with organizations over the past 17 years, many organizations are not even able to pay $1,500 for their ADC much less a test device.

The high level test guideline and associated deliverables, including scripts for running the tests, solve most of the problems listed above with the one major exception of cost which impacts a large number of organizations. While the guideline may be used in specifications for outsourcing digitization without impact on internal capital expenditures, for organizations that have internal digitization operations and wish to use a test system to apply the guideline, it is likely that only a small percentage that have not performed testing in the past would be empowered to do so in the future with the publishing of the high level guideline and associated deliverables due to cost.

On the flip side, the low cost test guideline and associated deliverables solve most of the problems with a major caveat that the quantity of tests in the test suite and the quality of the results fall significantly short of the high level test guideline. Speaking generally, a low cost device will simply not be as sophisticated or as precise as a higher cost device. However, the low cost test enables a large percentage of organizations that have internal digitization operations and wish to use a test system to apply the guideline to do so, enabling many to begin routine testing for the first time. As mentioned above, the transition from not-testing-at-all to testing routinely, even if using less than ideal means is probably the biggest gain to be had. At minimum this can answer the question of whether or not the ADC/system is failing.

The observations that arose and the thinking that went into this led us to realize that there are really three primary questions that an organization aims to answer through use of a testing guideline:

1. Is my ADC/system failing?
2. How does my calibrated, well performing ADC performs relative to the guideline and other ADCs?
3. Is my ADC/system performing optimally relative to its own specifications?

A high level test device that meets the guideline can answer all of these questions. A low cost test device can likely answer question number one for any converter, but it can only answer questions number two and three for ADCs which have performance specifications that are significantly lower than the device itself. This leads to the belief that:

1. A low cost guideline is a worthwhile endeavor with meaningful and practical utility.
2. Perhaps those without sufficient budget for performing the high level test guideline might temporarily procure a test device or utilize a service provider at key points (e.g. upon purchasing an ADC, performing quarterly testing), while frequently using the low cost test guideline and device on an ongoing and routine basis.

Introduction

In 2016, the test guideline and a user guide were drafted and submitted to the Library. Following this work, the Library contracted with AVP to develop a beta-version software application that would perform the test methods and report the results relative to the performance metrics in an automated or semi-automated way. AVP first attempted to work with the developers of existing test and measurement software applications, such as ARTA and RMAA, to work on a FADGI-specific application. These attempts were unsuccessful, leaving us with needing to start from scratch. The first thing we did was to draft a list of requirements. At a high level these requirements and features consisted of the following:

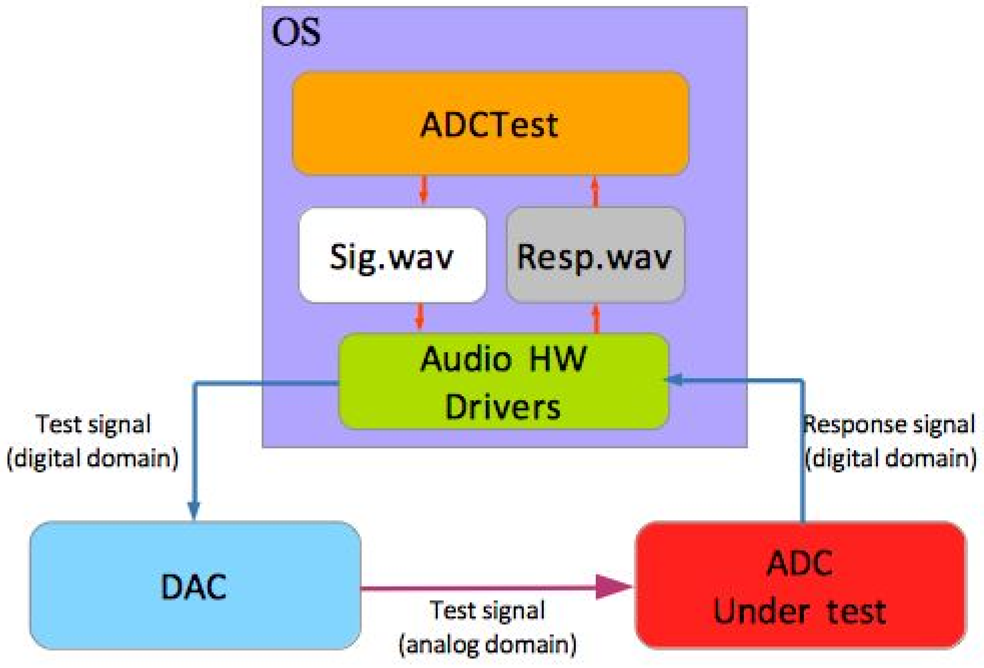
1. Perform all of the tests in the low cost guideline
2. Report results of tests according to the low cost guideline methods and metrics
3. Provide simple pass-fail reporting as well as more detailed results
4. Decouple the signal generation from the signal analysis to enable non-real time analysis and the use of alternative signal generators
5. Provide a simple calibration mechanism
6. Perform the tests using as much automation as possible
7. Provide ability to enable and disable individual tests
8. Support up to 24 bit, 96 kHz signals
9. Support USB, SPDIF, Firewire and AES interfaces

Following the drafting of these requirements AVP worked with application developer Christian Landone to develop an application referred to as ADCTest using C++ as the coding language.

Use-Cases and Application Architecture

With the base requirements in mind we set out to design the application architecture and approach and came up with the name ADCTest. To begin, we established the basic test setup and protocol envisioned for use of ADCTest.

The first envisioned test setup and protocol considers a scenario in which an operator is using ADCTest for signal generation and analysis to test their own ADC. This test type is referred to as “normal” and uses the following setup and protocol:


OS
ADCTest
Sig.wav
Resp.wav
Audio HW Drivers
Test signal (digital domain)
Response signal (digital domain)
DAC
Test signal (analog domain)
ADC Under test

1. The DAC is connected to one of the audio interfaces on the computer running ADCTest.

2. The ADC under test is connected to one of the audio interfaces on the computer running ADCTest.

3. ADCTest is started and upon opening ADCTest shows the list of tests in the guideline. The parameters for each test procedure are pre-defined, as specified in the guideline.

4. The operator selects both the DAC and ADC from a list of devices.

5. The operator performs calibration to establish levels to be used as a point of reference by ADCTest when performing testing.

6. The operator reviews the list of test procedures to be performed and has the option to enable or disable each test.

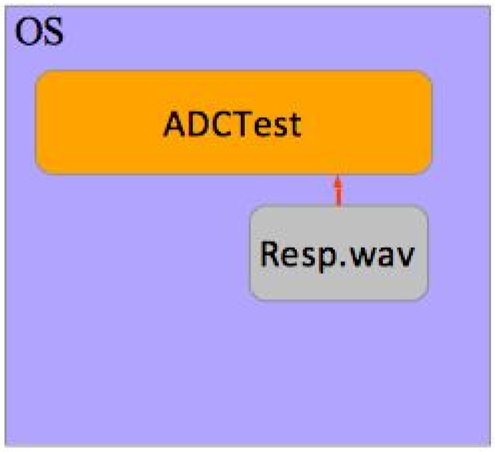
7. The operator initiates the test procedure with a simple button press.

8. The application automatically performs each test procedure in the list, pausing and prompting the operator for input at points where human intervention is required.

9. Once the end of the test procedures list is reached, results are saved to file both in machine and human-readable formats.

10. The operator is able to review simple pass-fail results as well as see details for each test result.

The second envisioned test setup and protocol considers a scenario where a client is analyzing and reporting on a set of response files received from a digitization vendor that has produced them using the guideline and is submitting them to the client for proof of ADC conformance. This test type is referred to as “offline” and uses the following setup and protocol:



1. The vendor captures response files that have been passed through the ADC being used for client work. The test signals used and the response files captured conform to the FADGI low cost guideline and ADCTest parameters.

2. The vendor delivers the response files to the client.

3. The client opens ADCTest and loads or points ADCTest to the response files. The computer that the client is running does not need to have any specialized audio hardware or software.

4. The client initiates the analysis procedure with a simple button press.

5. Once the end of the analysis procedure is reached, the client is able to review simple pass-fail results as well as see details for each test result in order to audit whether the vendor’s ADC conforms to the FADGI guideline performance metrics, and if interested, the extent to which it passed and/or failed.

6. Results are saved to file both in machine and human-readable formats.

Following the establishment of these scenarios we explored the application architecture and concluded the following:

1. The architecture should permit the introduction of additional classes that extend the type of waveforms that can be produced by the application’s internal signal generation engine, e.g. random noise (white/pink), chirps.

2. The architecture should permit the introduction of additional classes that extend the type of tests can be performed by the application's analysis engine, e.g. Impulse response of the system.

3. The architecture should use a fully modular approach, treating each individual test as its own module and allowing the source type, test signal characteristics, and measurement algorithm parameters to be configured for each newly created module.

4. The architecture should decouple the signal generation and signal analysis routines so that testing can be performed in real-time, non-real-time and use test signals from other generators.
    a. ADCTest will generate the test signals and save them as WAVE files (signal files).

    b. ADCTest will play the signal files through the DAC and ADC and save the resulting WAVE files (response files)

    c. ADCTest will load the response files, analyze them, report the results and save the analysis results as XML files.

Tests and Test Parameters

In order to keep the user interface simple, yet offer flexibility and expandability, we decided to utilize an XML schema to identify and document all of the parameters and associated settings for test signal generation and analysis. The default configuration will conform to the FADGI low cost guideline, but users may further setup alternate configurations as they wish by altering the XML file and saving it.

The main node, named "FADGIProject" contains basic parameters that may be used by all tests in the procedures, in particular, the property *"datafolder"* specifies the path that will be used by the application to write the necessary audio files and test results.

This can be manually edited by the user and, if left empty, the path will be automatically set on Windows machines as the public documents folder (CSIDL_COMMON_DOCUMENTS).

The child node *"procedures"* contains the description and operational parameters of each test.

The following properties are defined for each *"test"* node:

1) id
The test identification number; This must be incremental.

2) name
The test short-hand name.

3) alias
A human-readable brief description of the test procedure.

4) enable
Specifies whether the specific test procedure is enabled. If set to false, the test will not be run. This parameter can be changed using the application user interface or within the XML.

The *"parameters"* child node enumerates individual operational parameters for each test. A number of parameters are common to all tests, in particular:

1. testtype
   Specifies whether the testing will be performed using ADCTest signal generation (normal) or using pre-recorded response files (offline).

2. signal
   Specifies the type of stimulus signal used by the test procedure. Currently the signal types are: "octsine", or octave stepped sine for the frequency response tests; "dualsine", or dual sine for the intermodulation distortion tests, and "singlesine", or single sine for all other tests.

3. analyser
   Specifies which analysis module will be used for the test. Currently the analyser types are: "stepfreq" for the frequency response tests; "thdn" for the THD + noise and dynamic range tests; "lfimd" for both of the frequency intermodulation distortion tests; "spis" for spurious inharmonic tests, and "xtalk" for cross talk tests.

4. chidx
   Specifies the index of the channel under test.

5. detectionlevel
   Specifies the level threshold at which a signal is determined to be valid to begin analysis.

6. inttime
   Specifies the minimum length in mS of the stimulus signal.

7. transtime
   Specifies a "guarding period" in mS that will be added at the beginning and end of the stimulus signal. This period will be discarded by the analysis module in order to avoid the effect of sudden transients.

8. bursttime
   If the test procedure requires multiple burst signals, this parameter specifies the pause period in mS between bursts.

9. outputfreqresponse
   Whenever the specific test generates a frequency plot during the performance metrics evaluation, if this parameter is enabled, the plot will be saved in the results file.

10. workfolder

Defines the location where the files associated with a test should be saved. As a default this parameter inherits the value for “datafolder” parameter in the main node.

11. signalfile
    Specifies the name of the audio file containing the stimulus signal.

12. responsefile
    Specifies the name of the audio file containing the audio stream recorded from the ADC under test.

13. resultsfile
    Specifies the name of the XML file containing calculated performance and pass/fail result.

    One final parameter that is hard coded into the application for all tests is the window type that it is used. The only window specified within the FADGI guideline currently is Kaiser 7. Therefore Kaiser 7 is hard coded into all tests. The application has the ability to turn this into a parameter, allowing users to select alternate windows, but Kaiser 7 utilizes a good trade-off between main and side lobes, making it a good candidate for general use in test and measurement. It is also worth noting that level deviations caused by windowing are compensated for, so the level of a harmonic component is accurate, regardless of the type of window used.

The following parameters are specific to only certain tests as identified:

14. freqstart [frequency response]
    The frequency at which the frequency sweep will begin.

15. freqstop [frequency response]
    The frequency at which the frequency sweep will end.

16. octsteps [frequency response]
    The number of steps per octave used in the frequency sweep.

17. level [frequency response]
    The level at which the tones in the frequency sweep should be at the output of the ADC.

18. tonefreq [THD+N, spurious inharmonic, cross-talk]
    The frequency of the stimulus signal.

19. tonelevel [THD+N, intermodulation distortion, spurious inharmonic, cross-talk]
    The level at which the stimulus signal should be at the output of the ADC.

20. fftlength [THD+N, intermodulation distortion, spurious inharmonic, cross-talk]
    The size, or number of points used, for FFT based analysis.

21. fftnoavg [THD+N, intermodulation distortion, spurious inharmonic, cross-talk]
    The number of averages to use for the FFT in performing analysis.

22. fftavgtype [THD+N, intermodulation distortion, spurious inharmonic, cross-talk]
    Specifies the type of averaging used when performing analysis. For instance, linear or exponential.

23. notchbw [THD+N, intermodulation distortion, spurious inharmonic]
    Specifies the bandwidth of the notch used for the removal of a stimulus frequency in the analysis of a response file.

24. harmsearchbw [THD+N]
    For each nth order harmonic, a search is made in the response spectrum for the highest value in the range: (n*Fc – harmsearchbw : n*Fc + harmsearchbw). In practice, due to the limited FFT resolution, the harmonic might fall in an adjacent frequency bin, using a search range guarantees that the harmonic is found.

25. lowerlimit [THD+N, intermodulation distortion, spurious inharmonic]
    The lowest frequency at which a signal will be analyzed.

26. tonefreq1 [intermodulation distortion]
    The frequency of the first tone in a dual tone stimulus signal.

27. tonefreq2 [intermodulation distortion]
    The frequency of a second tone in a dual tone stimulus signal.

28. levelratio [intermodulation distortion]
    The level ratio between the first and second tone in a dual tone stimulus signal.

29. imdtype [intermodulation distortion]
    Specifies the intermodulation distortion analysis type. For instance, SMPTE or CCIT.

30. higherlimit [spurious inharmonic]
    The highest frequency at which a signal will be analyzed.

The “performancespecs” node lists which performance metric will be used by the application to return a pass or fail result at the end of the specific test. The parameters are as follows:

1. name
   The name given to the performance metric.

2. type
   Variable type (either double or string), currently this specifier is not used by the application.

3. value
   The pass-fail value for the performance metric.

4. units

The unit of measure being used for the performance metric. For instance dB or dBFS.

5. criterion
   The relationship of the result compared to the pass-fail value. For instance, lessthan.

Configuration File

ADCTest saves a configuration file in the datafolder location, which stores configuration settings that are saved as part of the calibration procedure. The parameters stored within the configuration file are as follows:

1. NewPrefsInitialized
   Version control managed internally by the application.

2. Version
   Version control managed internally by the application.

3. DataDumpDir
   Identifies the directory being used to place the files.

4. AudioSRate
   Identifies the sample rate being used.

5. FrameSize
   Identifies the frame size being used for audio processing.

6. AudioHostName
   Identifies the audio drivers being utilized.

7. InputDevName
   Identifies the ADC being used.

8. InputDevChans
   Identifies the number of channels on the ADC.

9. OutputDevName
   Identifies the DAC being used.

10. OutputDevChans
    Identifies the number of channels on the DAC.

11. InBufferLength

Specifies the size of the input (ADC) audio buffers, managed from the audio device setup panel.

12. InBufferThreshold
    Specifies the point at which the buffer gets filled again if it falls below the stated threshold. This is managed from the audio device setup panel.

13. OutBufferLength
    Specifies the size of the output (DAC) audio buffers, managed from the audio device setup panel.

14. OutBufferThreshold
    Specifies the point at which the buffer gets filled again if it falls below the stated threshold. This is managed from the audio device setup panel.

15. OutputStreamGain
    Identifies the level set within ADCTest as a reference point for all signal levels.

16. RTALength
    Length, or resolution, of the FFT plot in the calibration panel, managed from the audio devices setup panel.

17. RTAWindow
    Window type for the real time FFT in the calibration panel. This is no longer in use. The window is currently fixed to kaiser7.

18. RTAExAvg
    Specifies the settings for the real time FFT averaging in the calibration panel, managed from the audio devices setup panel.

19. EnumerateDevicesToFile
    No longer in use.

20. EnumerateDevicesToFilePath
    No longer in use.

21. LogToFile
    No longer in use.

22. LogToFilePath
    No longer in use.

Test Results File

When a test is performed there are three files generated per test. These are the signal audio file, the response audio file, and the result XML file. The result XML file has the following structure.

FADGIResults is the parent node and has the title of the result file as well as the channel index of the audio channel being reported on. FADGIResults contains dataset, performancespecs, and testoutcome.

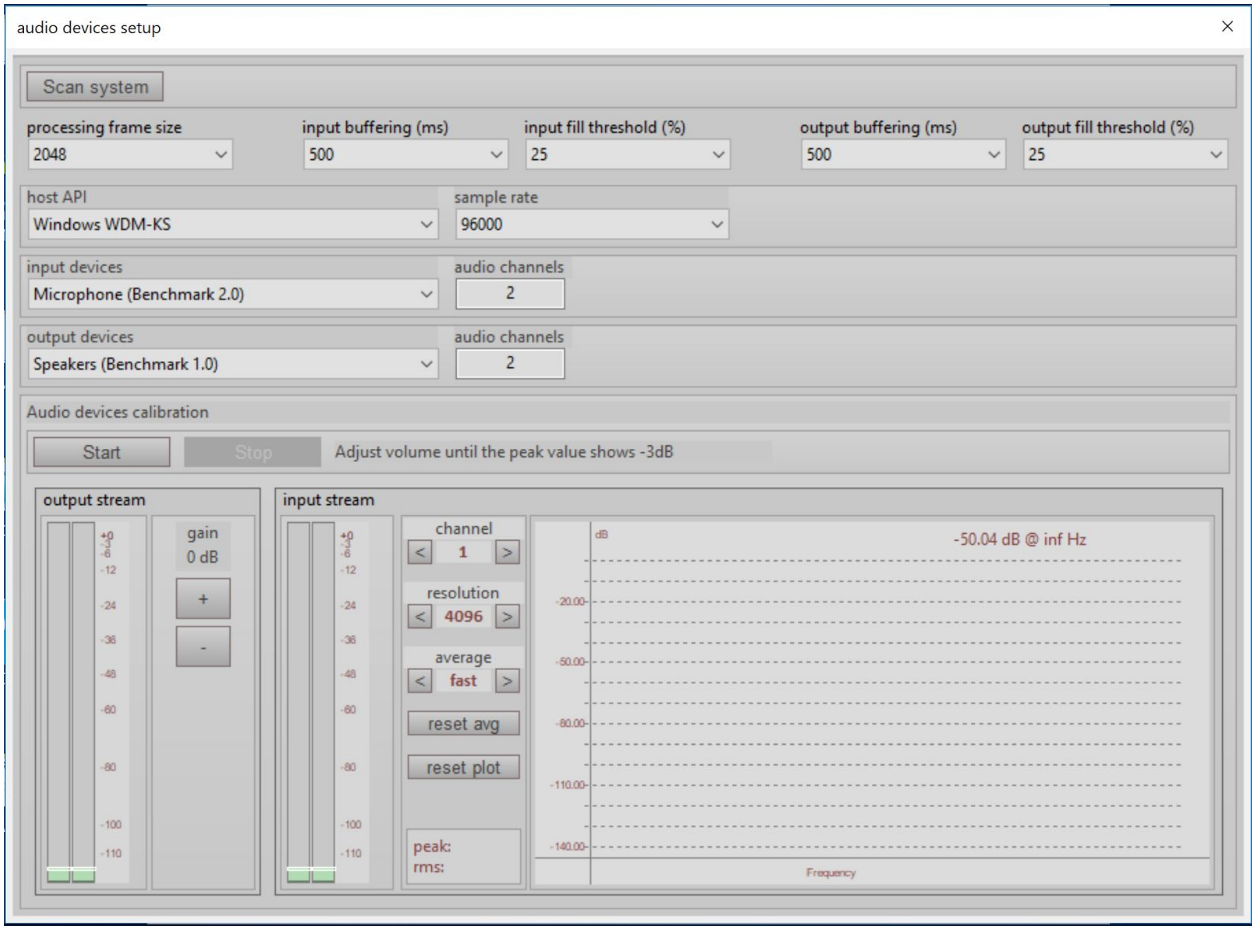
dataset has an ID and contains testmetrics and freqresponse. testmetrics contains a variable list of parameters. These parameters contain reporting on results for different relevant metrics for each test, providing the name of the metric, the result for that metric, and the unit being reported. Within the application these are the metrics which are reported within the results user interface on the left hand side of the graph. freqresponse contains the frequency and associated resulting level for each frequency measured.

performancespecs contains the specification name, data type, value, unit of measure, and criterion serving as the reference for pass/fail reporting.

testoutcome contains either a pass or fail value, reporting the outcome of the results compared to the performance specification.

Performing Calibration

To perform calibration, select “Devices” from the “Settings” menu within the application.


audio devices setup

Scan system

processing frame size: 2048
input buffering (ms): 500
input fill threshold (%): 25
output buffering (ms): 500
output fill threshold (%): 25

host API: Windows WDM-KS
sample rate: 96000

input devices: Microphone (Benchmark 2.0)
audio channels: 2

output devices: Speakers (Benchmark 1.0)
audio channels: 2

Audio devices calibration

Start | Stop | Adjust volume until the peak value shows -3dB

output stream: gain 0 dB

input stream: channel 1, resolution 4096, average fast, reset avg, reset plot, peak:, rms:

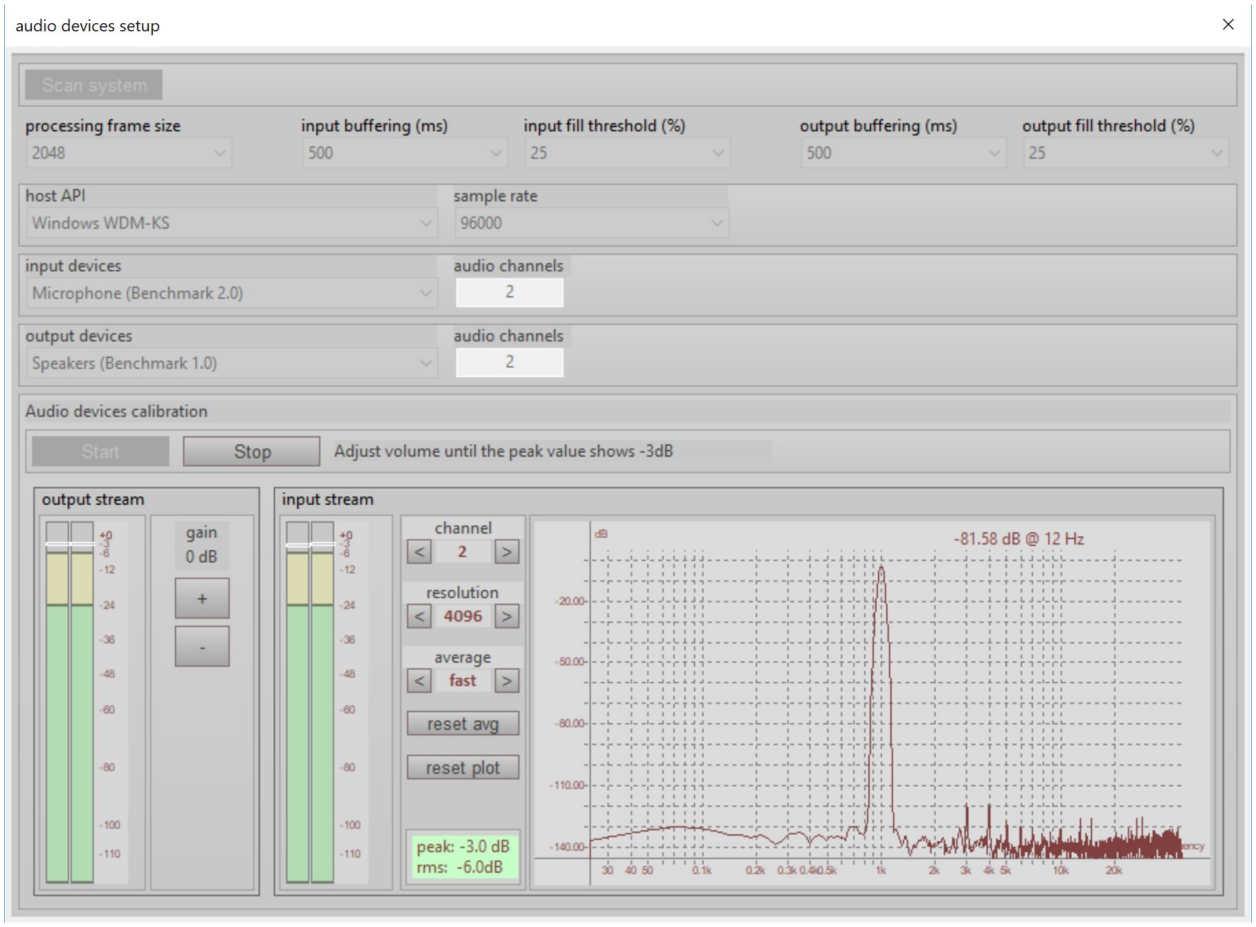
-50.04 dB @ inf Hz

Select the audio drivers you would like to use for testing using the dropdown menu under "host API". Note that different audio drivers have different performance levels and this choice may impact your results.

Once the audio drivers are selected, the user may select the ADC under "input devices", DAC under "output devices", and sample rate.

The top row of information containing processing frame size, input buffering, input fill threshold, output buffering and output fill threshold should be left on the default settings unless there are audio driver and performance issues.

After the audio drivers, ADC, DAC and sample rate are selected, select the "Start" button under "Audio devices calibration". This will send a 1 kHz signal out of the output of the DAC and to the input of the ADC. Activity should be seen at the "output stream" level meters, "input stream" level meters, and on the FFT graph.


audio devices setup

processing frame size: 2048
input buffering (ms): 500
input fill threshold (%): 25
output buffering (ms): 500
output fill threshold (%): 25

host API: Windows WDM-KS
sample rate: 96000

input devices: Microphone (Benchmark 2.0)
audio channels: 2

output devices: Speakers (Benchmark 1.0)
audio channels: 2

Audio devices calibration

Adjust volume until the peak value shows -3dB

-81.58 dB @ 12 Hz

peak: -3.0 dB
rms: -6.0dB

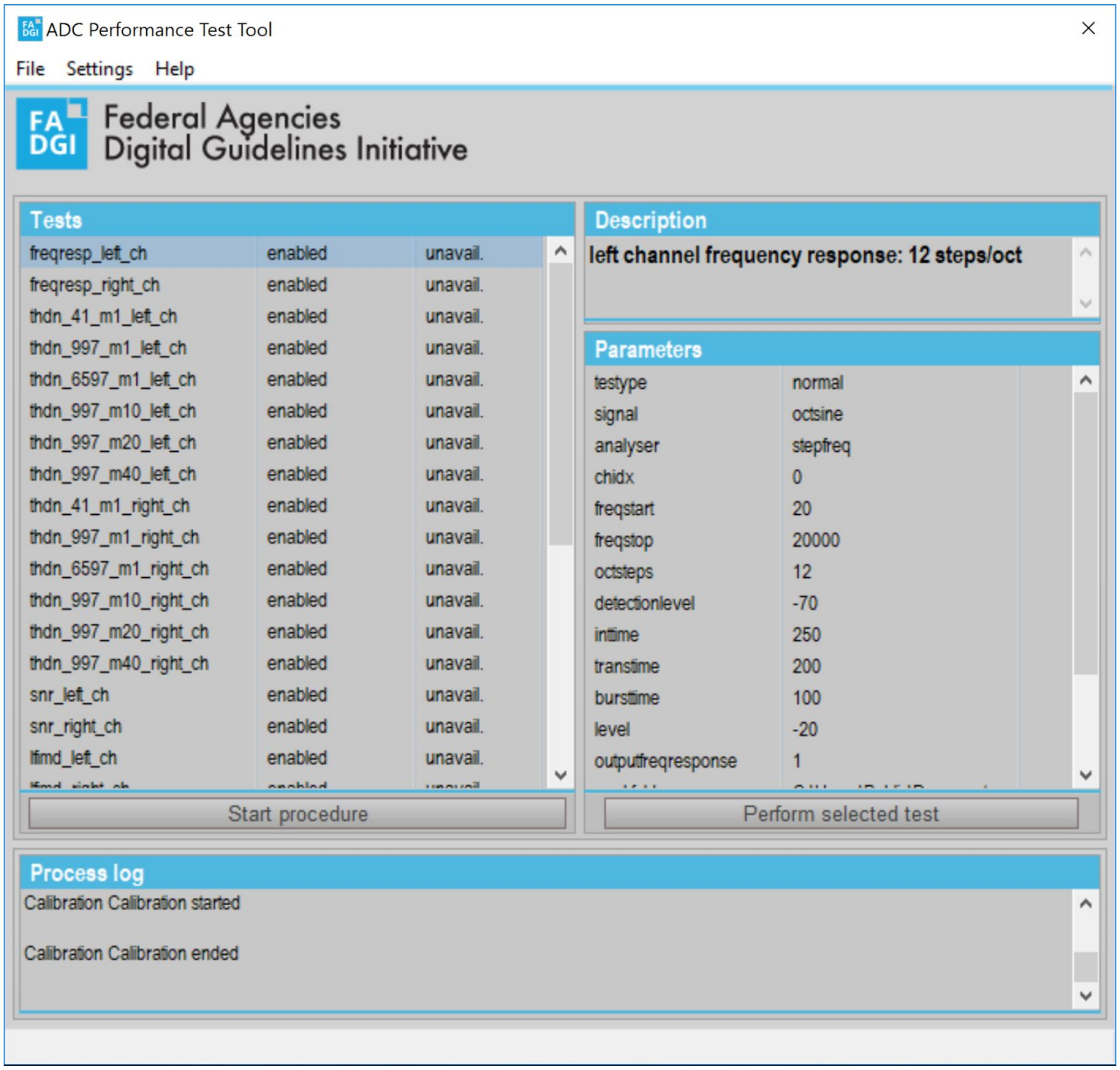
The user must adjust each channel to -3dB using some combination of the level meters on the DAC, ADC and gain within ADCTest. The graph and reporting of peak levels default to channel 1. When levels are adjusted so that channel 1 is set to somewhere between -2.9dB and -3.1dB the peak level reporting will turn green. After channel 1 is set, adjust the other channels in the same manner. Be sure not to overload the output of the DAC or the input of the ADC when performing calibration. The meters and FFT graph should provide visual feedback to help with performing calibration successfully. The resolution and average parameters in the input stream pane are simply for controlling the FFT display and do not impact the calibration.

Once all channels are set at -3dB select “Stop” under “Audio devices calibration” and close the audio devices setup window using the “x” in the top right corner of the window. At this point all calibration settings are saved.

Performing a Full Test Suite (Normal Test Type)

A test suite is a series of individual tests performed in sequence. The default test suite that is loaded upon launching the application is the FADGI test suite. The settings for this file are stored in default.xml which is located next to the executable file for ADCTest. Note that to alter this file it must be stored in a location where the user has permissions to write data.

Clicking on each test will bring up the associated info in the Description and Parameters panes to the right of the Tests pane.



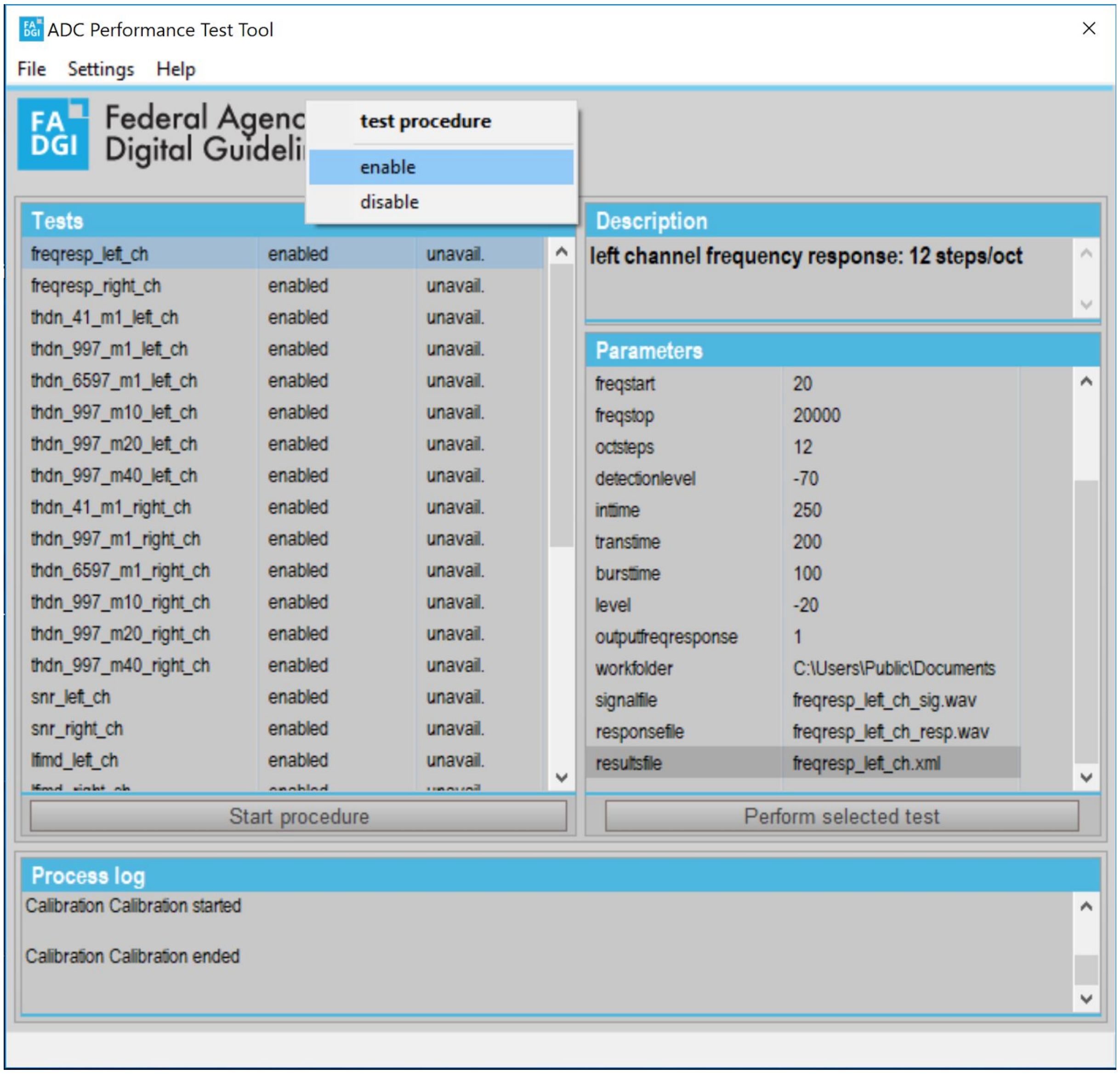
freqresp_left_ch	enabled	unavail.
freqresp_right_ch	enabled	unavail.
thdn_41_m1_left_ch	enabled	unavail.
thdn_997_m1_left_ch	enabled	unavail.
thdn_6597_m1_left_ch	enabled	unavail.
thdn_997_m10_left_ch	enabled	unavail.
thdn_997_m20_left_ch	enabled	unavail.
thdn_997_m40_left_ch	enabled	unavail.
thdn_41_m1_right_ch	enabled	unavail.
thdn_997_m1_right_ch	enabled	unavail.
thdn_6597_m1_right_ch	enabled	unavail.
thdn_997_m10_right_ch	enabled	unavail.
thdn_997_m20_right_ch	enabled	unavail.
thdn_997_m40_right_ch	enabled	unavail.
snr_left_ch	enabled	unavail.
snr_right_ch	enabled	unavail.
lfimd_left_ch	enabled	unavail.

left channel frequency response: 12 steps/oct

testype	normal
signal	octsine
analyser	stepfreq
chidx	0
freqstart	20
freqstop	20000
octsteps	12
detectionlevel	-70
inttime	250
transtime	200
burstime	100
level	-20
outputfreqresponse	1

```
Calibration Calibration started

Calibration Calibration ended
```

Within the ADCTest user interface the user may enable or disable tests by right/control clicking on the middle column within the Tests window and selecting enable or disable.



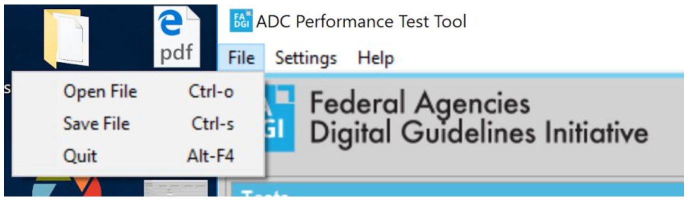
Tests		
freqresp_left_ch	enabled	unavail.
freqresp_right_ch	enabled	unavail.
thdn_41_m1_left_ch	enabled	unavail.
thdn_997_m1_left_ch	enabled	unavail.
thdn_6597_m1_left_ch	enabled	unavail.
thdn_997_m10_left_ch	enabled	unavail.
thdn_997_m20_left_ch	enabled	unavail.
thdn_997_m40_left_ch	enabled	unavail.
thdn_41_m1_right_ch	enabled	unavail.
thdn_997_m1_right_ch	enabled	unavail.
thdn_6597_m1_right_ch	enabled	unavail.
thdn_997_m10_right_ch	enabled	unavail.
thdn_997_m20_right_ch	enabled	unavail.
thdn_997_m40_right_ch	enabled	unavail.
snr_left_ch	enabled	unavail.
snr_right_ch	enabled	unavail.
ifmd_left_ch	enabled	unavail.

Description: left channel frequency response: 12 steps/oct

Parameters	
freqstart	20
freqstop	20000
octsteps	12
detectionlevel	-70
intime	250
transtime	200
bursttime	100
level	-20
outputfreqresponse	1
workfolder	C:\Users\Public\Documents
signalfile	freqresp_left_ch_sig.wav
responsefile	freqresp_left_ch_resp.wav
resultsfile	freqresp_left_ch.xml

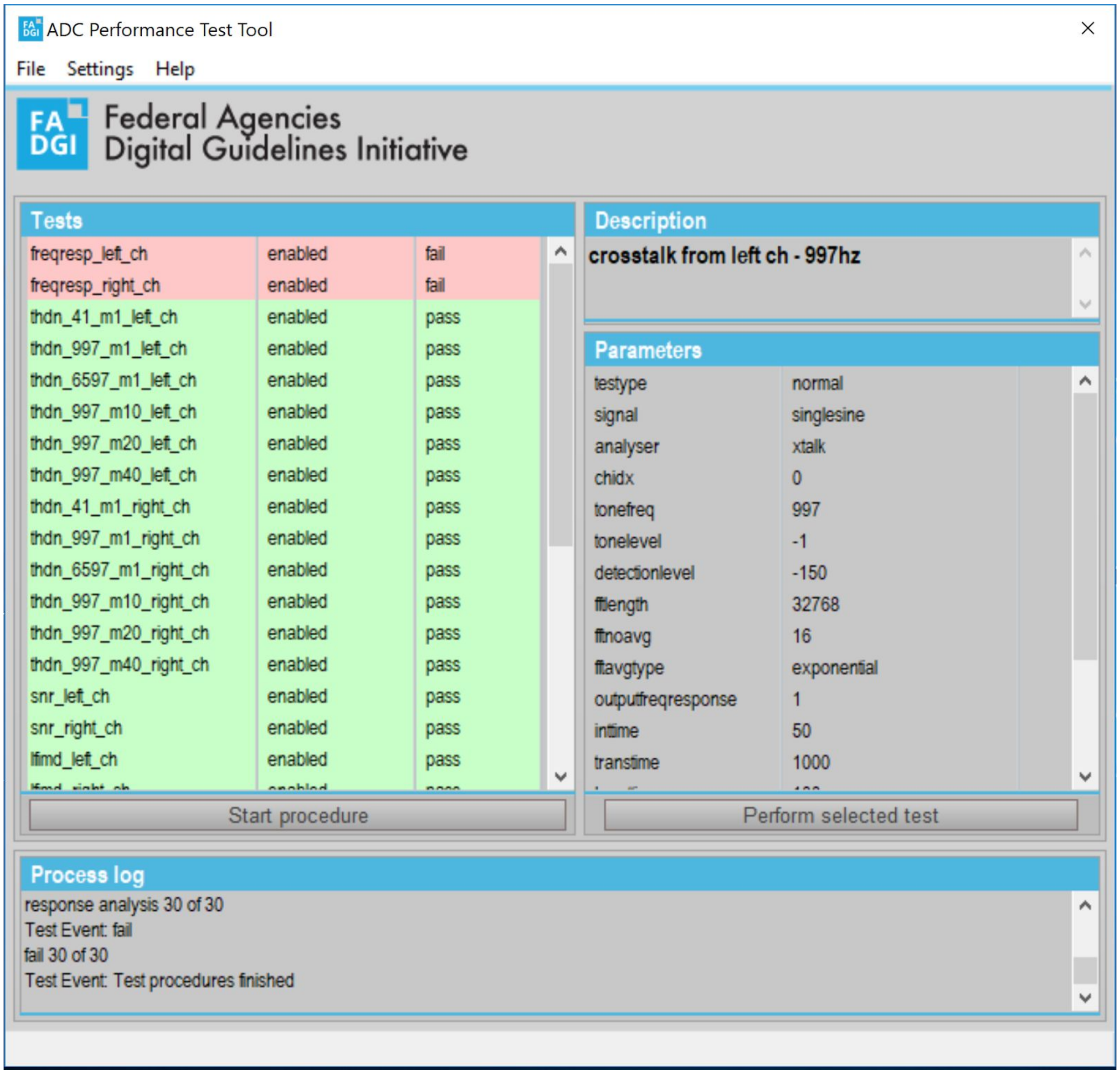
```
Calibration Calibration started

Calibration Calibration ended
```

If users want to create their own test suites they may do so by creating a test suite XML file and open it from the file menu to replace the FADGI test suite represented by default.xml.



To begin the test suite select Start Procedure. All enabled tests in the suite will be performed in sequence from top to bottom in an automated fashion. The Process log pane will report on progress of the testing.

Once the test is complete the user is prompted with a success message. Tests that have passed will be green and the right column within the Tests pane will be labeled "pass". Tests that have failed will be red and will be labeled "fail".


ADC Performance Test Tool
File
Settings
Help
Federal Agencies Digital Guidelines Initiative
Start procedure
Perform selected test

Tests

freqresp_left_ch	enabled	fail
freqresp_right_ch	enabled	fail
thdn_41_m1_left_ch	enabled	pass
thdn_997_m1_left_ch	enabled	pass
thdn_6597_m1_left_ch	enabled	pass
thdn_997_m10_left_ch	enabled	pass
thdn_997_m20_left_ch	enabled	pass
thdn_997_m40_left_ch	enabled	pass
thdn_41_m1_right_ch	enabled	pass
thdn_997_m1_right_ch	enabled	pass
thdn_6597_m1_right_ch	enabled	pass
thdn_997_m10_right_ch	enabled	pass
thdn_997_m20_right_ch	enabled	pass
thdn_997_m40_right_ch	enabled	pass
snr_left_ch	enabled	pass
snr_right_ch	enabled	pass
Ifimd_left_ch	enabled	pass

Description

crosstalk from left ch - 997hz

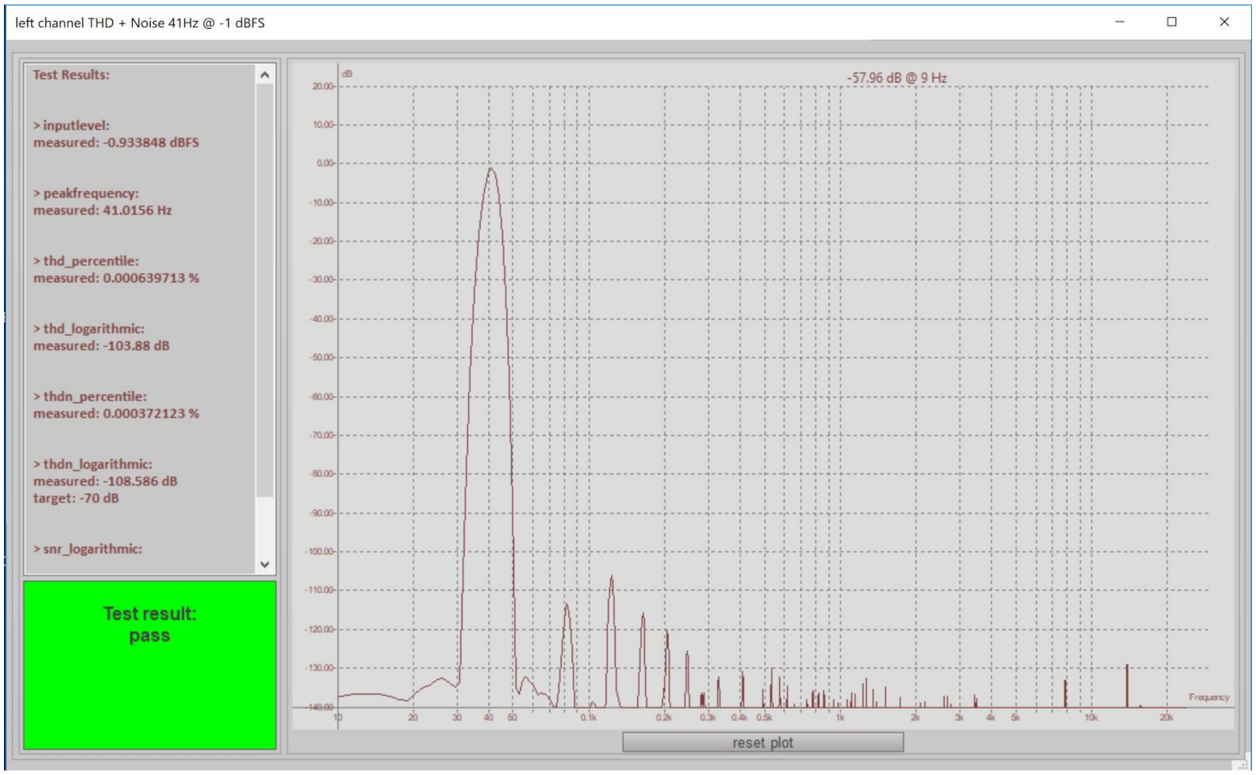
Parameters

testype	normal
signal	singlesine
analyser	xtalk
chidx	0
tonefreq	997
tonelevel	-1
detectionlevel	-150
fftlength	32768
fftnoavg	16
fftavgtype	exponential
outputfreqresponse	1
inttime	50
transtime	1000

Process log

```
response analysis 30 of 30
Test Event: fail
fail 30 of 30
Test Event: Test procedures finished
```

Double clicking on any test will open a window that contains the relevant performance metrics, results and a pass-fail outcome in one pane and a graph of the results in another pane. The user may place the cursor over the graph to see frequency and level information for a given location on the graph. The user may also select an area of the graph to zoom in on. Selecting "reset plot" will fit the results in the pane.


left channel THD + Noise 41Hz @ -1 dBFS

Test Results:

> inputlevel:
measured: -0.933848 dBFS

> peakfrequency:
measured: 41.0156 Hz

> thd_percentile:
measured: 0.000639713 %

> thd_logarithmic:
measured: -103.88 dB

> thdn_percentile:
measured: 0.000372123 %

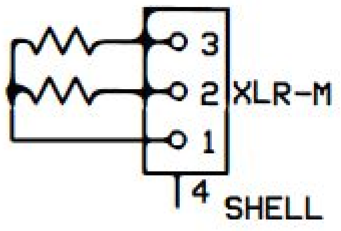
> thdn_logarithmic:
measured: -108.586 dB
target: -70 dB

> snr_logarithmic:

Test result:
pass

Cross-Talk Test

The cross-talk test requires special consideration and an accessory. The test method for cross-talk testing requires removing the cable from the channel under test and terminating the input of the ADC with a shorting plug. Below is an example schematic for an XLR shorting plug. The resistor values should be matched to the output impedance of the DAC/signal generator and matched within 5% of each other. The necessary pause in the test suite to perform this activity requires a pause function in the application. If you scroll down the list of tests in the default.xml you will note that there is a test labeled “user action”. This pauses the test suite and prompts the user to hit ok when the shorting plug is in place.

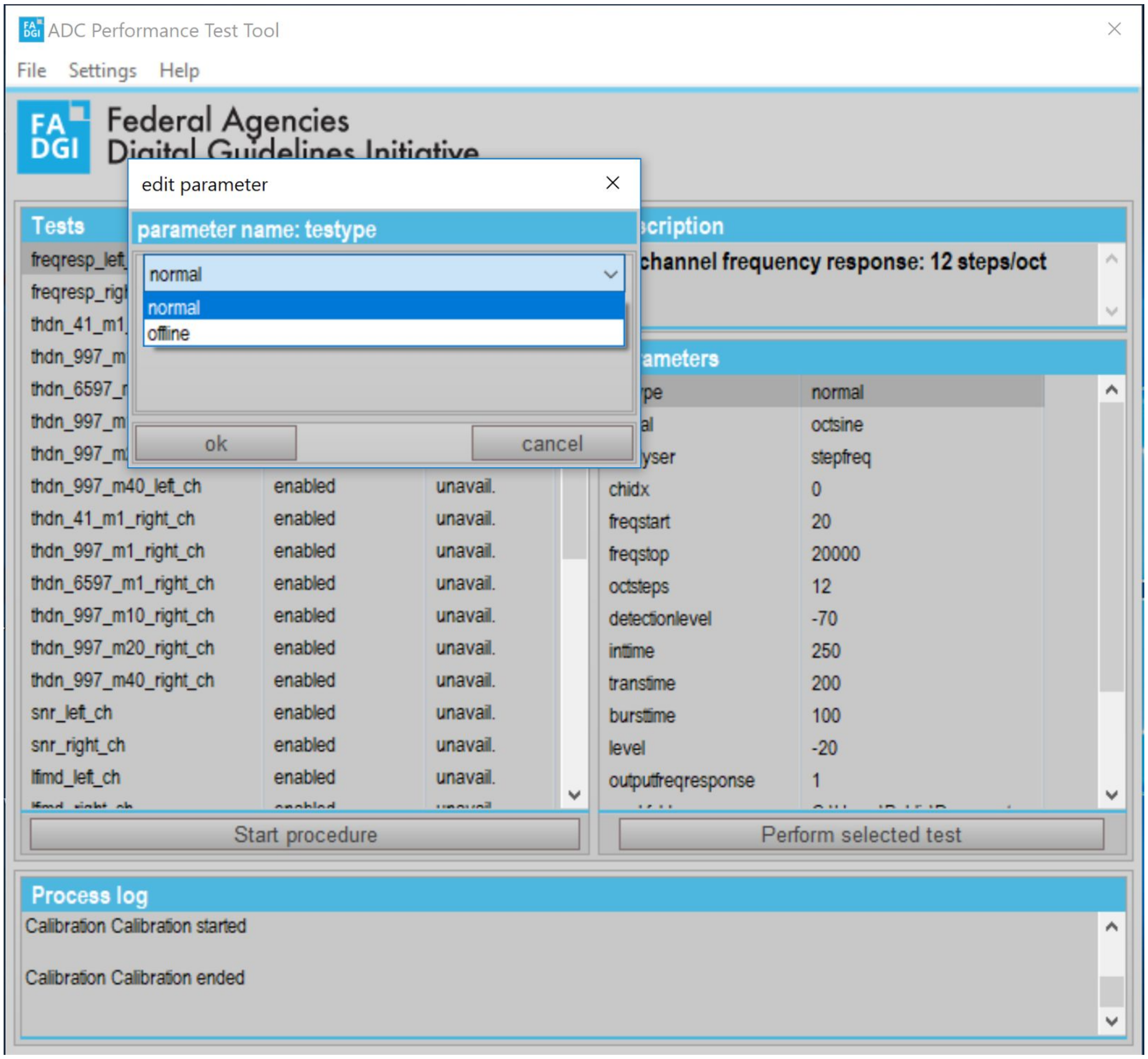


Schematic drawn by Phillip Sztenderowicz.

Performing Analysis and Reporting Only (Offline Test Type)

In cases where test signals have been generated and captured already, ADCTest can operate in an “offline” mode where it skips the step of generating test signals and performs analysis and reporting only. This mode of operation may be used by a client that has outsourced the digitization of their materials to a vendor. The client may provide test methods and ADC performance metrics that vendors must comply with. Vendors may generate signal files and capture response files using the same ADC employed in the digitization project and send these to the client. The client may then analyze the response files to ensure that the vendor’s ADC complies with the stated specifications.

In order to analyze a set of response files and generate the associated reporting a user can change the value for the parameter “testtype” to “offline”.



parameter name: testype

normal
offline

Tests

freqresp_left		
freqresp_righ		
thdn_41_m1		
thdn_997_m		
thdn_6597_		
thdn_997_m		
thdn_997_m		
thdn_997_m40_left_ch	enabled	unavail.
thdn_41_m1_right_ch	enabled	unavail.
thdn_997_m1_right_ch	enabled	unavail.
thdn_6597_m1_right_ch	enabled	unavail.
thdn_997_m10_right_ch	enabled	unavail.
thdn_997_m20_right_ch	enabled	unavail.
thdn_997_m40_right_ch	enabled	unavail.
snr_left_ch	enabled	unavail.
snr_right_ch	enabled	unavail.
lfmd_left_ch	enabled	unavail.

cription

channel frequency response: 12 steps/oct

ameters

pe	normal
al	octsine
yser	stepfreq
chidx	0
freqstart	20
freqstop	20000
octsteps	12
detectionlevel	-70
inttime	250
transtime	200
bursttime	100
level	-20
outputfreqresponse	1

Process log

```
Calibration Calibration started

Calibration Calibration ended
```

When this is done the test suite will look in the filepath identified in the “workfolder” parameter for the filename identified in the “responsfile” parameter to perform analysis and reporting. The values for these parameters can be updated in the associated XML file for each test. Once this is done the file may be opened in ADCTest and the user may select “Start procedure” to perform the analysis and testing.

Performing a Single Test

Sometimes users may want to run a single test instead of a full test suite. To do this the user must select the test they want to perform and then select “Perform selected test”.

Copyright and Licensing

ADCTest is in the public domain and is licensed under a 3-Clause BSD license.