USER MANUAL AND MATLAB SOFTWARE DEVELOPMENT KIT

CREARE CLINICAL HEARING ASSESSOR (CHA)

Prepared by:

Odile H. Clavier, Ph. D.
Principal Investigator

Jed C. Wilbur
Project Engineer

Disclaimer—Creare Receipt and Release Agreement

Creare has used commercially reasonable efforts in the performance of services under Contract Nos. W81XWH-13-C-0194 and W81XWH-14-C-1409 and in preparing this Software Development Kit (SDK) but makes no guarantee or warranty of any nature with regard to the use, performance or operation of the SDK or Clinical Hearing Assessor (CHA). Creare makes no representations or warranties, and Creare shall incur no liability or other obligation of any nature whatsoever to any person from any and all actions arising from the use of the Noise Recording System. THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE HEREBY EXPRESSLY EXCLUDED. The final responsibility for the proper use and functioning of the SDK and CHA delivered by Creare shall rest solely with the user.

Creare LLC
16 Great Hollow Road
Hanover, NH 03755

TM-~~3446A~~3446x
Projects 7114/7116
~~July~~ October 12, 2016

TABLE OF CONTENTS

Revision No.	SVN Rev No.	Up-to-Date as of	Initials	Note
Rev 1	-	-	-	Original
Rev 2	5587	7/7/2016	TMR	Adding new methods and exams; Adding the Wireless Headsets
Rev 2.1	5828	10/12/2016	JAN	Updating WiScreener On/Off

1 CLINICAL HEARING ASSESSOR OPERATION

Figure 1~~Figure 1~~ is a photograph of the Creare Hearing Assessment System. The system consists of the Creare Hearing Assessment (CHA) module, a calibrated headset or probe, and a PC interface. The system is also supported by Creare's Wireless Audiometry Headset—a headset with an integrated CHA circuit. Figure 2 is a detailed overview of the CHA itself. The CHA can function in wired (USB) or wireless (Bluetooth) modes. The CHA is unique relative to other PC-based systems in that the majority of the auditory test algorithms and all test media reside on the CHA—the PC is used only as a user interface and for database storage. Additionally, the CHA overcomes the challenging issue of calibration by storing calibration data in nonvolatile memory on the headset or probe.

Figure 1. The Clinical Hearing Assessment System. The system consists of a handheld Clinical Hearing Assessment (CHA) module, a calibrated headset, and a PC interface. The CHA can function in wired (USB) or wireless (Bluetooth) modes.

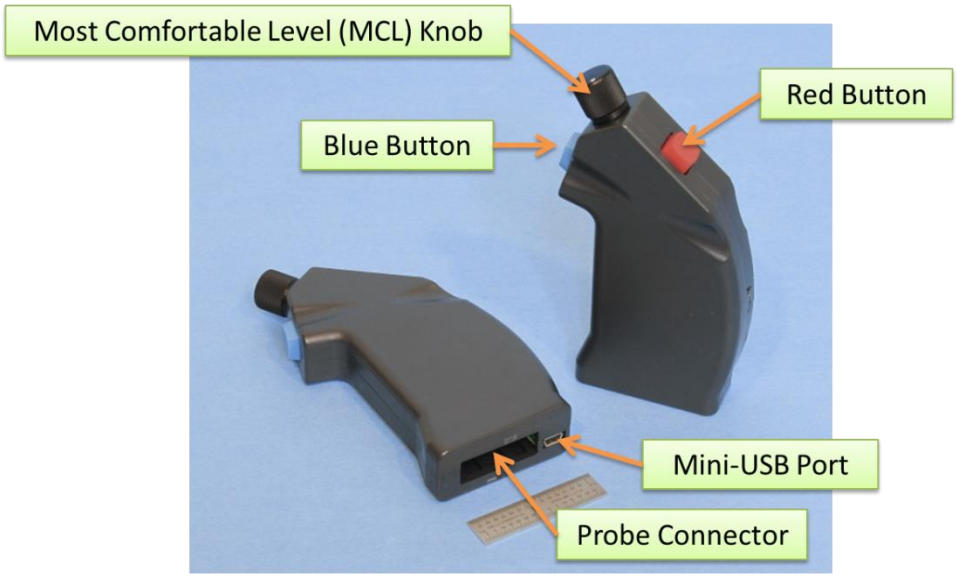

Figure 2. Overview of the Handheld CHA Device

1.1 HANDHELD CHA COMPONENTS

A novel feature of the CHA is the inclusion of a nonvolatile memory chip on the probe-side connector. Figure 3 is a photograph of the probe-side connector. The left side of the photograph shows the probe circuit board. The single chip visible on the board is a nonvolatile EEPROM for the storing of calibration data. The right side of the photograph shows the probe connector after wiring the probe cable and potting with a hard urethane.

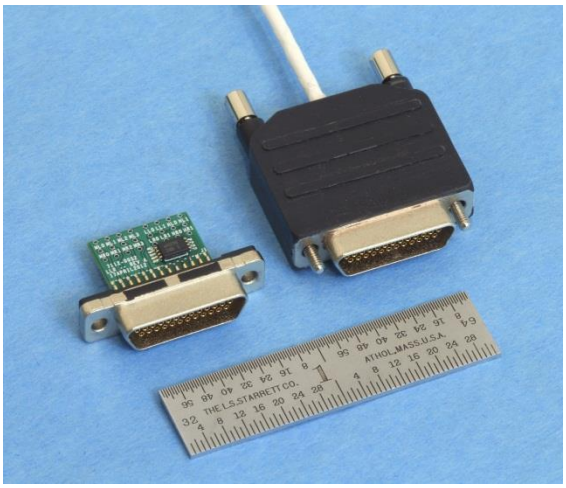

Figure 3. Photographs of the Probe-Side Connector. The left side of the photograph shows the probe circuit board. The single chip visible on the board is a nonvolatile EEPROM for the storing of calibration data. The right side of the photograph shows the probe connector after wiring the probe cable and potting with a hard urethane.

Additionally, there are two custom circuit boards for specific applications. Figure 4~~Figure 4~~ is a photograph of the calibration (left) and breakout (right) boards. On the calibration board a BNC connector is used to route in the signal from a calibrated microphone source. This signal is used as a reference for calibrating probe speakers and microphones. The breakout board is designed for interfacing the CHA with standard audio equipment. The breakout board has eight 3.5 mm TRS jacks, one for each microphone (pink), line in (blue), and headphone out (green) stereo pair.

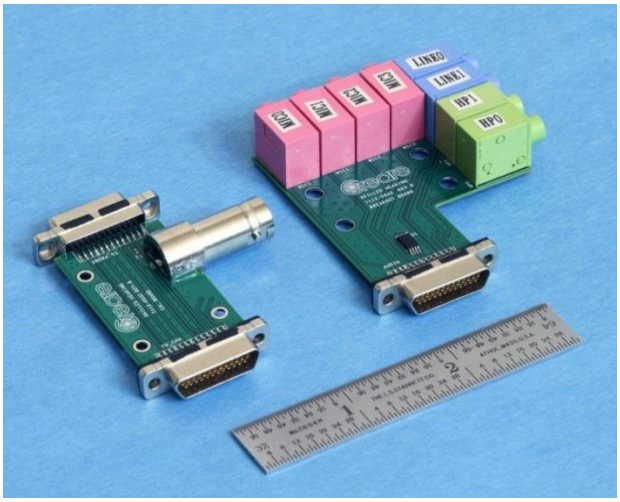

Figure 4. Photograph of CHA Accessory Connectors. At left is the calibration board; the BNC connector is used to route in the signal from a calibrated microphone source. At right is a generic breakout board; there are eight 3.5 mm TRS jacks—one for each microphone (pink), line in (blue), and headphone out (green) stereo pair.

1.2 CREARE WIRELESS AUDIOMETRIC HEADSET

The Creare Wireless Audiometric Headset (Figure 5), where a CHA is integrated into a noise-attenuating headset. Exams using the headset are administered over Bluetooth but can communicate with a computer via USB or Bluetooth when uploading media or configuring the device. The headset provides sufficient attenuation (see Appendix A) for audiometric exams to be performed outside of the sound booth in relatively quiet office-like spaces.

Figure 5. Photograph of the Creare Wireless Audiometric Headset

1.3 HANDHELD CHA USE

1.3.1 Attaching a Probe or Headset

Note: Please be certain that the CHA is turned "off" before attaching or removing a probe.

To attach a probe, insert the probe's multi-pin connector into the probe connector slot on the CHA. Tighten the jack screws to secure the probe.

1.3.2 Turning a Handheld CHA On or Off

Note: Please be certain that the CHA is turned "off" before attaching or removing a probe.

A CHA can be turned on in one of two ways. First, attaching the CHA to a PC via a USB cable will immediately start the CHA. The CHA cannot be turned off when connected via USB. Alternatively, the CHA can be turned on by depressing both the red and blue buttons simultaneously for six seconds. The CHA will then need to be connected to the PC via a Bluetooth connection.

1.4 CREARE WIRELESS AUDIOMETRIC HEADSET USE

1.4.1 Turning the Headset On and Off

The wireless headsets include a small switch inside the right ear cup. This switch disconnects the battery from the rest of the circuit and must be turned off when transporting the headsets per shipping regulations. The switch must be in the "On" position (Figure 6, ~~left~~right) before it can be used or charged. The switch may be left in the "On" position when not in transport.

The headset will enter a low-power idle automatically after 15 minutes of inactivity (see Section 2.2.23 to change this value) or when commanded over Bluetooth (see Section 2.2.25); communication with the headset is impossible during idle.

To wake the headsets from idle, simply tap the blue ear cup robustly—this will activate the "shake-to-wake" feature. If one looks closely, flashing green and blue lights may be visible through the black cloth covering the interior of the blue ear cup after the headset has booted up.

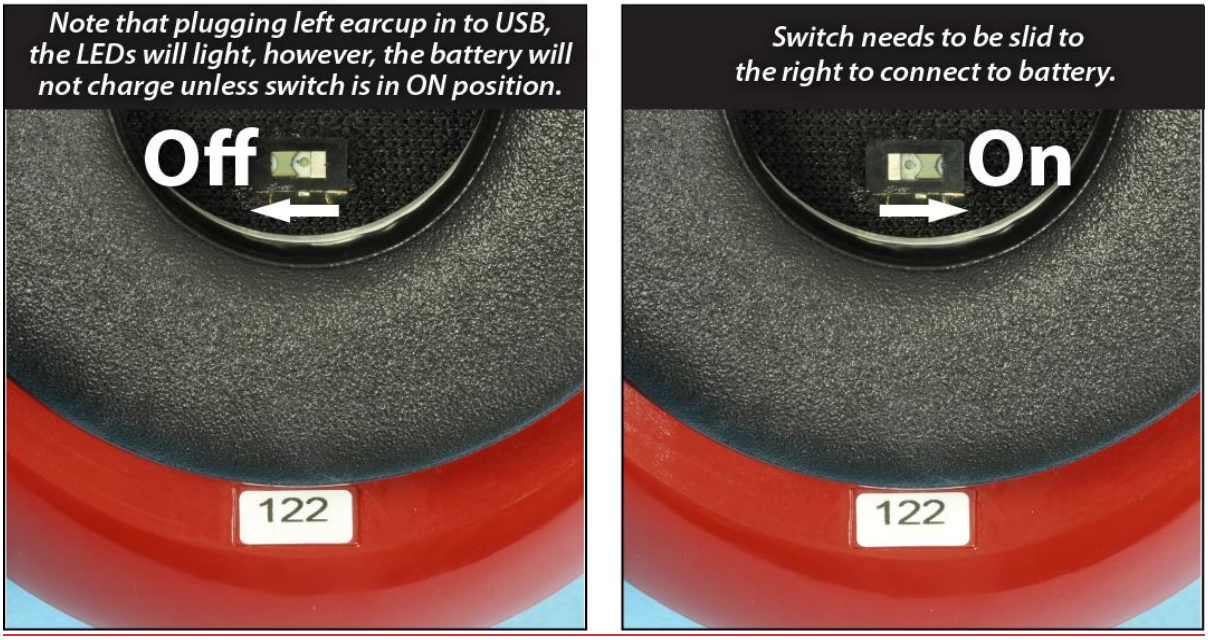

Figure 6. On/Off Position of the Switch in the Right Ear Cup

1.5 CONNECTING VIA USB

To connect to the CHA via USB, simply attach a USB cable between the CHA or headset and the PC. The handheld CHAs have a USB-Mini connection while the headsets have a USB-Micro connection. The CHA will automatically turn on and be ready for use.

1.6 CONNECTING VIA BLUETOOTH

Most modern computers will recognize and automatically install the proper drivers to connect the CHA via Bluetooth. Simply click on **Add a Device** in Control Panel—Bluetooth Devices. The pairing code for a CHA is 1234. See Section 2.2.15 for more details on working with Bluetooth.

Note: The handheld CHAs use a "Classic" Bluetooth (Bluetooth 3.0) interface while the Wireless Audiometric Headsets use Bluetooth "Low-Energy" (Bluetooth 4.0). Please ensure that your Bluetooth dongle is appropriate for your CHA device.

1.7 CHARGING A CHA'S BATTERY

The CHAs have an internal rechargeable lithium-polymer battery that must be charged before use. To charge a handheld CHA, simply plug it into a computer or USB charger via USB (Section 1.5~~1.4~~). To charge a wireless headset, ensure that the switch is in the "On" position (Section 1.4.1~~1.3.1~~) and plug it in over USB (Section 1.5~~1.4~~).

1.8 CALIBRATION CHECK

Calibration is an important part of any hearing assessment. A baseline calibration check—the response of a probe's input and output channels to a user-defined chirp—can be stored on the probe connector with the other calibration parameters. See Section 2.3.2 and the related example files for more details on the calibration check.

2 SOFTWARE DEVELOPMENT KIT

2.1 OVERVIEW

The software development kit contains a number of classes and functions useful for performing and building hearing assessment tests. These are discussed in more detail below.

2.2 CHAMI CLASS

The Clinical Hearing Assessor MATLAB® Interface (CHAMI) is a software class that allows communication with the CHA. CHAMI includes a number of methods which are described in the following subsections. A list of methods can be obtained by typing `methods chami` at the command line. Each method X can be called by invoking `chami.X`. Within MATLAB, help can be obtained by typing `help chami.X` on the command line.

A new CHA device connected to the computer can be initialized by using the call `cha_device=chami()`.

2.2.1 abort_exams()

This method is used to abort any exam currently being executed by the CHA, and clears the CHA's queue of any remaining exams. This is used in conjunction with the exam classes discussed in Section 2.3.

- Method Calls:
    - `cha_device.abort_exams`
- Inputs:
    - None
- Outputs:
    - None
- Example Scripts:
    - The following are some of the scripts that use the **abort_exam** method:
        - *HINT_example.m*
        - *HughsonWestlake_example.m*
        - *BekeseyLike_example.m*
        - *Dpgram_example.m*
        - *Map_example.m*
        - *CRM_example.m*

2.2.2 close

This method is used to close communication with a CHA.

- Method Calls:
  - `cha_device.close`
- Inputs:
  - None
- Outputs:
  - None
- Example Scripts:
  - All example scripts use this method

2.2.3 delete_file

This method is used to delete a file from the CHA's file system.

- Method Calls:
  - `cha_device.delete_file(`*`CHAFileName`*`)`
- Inputs:
  - *`CHAFileName`*: string with the name of the file to be removed from the CHA
- Outputs:
  - None
- Example Scripts:
  - *readWriteDelete_example.m* illustrates method functionality

2.2.4 dir

This method is similar to the MATLAB `dir` command. It returns information about files stored on the CHA file system.

- Method Calls:
  - *result* = cha_device.dir
  - [*result*, *space*] = cha_device.dir
- Inputs:
  - None
- Outputs:
  - *result*: a structure of length N, where N is the number of items on the file system, with the fields:
    - *result(n).name:* filename of the n^{th} file
    - *result(n).date*: modification date of the file
    - *result(n).bytes*: number of bytes allocated to the file
    - *result(n).isdir:* 1 if name is a directory and 0 if not
    - *result(n).datenum:* modification date as a MATLAB serial date number
  - *space*: number of available bytes on the SD card
- Example Scripts:
  - *readWriteDelete_example.m* calls this method

2.2.5 exam_submission

This exam is used to pass user-supplied responses for an exam to the CHA. For example, the user must tell the CHA if the subject correctly responded to a HINT presentation or not.

- Method Calls:
  - `cha_device.exam_submission(`*`exam, submission`*`)`
- Inputs:
  - *`exam`*: a previously defined exam class, see Section 2.3
  - *`submission`*: user response in an exam-specific format, for more information use call `help some_exam.createSubmission` where some_exam is the exam in question
- Outputs:
  - None
- Example Scripts:
  - *HINT_example.m* and *CRM_example.m* illustrate the use of this and the related methods

2.2.6 get_calibration

This method is used to get the calibration data stored on the CHA at the specified index.

- Method Calls:
  - *result* = cha_device.get_calibration(*index*)
- Inputs:
  - *index*: one-based index of the desired calibration
- Outputs:
  - *result*: cha_calibration object for that channel with the following properties:
    - *result.description*: description of the channel
    - *result.validInputs*: a 1 x 4 cell array cell array specifying the input channel; see **setup_pr** for a list of valid inputs; only one channel may be assigned
    - *result.validOutputs*: a 1 x 4 cell array cell array specifying the output channel; see **setup_pr** for a list of valid outputs; only one channel may be assigned
    - *result.amplifierGain_dB*: amplifier gain, in dB
    - *result.freqCalTable*: 85 x 2 table where the first column is the frequency and the second column is the stored calibration factor for that frequency
    - *result.limits*: a 1 x 1 structure with the following fields:
      - *result.limits.freq*: an array of frequencies at which RetSPL values are defined
      - *result.limits.RetSPL*: the RetSPL values associated with *freq*
      - *result.limits.old_min*: needed to support CHAs with legacy firmware
      - *result.limits.old_max*: needed to support CHAs with legacy firmware

- *result.speakerBaseline*: an instance of *chirp_exam*
- *result.minimumOutputLevels*: the minimum level the CHA can produce at each frequency in *freqCalTable*, dB SPL
- *result.maximumOutputLevels*: the maximum level the CHA can produce at each frequency in *freqCalTable*, dB SPL
- *result.microphoneBias*: microphone bias voltage, V

- Example Scripts:
  - *HINT_example.m* and *CRM_example.m* illustrate the use of this and the related methods

2.2.7 get_cha_count

This method is used to return the number of devices on an interface.

- Method Calls:
  - `result = cha_device.get_cha_count(intf)`
- Inputs:
  - `intf`: the interface to use; either ‘usb’, ‘Bluetooth’, or ‘ble’
  - If no inputs are passed, it defaults to (‘usb’)
- Outputs:
  - `result`: the number of CHAs connected to the specified interface
- Example Scripts:
  - *HINT_Bluetooth_example.m* illustrates the use of this method

2.2.8 get_exam_results

This method is used to query the CHA for the status of a particular exam, including exam results if available.

- Method Calls:
  - `results = cha_device.get_exam_results(exam, timeout)`
- Inputs:
  - `exam`: a previously defined exam class, see Section 2.3
  - `timeout:` amount of time to wait for results, in seconds, before an error is returned, optional
- Outputs:
  - `results`: a structure whose format is specific to the particular exam type (see Section 2.3)
- Example Scripts:
  - The following are some of the scripts that call this method:
    - *HINT_example.m*
    - *HughstonWestlake_example.m*
    - *BekeseyLike_example.m*
    - *Dpgram_example.m*
    - *Map_example.m*
    - *CRM_example.m*

2.2.9 get_id

This method is used to get CHA-specific information.

- Method Calls:
  - *result* = cha_device.get_id
- Inputs:
  - None
- Outputs:
  - *result:* a structure with the following fields:
    - *result.id:* a structure with the following fields:
      - *result.id.serialNumLsw*: least significant word of the CHA serial number
      - *result.id.serialNumMsw*: most significant word of the CHA serial number
      - *result.id.description*: a description of the particular CHA
      - *result.id.serialNum*: serial number of the CHA
    - *result.buildDateTime*: build date and time of the CHA firmware
    - *result.cpuId*: CHA CPU identifier
    - *result.electronicsRevision*: revision number of the CHA circuitry
- Example Scripts:
  - None

2.2.10 get_probe_id

This method is used to query the serial number and description of the probe or headset attached to the CHA.

- Method Calls:
  - *result* = cha_device.get_probe_id
- Inputs:
  - None
- Outputs:
  - *result:* a structure with the following fields:
    - *result.description*: descriptive string
    - *result.serialNum*: probe serial number
- Example Scripts:
  - None

2.2.11 get_settings

This method is used to query the CHA for the values of certain programmable settings.

- Method Calls:
  - `result = cha_device.get_settings`
- Inputs:
  - None
- Outputs:
  - *`result:`* a structure with the following fields:
    - *`result.autoOnBatteryShutdown`*: period of inactivity, in minutes, the CHA will remain on whilst not connected via USB
    - *`result.maxMcl`*: maximum level, in dB SPL for a sine wave spanning -1.0 to 1.0, allowed when controlling levels with the MCL knob
- Example Scripts:
  - *ChangeMaxMCL.m* is a function that calls this method

2.2.12 get_status

This method returns information about the status of the CHA, as discussed below.

- Method Calls:
  - `result = cha_device.get_status`
- Inputs:
  - None
- Outputs:
  - *result:* a structure with the following fields:
    - *result.state:* current mode the CHA is in
    - *result.flags:* raw bit flags set by the CHA (these are broken out in a more user-friendly fashion in the *flagBits* structure)
    - *result.lastCtrlError:* last error message reported by the CHA
    - *result.mcl:* current setting of the MCL knob (dB SPL)
    - *result.vUsb:* current voltage supplied by the USB connection (volts)
    - *result.vBattery:* current voltage measured across the onboard rechargeable battery (volts)
    - *result.samplesPlayed:* number of samples (at 96 kHz) played by the CHA since receiving the **push_play** command
    - *result.stateString:* current state of the CHA
    - *result.flagBits:* structure with the current bit status of the buttons:
      - *result.flagBits.blueButton:* current status of the blue button
      - *result.flagBits.redButton:* current status of the red button

- *result.flagBits.latchedBlueButton:* this bit will be high if the blue button has been pressed since the last **get_status** call
- *result.flagBits.latchedRedButton:* this bit will be high if the red button has been pressed since the last **get_status** call

The following *flagBits* are helpful for troubleshooting but are otherwise unused:

- *result.flagBits.i2cFail*
- *result.flagBits.configFail*
- *result.flagBits.fileSystemFail*
- *result.flagBits.btFail*
- *result.flagBits.btleFail*
- *result.flagBits.usbFail*
- *result.flagBits.aicFail*
- *result.flagBits.charging*
- *result.flagBits.usbConnected*
- *result.flagBits.bootloader*
- *result.flagBits.attenuator_L1*
- *result.flagBits.attenuator_L2*
- *result.flagBits.attenuator_R1*
- *result.flagBits.attenuator_R2*

- Example Scripts:
  - *PlaybackMultiTrack_example.m* uses **get_status** to monitor the MCL knob setting and the number of samples played by the CHA

- *ButtonState_example.m* uses **get_status** to poll and plot the CHA button states

2.2.13 list_calibrations

This method returns a list of calibration on the CHA.

- Method Calls:
  - *result* = cha_device.list_calibrations
- Inputs:
  - None
- Outputs:
  - *result*: a 16x1 structure with fields:
    - *result(n).validInputs*: 1 x 4 cell array specifying the input channel; see **setup_pr** for a list of valid inputs; only one channel may be assigned
    - *result(n).validOutputs*: 1 x 4 cell array specifying the output channel; see **setup_pr** for a list of valid outputs; only one channel may be assigned
    - *result(n).description*: string with channel description
    - Please note that for the majority of devices most of the channels will be empty. To view a specific calibration, see get_calibration.
- Example Scripts:
  - None

2.2.14 mkdir

This method creates a directory on the CHA file system.

- Method Calls:
  - `cha_device.mkdir(`*`path`*`)`
- Inputs:
  - *`path`*`:` path where the directory will be created
- Outputs:
  - None
- Example scripts:
  - None

2.2.15 open

This method opens a handle to the CHA. This method must be called prior to calling any other method.

- Method Calls:
  - `cha_device.open(interface, which)`
- Inputs:
  - *`interface:`* the interface to use; either ‘usb’, or one of two Bluetooth options: ‘bluetooth’ or ‘ble’; use ‘bluetooth’ for the handheld CHAs (traditional Bluetooth) and ‘ble’ for the Wireless Audiometric Headsets (Bluetooth Low Energy)
  - *`which:`* zero-based index of the CHA for ‘usb’; COM port for ‘bluetooth’
  - If no inputs are passed, it defaults to (‘usb’,0)
- Outputs:
  - None
- Example Scripts:
  - All example scripts call the open method
  - *HINT_Bluetooth_example.m* uses the Bluetooth interface, all others use the USB interface

2.2.16 play_with_noise

This is a high-level method that mixes two WAV files (a "signal" file and a "noise" file) at a given noise level (in dB SPL) and signal-to-noise ratio (SNR). The method accepts a "reference" file for either input files (or both) that can be used for scaling the output levels (see the **play_wrt_reference** method). Unlike the low-level methods, this method calculates the RMS level of the input data and scales the output accordingly to reach the target output levels—no additional user scaling or correction is required.

- Method Calls:
  - `default_options = cha_device.play_with_noise`
  - `cha_device.play_with_noise(desired_options)`
- Inputs:
  - `desired_options`: a structure with the following fields:
    - `desired_options.noiseFile`: string with the location and name of the "noise" WAV file; if calling a file stored on the CHA, use 'CHA:filename.wav'
    - `desired_options.noise_LEQ`: LEQ of the noise, dB SPL
    - `desired_options.noiseRef`: string with the location and name of a the noise level reference file, optional
    - `desired_options.signalFile`: string with the location and name of the "signal" WAV file
    - `desired_options.snr_dB`: SNR in dB
    - `desired_options.signalRef`: string with the location and name of a the signal level reference file, optional
    - `desired_options.signalDelay_sec`: delay, in seconds, between start of "noise" playback and start of mixing the "signal"
- Outputs:
  - `default_options`: a structure with the default field settings
- Example Scripts:
  - *playWithNoise_example.m* illustrates use of all method functionality

2.2.17 play_wrt_reference

A number of speech-in-noise tests used normalized target material: the actual level of an individual speech presentation may be scaled up or down during playback (relative to the nominal playback level) in order to obtain normative results similar to other target signals. In these cases, there exists a "reference" file that is used in determining the scaling factor for output playback. The Hearing in Noise Test (HINT) uses this approach. This method allows for playback of a signal file at an output level determined by the RMS level of a reference file. Unlike the low-level methods, this method calculates the RMS level of the input data and scales the output accordingly to reach the target output levels—no additional user scaling or correction is required.

- Method Calls:
    - `cha_device.play_wrt_reference(`*`referenceFile`*`, `*`signalFile`*`, `*`desiredLevel`*`)`
- Inputs:
    - *`referenceFile:`* string with the location and name of the "reference" WAV file; if calling a file stored on the CHA, use 'CHA:filename.wav'
    - *`signalFile:`* string with the location and name of the "signal" WAV file; if calling a file stored on the CHA, use 'CHA:filename.wav'
    - *`desiredLevel:`* Playback level in dB SPL
- Outputs:
    - None
- Example Scripts:
    - *PlayWrtReference_example.m* illustrates use of all method functionality

2.2.18 pop_recording

This method returns data recorded on an input channel that was previously initialized using the **setup_pr** method. Recording begins immediately after the **setup_pr** method is called.

- Method Calls:
  - `data = cha_device.pop_recording(timeout)`
- Inputs:
  - *`timeout:`* if no data are immediately available the function will wait up to *`timeout`* seconds until data are available or an error occurs, optional
- Outputs:
  - *`data:`* matrix with one row for each configured input channel and as many columns as the input block size (see **setup_pr** method)
- Example Scripts:
  - *simpleRecording_example.m* shows how to use **setup_pr** and **pop_recording** to simultaneously stream and record data

2.2.19 push_play

This method is used to start streaming data to the CHA for playback. It also initiates the start of recording if input channels were configured. The CHA must be configured using **setup_pr** prior to calling **push_play**.

- Method Calls:
  - `cha_device.push_play(data)`
- Inputs:
  - If `options.multitrack = false` in when calling **setup_pr:**
    - *`data:`* array of data containing the samples to be played by the CHA; `data` must have as many rows as the number configured output channels; values with an absolute value greater than 1.0 will be clipped; the output is scaled during playback such that a sine wave spanning -1.0 to 1.0 will have the output level specified during configuration via **setup_pr**
  - If `options.multitrack = true` in when calling **setup_pr:**
    - *`data:`* array of data containing the samples to be played by the CHA; `data` must have four rows:
      - First two rows: The output levels of these tracks (stereo) are determined by the *options.outputLEQ* input to **setup_pr**; a sine wave spanning -1.0 to 1.0 will have the specified level
      - Last two rows: The output levels of these tracks (stereo) are determined by the setting of the MCL knob in real time; a sine wave spanning -1.0 to 1.0 will have the level specified by the knob; use **get_status** to query the knob setting
- Outputs:
  - None
- Example Scripts:
  - *simplePlayback_example.m* shows how to configure the CHA using **setup_pr** and playback data at the desired level using **push_play**
  - *PlaybackMultitrack_example.m* shows how to use the method when multitrack playback is enabled

2.2.20 queue_exam

This method is used to add an exam to the CHA's execution queue. If no other exams are currently queued the exam will execute immediately.

- Method Calls:
  - `cha_device.queue_exam(`*`exam`*`)`
- Inputs:
  - *`exam`*: a previously defined exam class, see Section 2.3
- Outputs:
  - None
- Example Scripts:
  - The following are some of the scripts that use the **queue_exam** method:
    - *HINT_example.m*
    - *HughstonWestlake_example.m*
    - *BekeseyLike_example.m*
    - *Dpgram_example.m*
    - *Map_example.m*
    - *CRM_example.m*

2.2.21 read_file

This method reads a file stored on the CHA and writes it to a local disk.

- Method Calls:
  - `cha_device.read_file(`*`localFileName`*`,`*`CHAFileName`*`)`
- Inputs:
  - *`localFileName`*: a string with the location and file name for the file to be written locally
  - *`CHAFileName`*: the file name of the file to be retrieved from the CHA
- Outputs:
  - None
- Example Scripts:
  - *readWriteDelete_example.m* illustrates method functionality

2.2.22 set_mcl

This method is used to change the current MCL on the CHA.

- Method Calls:
  - `cha_device.set_mcl(`*`new_mcl`*`)`
- Inputs:
  - *`new_mcl`*: new MCL, in dB SPL
- Outputs:
  - None
- Example Scripts:
  - *PlaybackMultiTrack_example.m* illustrates method functionality

2.2.23 set_settings

This method is used to change system settings on the CHA.

- Method Calls:
  - `cha_device.set_settings(settings)`
- Inputs:
  - *`settings:`* a structure with the following fields:
    - *`settings.autoOnBatteryShutdown`*: period of inactivity, in minutes, the CHA will remain on whilst not connected via USB
    - *`settings.maxMcl`*: maximum level, in dB SPL for a sine wave spanning -1.0 to 1.0, allowed when controlling levels with the MCL knob
- Outputs:
  - None
- Example Scripts:
  - *ChangeMaxMCL.m* is a function that calls this method

2.2.24 setup_pr

This method is used to configure simultaneous playback and recording. If input channels are specified when calling this method, recording will begin immediately.

- Method Calls:
  - *default_options* = cha_device.setup_pr
  - cha_device.setup_pr(*desired_options*)
- Inputs:
  - *desired_options*: a structure with the following fields
    - *desired_options.inputs:* a 1 x 4 cell array specifying which input channels are to be used for recording; a calibration must exist for the given channel or unpredictable behavior may result; valid values are:
      - First element:
        - NONE: not used
        - SMICL0: left channel, microphone "0"
        - SMICL1: left channel, microphone "1"
        - SLINEL0: left channel, line-level input "0"
        - SMICR0: right channel, microphone "0"
        - DMIC1: differential input, microphone "1"
        - DLINE0: differential line-level input "0"
      - Second element:
        - NONE: not used
        - SMICR0: right channel, microphone "0"
        - SMICR1: right channel, microphone "1"
        - SLINER0: right channel, line-level input "0"
        - SMICL1: left channel, microphone "1"

- `DMIC0`: differential input, microphone “0”
- `DLINE0`: differential line-level input “0”

- Third and fourth elements:
  - Currently unsupported, reserved for the second CHA AIC

- *`desired_options.outputs`*: a 1 x 4 cell array specifying which output channels are to be used for playback; a calibration must exist for the given channel or unpredictable behavior may result; valid values are:
  - First element:
    - `NONE`: not used
    - `HPL0`: left channel, headphone “0”
  - Second element:
    - `NONE`: not used
    - `HPR0`: right channel, headphone “0”
  - Third and fourth elements:
    - Currently unsupported, reserved for the second CHA AIC

- *`desired_options.inputBlockSize`*: data block size for input channels
- *`desired_options.playbackPrequeueDepth:`* not yet implemented
- *`desired_options.outputLEQ:`* a 1 x 4 matrix with the desired LEQ in dB SPL for each of the four possible input channels; this is the level that a sine wave spanning -1.0 to 1.0 (0.707 RMS) would be played at
- *`desired_options.useMCL:`* a 1 x 4 Boolean matrix; if an element is true the corresponding channel’s LEQ will be set by the MCL knob setting

- `desired_options.multitrack`: Boolean controlling multitrack playback functionality. In multitrack playback four tracks are passed to the CHA using the **push_play** method: the first two tracks are played at the level specified by the *outputLEQ* field with the levels of the remaining two controlled by the MCL knob in real time; multitrack playback must be in stereo (two output channels must be defined above); see **push_play** for more information

- Outputs:
  - `default_options`: a structure with the default values for all fields
- Example Scripts:
  - *simplePlayback_example.m* uses the method for calibrated playback
  - *PlaybackMultitrack_example.m* shows how to use the multitrack playback functionality
  - *simpleRecording_example.m* illustrates use of the method for recording

2.2.25 shutdown

This method is used to shut down the CHA. Note that the CHA will not shut down if connected via USB.

- Method Calls:
  - `cha_device.shutdown`
- Inputs:
  - `None`
- Outputs:
  - None
- Example Scripts:
  - None

2.2.26 stop_pr

This method is used to stop playback and recording initiated by the **setup_pr** and **push_play** methods, freeing the CHA for other tasks.

- Method Calls:
  - `cha_device.stop_pr`
- Inputs:
  - None
- Outputs:
  - None
- Example Scripts:
  - *simplePlayback_example.m*, *PlaybackMultitrack_example.m*, and *simpleRecording_example.m* all call this method

2.2.27 write_file

This method is used to write a file to the CHA's SD card.

- Method Calls:
  - `cha_device.write_file(localFileName,CHAFileName)`
- Inputs:
  - `localFileName`: a string with the location and filename for the file to be copied to the CHA
  - `CHAFileName`: the filename to be used when writing to the CHA
- Outputs:
  - None
- Example Scripts:
  - *readWriteDelete_example.m* illustrates method functionality

2.3 EXAM CLASSES

A number of hearing exams have been implemented on the CHA. Exams in CHAMI are implemented as classes that interact with CHAMI methods described in the previous section. Example scripts are provided for each exam and are the best way to familiarize one's self with the different classes. Each exam is described briefly in the following sections.

Basic exam use is:

```
cha_device = chami();
cha_device.open()
this_exam = some_exam();
this_exam.property = new property;
cha_device.queue_exam(this_exam)
intermediate_result =
cha_device.get_exam_results(this_exam);
submission = this_exam.createSubmission(submission_inputs);
cha_device.exam_submission(this_exam, submission)
final_result = cha_device.get_exam_results(this_exam);
cha_device.close
```

2.3.2 BekesyLike_exam (6.0.2)

This class defines an audiometric exam using a Bekesy-like algorithm. The Bekesy-like algorithm requires the test subject to depress and hold either of the CHA buttons while they can hear a train of presentations. The exam is defined for a single frequency only; to perform an audiogram multiple calls to the exam at the desired frequencies must be made.

- Exam Properties:
    - *IncrementStart:* level step size in dB until *ReversalDiscard* reversals are reached
    - *IncrementNominal:* level step size in dB after *ReversalDiscard* reversals are reached
    - *ReversalDiscard:* number of initial reversals to ignore when computing the threshold
    - *ReversalKeep:* number of reversals to use when computing the threshold
    - *ToneGeneration:* a structure with the following fields:
        - *OctaveBandSize:* Fractional octave band size (e.g., if 3, then 1/3 octave band) of stimulus if *UseNthOctave* is true
        - *ToneRamp:* ramp-up and ramp-down time of each presentation, ms
        - *FDev:* warble tone frequency modulation deviation about nominal frequency, %
        - *FDevRate:* warble tone frequency modulation rate, Hz
        - *OutputChannel*: a 1 x 4 cell array specifying which output channels are to be used for playback; valid values are:
            - First element:
                - NONE: not used
                - HPL0: left channel, headphone “0”
            - Second element:
                - NONE: not used
                - HPR0: right channel, headphone “0”

- Third and fourth elements:
  - Currently unsupported, reserved for the second CHA AIC

- *ToneDuration:* duration of the tone, ms
- *UseNthOctave:* Boolean that, if true, uses 1/*OctaveBandSize* octave band-limited noise with center frequency *F*
- *FDevForm:* Frequency modulation functional form; only valid if *UseNthOctave* is false; one of the following:
  - *none:* a pure tone output (default)
  - *sine:* warble tone with sinusoidal frequency modulation
  - *triangle:* warble triangular frequency modulation

- *F:* test frequency, Hz
- *Lstart:* level of the first presentation, dB HL or dB SPL, depending on *LevelUnits* parameter
- *MaximumOutputLevel:* maximum allowable output level, dB SPL or dB HL, depending on *LevelUnits* parameter; note that the actual maximum level may be smaller depending on the sensitivity of the probe
- *MinimumOutputLevel:* minimum allowable output level, dB SPL or dB HL, depending on *LevelUnits* parameter; note that the actual minimum level may be higher depending on the sensitivity of the probe
- *ToneRepetitionInterval:* delay in ms from the end of one presentation until the start of the next
- *PresentationMax:* maximum number of presentations allowed
- *UnresponsiveMax:* maximum number of presentations allowed at the maximum or minimum allowable output level before the exam is terminated
- *LevelUnits:* sets dB reference scale; valid options are:
  - `dB HL` (default)
  - `dB SPL`

- *UseSoftwareButton:* Boolean that, when true, uses a submission input (instead of a button press) to record a subject response

- Submission Inputs:
  - `this_exam.createSubmission(`*`buttonState`*`)`
    - *buttonState*: current state of a software button: if the button is depressed, use a 1, otherwise use 0 (only valid when *UseSoftwareButton* is true)

- Results Properties:
  - *Threshold:* calculated hearing threshold, dB HL or dB SPL, depending on *LevelUnits* parameter
  - *Units:* passes the *LevelUnits* parameter
  - *ResultType:* one of the following:
    - *Threshold:* the exam ran normally and converged on a threshold
    - *Hearing Potentially Outside Measurable Range:* more than *UnresponsiveMax* consecutive presentations were presented without subject response at the maximum or minimum possible output level; this can occur if:
      - The subject may not have understood the instructions
      - The subject's hearing is better than what the exam settings could measure (i.e., they would reach the lowest possible sound level)
      - The subject's hearing is worse than what the exam settings could measure (i.e., they would reach the highest possible sound level)
    - *Failed to Converge:* the exam reached *PresentationMax* without converging on a threshold
  - *L:* array of presentation levels, dB HL or dB SPL, depending on *LevelUnits* parameter
  - *MaximumExcursion:* maximum difference between consecutive user responses that occurred after *ReversalDiscard* reversals, dB

- `RetSPL:` reference equivalent threshold sound pressure level for the current probe at `F`

- Example Scripts:
  - *BekesyLike_example.m*

2.3.3 BekesyFrequency_exam (6.0.2)

This class defines a fixed-level frequency threshold (FLFT) audiometric exam using a Bekesy-like algorithm. The Bekesy-like algorithm requires the test subject to depress and hold either of the CHA buttons while they can hear a train of presentations. In this case, the stimulus is presented at constant *level* and the *frequency* of the stimulus is changed based on the subject response (the frequency increases while the subject responds and decreases while the subject is not responding).

- Exam Properties:
  - *IncrementStart:* frequency step size in octaves until *ReversalDiscard* reversals are reached
  - *IncrementNominal:* frequency step size in octaves after *ReversalDiscard* reversals are reached
  - *ReversalDiscard:* number of initial reversals to ignore when computing the threshold
  - *ReversalKeep:* number of reversals to use when computing the threshold
  - *ToneGenerationLevel:* a structure with the following fields:
    - *Level:* level of each presentation, dB HL or dB SPL, depending on *LevelUnits* parameter
    - *OctaveBandSize:* fractional octave band size (e.g., if 3, then 1/3 octave band) of stimulus if *UseNthOctave* is true
    - *ToneRamp:* ramp-up and ramp-down time of each presentation, ms
    - *FDev:* warble tone frequency modulation deviation about nominal frequency, %
    - *FDevRate:* warble tone frequency modulation rate, Hz
    - *OutputChannel*: a 1 x 4 cell array specifying which output channels are to be used for playback; valid values are:
      - First element:
        - NONE: not used
        - HPL0: left channel, headphone “0”

- Second element:
  - `NONE:` not used
  - `HPR0:` right channel, headphone “0”
- Third and fourth elements:
  - Currently unsupported, reserved for the second CHA AIC

▪ *`ToneDuration:`* duration of the tone, ms

▪ *`UseNthOctave:`* Boolean that, if true, uses 1/*`OctaveBandSize`* octave band-limited noise with center frequency *`F`*

▪ *`FDevForm:`* frequency modulation functional form; only valid if *`UseNthOctave`* is false; one of the following:

- *`none:`* a pure tone output (default)
- *`sine:`* warble tone with sinusoidal frequency modulation
- *`triangle:`* warble triangular frequency modulation

o *`Fstart:`* frequency of the first presentation, Hz

o *`MaximumOutputFrequency:`* maximum allowable frequency, Hz; note that the actual maximum frequency may be smaller depending on the calibration of the probe

o *`MinimumOutputFrequency:`* minimum allowable frequency, Hz; note that the actual minimum frequency may be larger depending on the calibration of the probe

o *`ToneRepetitionInterval:`* delay in ms from the end of one presentation until the start of the next

o *`PresentationMax:`* maximum number of presentations allowed

o *`UnresponsiveMax:`* maximum number of presentations allowed at the maximum or minimum allowable output level before the exam is terminated

- *LevelUnits:* sets dB reference scale; valid options are:
  - dB HL (not recommended)
  - dB SPL (default)
- *UseSoftwareButton:* Boolean that, when true, uses a submission input (instead of a button press) to record a subject response

- Submission Inputs:
  - this_exam.createSubmission(*buttonState*)
    - *buttonState*: current state of a software button: if the button is depressed, use a 1, otherwise use 0 (only valid when *UseSoftwareButton* is true)

- Results Properties:
  - *Threshold:* calculated FLFT, Hz
  - *Units:* always *Hz*
  - *ResultType:* one of the following:
    - *Threshold:* the exam ran normally and converged on a threshold
    - *Hearing Potentially Outside Measurable Range:* more than *UnresponsiveMax* consecutive presentations were presented without subject response at the maximum or minimum possible output frequency; this can occur if:
      - The subject may not have understood the instructions
      - The subject's hearing is better than what the exam settings could measure (i.e., they would reach the highest possible frequency at the test level)
      - The subject's hearing is worse than what the exam settings could measure (i.e., they would reach the lowest possible frequency at the test level)
    - *Failed to Converge:* the exam reached *PresentationMax* without converging on a threshold
  - *F:* array of presentation frequencies, Hz

- `MaximumExcursion:` maximum difference between consecutive user responses that occurred after `ReversalDiscard` reversals, octaves

- Example Scripts:
  - *BekesyFrequency_example.m*

2.3.4 chirp_exam (9.0.1)

This class defines a chirp exam. In a chirp exam a chirp is played on one output channel while the response of one input channel is measured. This response can be compared against a "baseline" response if requested. Chirps can be used to check the calibration of a headset (if a calibration board and artificial ear are present) or a probe in a calibration check cavity. Chirp exams can also be used to qualitatively evaluate the "sealing" of a probe in a subject's ear.

- Exam Properties:
  - *F0:* chirp start frequency, Hz
  - *F1:* chirp end frequency, Hz
  - *F0comp:* start frequency for comparison to baseline, Hz
  - *F1comp:* end frequency for comparison to baseline, Hz
  - *Achirp:* amplitude of the chirp, dB SPL
  - *T:* length of chirp, seconds
  - *Tol_cal:* tolerance band for comparison to baseline, dB
  - *F0seal:* start frequency for the "seal check," Hz
  - *F1seal:* end frequency for the "seal check," Hz
  - *Aseal:* minimum measured level for passing "seal check," dB SPL
  - *Speaker:* channel of output device
  - *Nchirps:* number of chirps to emit and average
  - *Nsmooth:* number of points in running average
  - *Nbaseline:* minimum number of points to keep
  - *Input:* 1 x 4 cell array specifying the input channel; see **setup_pr** for a list of valid inputs; only one channel may be assigned
  - *Baseline:* Baseline spectrum for comparison; same format as the *spectrum* results property (see below)

- Submission Inputs:
    - None
- Results Properties:
    - *`Status:`* true if comparison to baseline passed, false otherwise
    - *`Seal:`* true if the “seal check” passed, false otherwise
    - *`Freq:`* frequency array, Hz
    - *`Spectrum:`* averaged measured spectrum level, dB SPL
- Example Scripts:
    - *ChirpCalibrationCheck_example.m*

2.3.5 crm_exam (6.0.1)

This class defines a Coordinate Response Measures (CRM) exam.

- Exam Properties:
  - *ConditionPresentations*: array specifying the number of presentations at each CRM condition, the format is: [*condition_number number_of_presentations; condition_number number_of_presentations; …*]; condition numbers are:
    - 1: + 6dB target SNR
    - 2: different gender talkers
    - 3: spatially separated talkers
  - *UseMcl:* if true the output level is controlled by the MCL knob setting
  - *Level:* output level (if *UseMcl* is false) in dB SPL
  - *MaxResponseTime:* maximum allowed response time, seconds
- Submission Inputs:
  - this_exam.createSubmission(*color, number, pause*)
    - *color*: subject's response color, format is:
      - 1: blue
      - 2: red
      - 3: white
      - 4: green
    - *number*: subject's response number
    - *pause*: if true the exam will be paused, pass a false to resume the exam

- Results Properties:
    - *`State:`* current state of the exam, one of the following:
        - `LOADING:` the exam is loading
        - `PLAYING:` the presentation is playing
        - `WAITING_FOR_RESULT:` the CHA is waiting for an input submission
        - `PAUSED:` the exam is paused
        - `DONE:` the exam has completed
    - *`ConditionCode:`* condition code (see *`ConditionPresentations`*) for the current presentation
    - *`CorrectColor:`* color code for the current presentation
    - *`CorrectNumber:`* number for the current presentation
    - *`WavFileName:`* filename for the current presentation
    - *`ResponseTime:`* response time in seconds
    - *`CorrectPercent:`* array of condition codes and the subject's score for each code, last value is the overall successful percentage
    - *`FinalScore:`* overall score for the exam
- Example Scripts:
    - *CRM_example.m*

2.3.6 dpoae_exam (3.1.2)

This class defines a DPOAE exam. Multiple DPOAE exams can be stacked in the CHA queue to perform DP-grams or DP-maps.

- Exam Properties:
  - *F1:* frequency of the first tone, Hz
  - *F2:* frequency of the second tone, Hz
  - *L1:* level of the first tone, dB SPL
  - *L2:* level of the second tone, dB SPL
  - *MinDpNoiseFloorThresh:* when the low DP exceeds the noise floor by this amount (in dB), the test will conclude (provided the *MinTestAverages* have been met)
  - *TransientDiscard:* initial period of data discarded at start of tone, in milliseconds
  - *MinTestAverages:* minimum number of blocks to consider for averaging
  - *MaxTestAverages:* maximum number of blocks to consider for averaging; ignored when *NoiseRejection* is `false`
  - *BlockSize:* the number of samples in a block used for the FFT; must be a power of 2
  - *NoiseHalfBandwidth:* bandwidth, in Hz, over which to calculate the noise floor
  - *NoiseRejection:* if `true` a noise rejection algorithm is used to discard noisy data blocks
  - *DisableSpectrum:* if `true`, the raw FFT spectrum feature is disabled
  - *InputChannel:* 1 x 4 cell array specifying the input channel; see **setup_pr** for a list of valid inputs; only one channel may be assigned
- Submission Inputs:
  - None

- Results Properties:
    - *DpLow:* structure containing the following information about the measured lower-frequency DP:
        - *Frequency:* actual measured frequency, Hz
        - *Amplitude:* measured amplitude, dB SPL
        - *Phase:* measured phase of the signal, radians
        - *NoiseFloor:* noise floor near the signal, dB SPL
    - *DpHigh:* a similar structure for the higher DP frequency
    - *F1:* a similar structure for the F1 frequency
    - *F2:* a similar structure for the F2 frequency
    - *TestAverages:* actual number of blocks averaged
    - *FftSpectrum:* structure with the following fields:
        - *Frequencies:* array of frequency values for the FFT
        - *FFT:* complex FFT response, counts
- Example Scripts:
    - *Dpgram_example.m* uses the **dpoae_exam** class to produce a DP-gram
    - *Map_example.m* uses the **dpoae_exam** class to produce a DP-map

2.3.7 gap_exam (7.3)

This class defines a gap-in-noise exam. In a gap-in-noise exam the subject is asked to respond (via button press) to a gap in otherwise white noise.

- Exam Properties:
  - *Channel:* output channel; `0` for left, `1` for right, `2` for both
  - *TimePres:* total length of each noise presentation, ms
  - *LNoise:* presentation level, dBA SPL
  - *TimeLead:* length of leading delay before a gap can be inserted, ms
  - *TimeTrail:* length of trailing delay needed after a gap, ms
  - *TimeWindow:* length of window during which a response can be accepted, ms
  - *TimeNoResp:* delay between the beginning of gap and beginning of response window, ms
  - *TimePause:* time between presentations, ms
  - *GapLengthStartIndex:* zero-based index of the `AllowableGapLengths` array for the starting presentation
  - *NReversalsCalc:* number of reversals to use in computation of threshold
  - *NReversals:* number of reversals to complete the exam
  - *NLowestReversals:* number of "lowest reversals" to track for second threshold; see the `GapLowestThreshold` result property
  - *NPresMax:* maximum number of presentations before aborting the exam
  - *NHits:* number of consecutive hits (valid responses) necessary before reducing gap length
  - *NMiss:* number of consecutive misses (invalid or non-responses) necessary before increasing gap length
  - *NPresCheck:* number of allowed consecutive presentations with the same gap length (will increase gap length if this is reached)
  - *MaxFreq:* maximum frequency of the white noise

- *AllowableGapLengths:* array of allowable gap lengths, ms

- Submission Inputs:
  - None

- Results Properties:
  - *State:* current state of the CHA, one of the following:
    - IN PROGRESS: the exam is running
    - SUCCESS: the exam has finished and returned a valid threshold
    - FAILED: the exam has finished without returning a valid threshold
  - *PresentationCount:* the number of completed presentations
  - *HitOrMiss:* Boolean that is true if the subject responded successfully, false if either (a) he subject has not yet responded or (b) the subject responded incorrectly
  - *CurrentGapStartTime:* location of the gap in the current presentation, in ms, measured from the start of the noise
  - *CurrentGapLength:* length of the gap in the current presentation, ms
  - *CurrentTimeResp:* response time measured from the end of the gap to the response of the subject, ms
  - *PlayPosition:* time elapsed since the beginning of the presentation, ms
  - *ActualMaxFreq:* actual cutoff frequency of the flow-pass filter used to generate the noise
  - *GapThreshold:* average gap length calculated based on the last NReversalsCalc reversals, ms
  - *GapLowestThreshold:* average gap length of the shortest NLowestReversals of the last NReversalsCalc reversals, ms
  - *GapLengthArray:* array of the gap durations presented during the exam, ms
  - *TimeRespArray:* array of the subject's response times, ms

- *`HitOrMissArray:`* array of Booleans that are `true` for hits (correct responses) and `false` for misses (incorrect or non-responses)

- *`ReversalUseForThresholdArray:`* array of Booleans that are `true` if a presentation is a reversal and used in calculating the threshold(s) and `false` otherwise

- Example Scripts:

  - *Gap_example.m* uses the **gap_exam** class to run a simple gap-in-noise test with a real-time status display

2.3.8 hint_exam (1.1.2)

Exam class for a Hearing in Noise Test (HINT). **This exam is currently licensed for research (i.e., not clinical) purposes only.**

- Exam Properties:
  - *`List:`* type of target material to use, must be one of the following:
    - `'normal'`: standard American English HINT lists
    - `'practice'`: American English practice lists
    - `'military'`: uses military-relevant phrases for presentations; note that these lists are not normalized
    - `'swahili'`: Swahili lists for presentations; note that these lists are not normalized
  - *`Direction:`* apparent direction of the noise source
    - `'front'`: noise appears located in front of the subject
    - `'left'`: noise appears located to the left of the subject
    - `'right'`: noise appears located to the right of the subject
    - `'quiet'`: no noise is played, see below for notes on the levels
  - *`DisableRepeatFirstUntilCorrect:`* the HINT algorithm, by default, repeats the first presentation (at increasing SNR) until the subject correctly answers the entire sentence. Setting this Boolean true disables this behavior: the first presentation will only be played at most once.
  - *`NoiseLevel:`* the SPL of the noise in dBA; if *`Direction`* is set to `'quiet'`, the level of the target is played at an SNR calculated relative to this level
  - *`StepSize:`* absolute increment, in dB, that the SNR is stepped between presentations (e.g., the SNR *decreases* by this amount if the previous presentation was correctly identified and it *increases* by this amount if the previous presentation was incorrectly identified)
  - *`InitialSNR:`* SNR, in dB, of the first presentation

- *ListNumber:* number of the target material list, useful if testing more than one condition without repeating the same list. Default behavior (*ListNumber* = 0) is to randomly select a list.
- *NumberOfPresentations:* number of presentations presented

- Submission Inputs:
  - this_exam.createSubmission(*subject_response*)
  - *subject_response*: one of the following:
    - Boolean array with one element for each word (true if the subject got the word correct, false otherwise)
    - **OR:**
    - Single Boolean (true if the subject got the entire sentence correct, false otherwise)

- Results Properties:
  - *State:* current state of the CHA, one of the following:
    - WAITING_FOR_RESULT: awaiting an input submission
    - PLAYING: playing a presentation
    - LOADING: loading a file for playback
    - DONE: the exam has completed
  - *SentencePath:* filename of the current presentation
  - *CurrentSentence:* string with the text of the current presentation
  - *ListLength:* number of total phrases in the current list
  - *CurrentSentenceIndex:* index of the current presentation
  - *sSRT:* the subject's speech reception threshold, dB SNR (dB SPL if *Direction* is set to 'quiet'). The speech reception threshold is the mean SNR of all presentations after the fifth.

- Example Scripts:
    - *HINT_example.m*

2.3.10 hughsonwestlake_exam (10.0.3)

This class defines an audiometric exam using the Hughson-Westlake algorithm. The Hughson-Westlake algorithm requires the test subject to depress either of the CHA buttons when they hear a short sequence of tones. The exam is defined for a single frequency only; to perform an audiogram multiple calls to the exam must be made as shown in the example script.

- Exam Properties:
    - *StepSize:* smallest level increment, dB
    - *Screener:* Boolean that, if true, uses the screener version of the Hughson-Westlake level exam
    - *PollingOffset:* period beyond last pulse where subject response still accepted, ms
    - *MinISI:* minimum value for inter-stimulus interval, ms
    - *MaxISI:* maximum value for inter-stimulus interval, ms
    - *TonePulseNumber:* number of tone pulses per presentation
    - *NumCorrectReq:* number of correct responses required to pass (only used when *Screener* is true)
    - *ToneGeneration:* a structure with the following fields:
        - *OctaveBandSize:* fractional octave band size (e.g., if 3, then 1/3 octave band) of stimulus if *UseNthOctave* is true
        - *ToneRamp:* ramp-up and ramp-down time of each presentation, ms
        - *FDev:* warble tone frequency modulation deviation about nominal frequency, %
        - *FDevRate:* warble tone frequency modulation rate, Hz
        - *OutputChannel*: a 1 x 4 cell array specifying which output channels are to be used for playback; valid values are:
            - First element:
                - NONE: not used
                - HPL0: left channel, headphone “0”

- Second element:
  - `NONE:` not used
  - `HPR0:` right channel, headphone “0”
- Third and fourth elements:
  - Currently unsupported, reserved for the second CHA AIC

▪ *`ToneDuration:`* duration of the tone, ms

▪ *`UseNthOctave:`* Boolean that, if true, uses 1/*`OctaveBandSize`* octave band-limited noise with center frequency *`F`*

▪ *`FDevForm:`* frequency modulation functional form; only valid if *`UseNthOctave`* is false; one of the following:

- *`none:`* a pure tone output (default)
- *`sine:`* warble tone with sinusoidal frequency modulation
- *`triangle:`* warble triangular frequency modulation

o *`F:`* test frequency, Hz

o *`Lstart:`* level of the first presentation, dB HL or dB SPL, depending on *`LevelUnits`* parameter

o *`MaximumOutputLevel:`* maximum allowable output level, dB HL or dB SPL, depending on *`LevelUnits`* parameter; note that the actual maximum level may be smaller depending on the sensitivity of the probe

o *`MinimumOutputLevel:`* minimum allowable output level, dB HL or dB SPL, depending on *`LevelUnits`* parameter; note that the actual minimum level may be higher depending on the sensitivity of the probe

o *`ToneRepetitionInterval:`* delay in ms from the end of one presentation until the start of the next

o *`PresentationMax:`* maximum number of presentations allowed

o *`UnresponsiveMax:`* maximum number of presentations allowed at the maximum or minimum allowable output level before the exam is terminated

- *LevelUnits:* sets dB reference scale; valid options are:
  - dB HL (default)
  - dB SPL
- *UseSoftwareButton:* Boolean that, when true, uses a submission input (instead of a button press) to record a subject response

- Submission Inputs:
  - this_exam.createSubmission(*buttonState*)
    - *buttonState*: current state of a software button: if the button is depressed, use a 1, otherwise use 0 (only valid when *UseSoftwareButton* is true)

- Results Properties:
  - *Threshold:* calculated hearing threshold, dB HL or dB SPL, depending on *LevelUnits* parameter
  - *Units:* passes the *LevelUnits* parameter
  - *ResultType:* one of the following:
    - *Threshold:* the exam ran normally and converged on a threshold
    - *Hearing Potentially Outside Measurable Range:* more than *UnresponsiveMax* consecutive presentations were presented without subject response at the maximum or minimum possible output level; this can occur if:
      - The subject may not have understood the instructions
      - The subject's hearing is better than what the exam settings could measure (i.e., they would reach the lowest possible sound level)
      - The subject's hearing is worse than what the exam settings could measure (i.e., they would reach the highest possible sound level)
    - *Failed to Converge:* the exam reached *PresentationMax* without converging on a threshold

- `L:` array of presentation levels, dB HL or dB SPL, depending on `LevelUnits` parameter
- `RetSPL:` reference equivalent threshold sound pressure level for the current probe at `F`
- `FalsePositive:` array of responses to each presentation that occurred outside the polling time window
- `NumCorrectResp:` number of presentations correctly answered (only used when `Screener` is true)

- Example Scripts:
  - *HughsonWestlake_example.m*

2.3.11 hughsonwestlakefrequency_exam (7.0.0)

This class defines a fixed-level frequency threshold (FLFT) audiometric exam using the Hughson-Westlake algorithm (in frequency space, not level space). The Hughson-Westlake algorithm requires the test subject to depress either of the CHA buttons when they hear a short sequence of tones. In this case, the stimulus is presented at constant *level* and the *frequency* of the stimulus is changed based on the subject response (the frequency increases while the subject responds and decreases while the subject is not responding).

- Exam Properties:
  - *StepSize:* smallest frequency increment, fractional octave band size (e.g., if 3, then 1/3 octave band)
  - *PollingOffset:* period beyond last pulse where subject response still accepted, ms
  - *MinISI:* minimum value for inter-stimulus interval, ms
  - *MaxISI:* maximum value for inter-stimulus interval, ms
  - *TonePulseNumber:* number of tone pulses per presentation
  - *ToneGenerationLevel:* a structure with the following fields:
    - *Level:* level of each presentation, dB HL or dB SPL, depending on *LevelUnits* parameter
    - *OctaveBandSize:* fractional octave band size (e.g., if 3, then 1/3 octave band) of stimulus if *UseNthOctave* is true
    - *ToneRamp:* ramp-up and ramp-down time of each presentation, ms
    - *FDev:* warble tone frequency modulation deviation about nominal frequency, %
    - *FDevRate:* warble tone frequency modulation rate, Hz
    - *OutputChannel*: a 1 x 4 cell array specifying which output channels are to be used for playback; valid values are:
      - First element:
        - NONE: not used
        - HPL0: left channel, headphone “0”

- Second element:
  - `NONE:` not used
  - `HPR0:` right channel, headphone “0”
- Third and fourth elements:
  - Currently unsupported, reserved for the second CHA AIC

▪ *`ToneDuration:`* duration of the tone, ms

▪ *`UseNthOctave:`* Boolean that, if true, uses 1/*`OctaveBandSize`* octave band-limited noise with center frequency *`F`*

▪ *`FDevForm:`* frequency modulation functional form; only valid if *`UseNthOctave`* is false; one of the following:

- *`none:`* a pure tone output (default)
- *`sine:`* warble tone with sinusoidal frequency modulation
- *`triangle:`* warble triangular frequency modulation

o *`Fstart:`* frequency of the first presentation, Hz

o *`MaximumOutputFrequency:`* maximum allowable frequency, Hz; note that the actual maximum frequency may be smaller depending on the calibration of the probe

o *`MinimumOutputFrequency:`* minimum allowable frequency, Hz; note that the actual minimum frequency may be larger depending on the calibration of the probe

o *`ToneRepetitionInterval:`* delay in ms from the end of one presentation until the start of the next

o *`PresentationMax:`* maximum number of presentations allowed

o *`UnresponsiveMax:`* maximum number of presentations allowed at the maximum or minimum allowable output level before the exam is terminated

- *LevelUnits:* sets dB reference scale; valid options are:
  - dB HL (not recommended)
  - dB SPL (default)
- *UseSoftwareButton:* Boolean that, when true, uses a submission input (instead of a button press) to record a subject response

- Submission Inputs:
  - this_exam.createSubmission(*buttonState*)
    - *buttonState*: current state of a software button: if the button is depressed, use a 1, otherwise use 0 (only valid when *UseSoftwareButton* is true)

- Results Properties:
  - *Threshold:* calculated FLFT, Hz
  - *Units:* always *Hz*
  - *ResultType:* one of the following:
    - *Threshold:* the exam ran normally and converged on a threshold
    - *Hearing Potentially Outside Measurable Range:* more than *UnresponsiveMax* consecutive presentations were presented without subject response at the maximum or minimum possible output frequency; this can occur if:
      - The subject may not have understood the instructions
      - The subject's hearing is better than what the exam settings could measure (i.e., they would reach the highest possible frequency at the test level)
      - The subject's hearing is worse than what the exam settings could measure (i.e., they would reach the lowest possible frequency at the test level)
    - *Failed to Converge:* the exam reached *PresentationMax* without converging on a threshold

- `F:` array of presentation frequencies, Hz
- `FalsePositive:` array of responses to each presentation that occurred outside the polling time window

- Example Scripts:
  - *HughsonWestlakeFrequency_example.m*

2.3.12 playsound_exam (1.0.0)

Playsound is not a hearing exam, but rather a useful feature that uses the exam class structure. The exam is used to directly play a WAV file stored on the CHA. Note that the WAV file sample rate must be 24, 48, or 96 (mono only) kHz.

- Exam Properties:
  - *Leq:* a 1 x 4 matrix with the desired LEQ in dB SPL for each of the four possible output channels; this is the level that a sine wave spanning -1.0 to 1.0 (0.707 RMS) would be played at
  - *SoundFileName:* path and filename of the target file; note that media uploaded using the **write_file** method are stored in the 'user' directory
- Submission Inputs:
  - `this_exam.createSubmission(`*pause*`)`
    - *pause*: use a 1 to pause the exam, and a 0 to resume the exam
- Results Properties:
  - *State:* the current state of the CHA, one of the following:
    - `OFF`: the CHA has not started the exam yet; this is usually an error condition
    - `PLAYING`: the file is being played
    - `LOADING`: the file is being loaded
    - `DONE`: the CHA is finished playing the file
    - `PAUSED`: the CHA has paused
- Example Scripts:
  - *PlaySound_example.m*

2.3.13 soae_exam (4.1.0)

This class defines a SOAE exam.

- Exam Properties:
  - *BlockSize:* samples per block, must be one of: 512, 1024, 2048, 4096, 8192
  - *SampleRate:* the CHA decimates its native 96 kHz sampling to this sample rate, must be one of: 16000, 24000, 36000, 48000
  - *MedianFilterWidth:* width of median filter used to calculate the noise floor during spectrum flattening
  - *MaxNumRejectedBlocks:* if this many blocks are rejected, the exam fails
  - *NumAcceptedBlocksRequired:* once this many blocks are accepted, the exam terminates successfully
  - *MaxNumPeaks:* the algorithm raises the peak threshold until the number of peaks returned is less than or equal to this number
  - *MaxLeq:* blocks with an LEQ greater than this are rejected
  - *SearchFrequencyMinHz:* minimum frequency of interest, Hz
  - *SearchFrequencyMaxHz:* maximum frequency of interest, Hz
  - *PeakThresholdDb:* the peak-finding algorithm uses this as an initial threshold, dB SPL
  - *InputChannel:* 1 x 4 cell array specifying the input channel; see **setup_pr** for a list of valid inputs; only one channel may be assigned
- Submission Inputs:
  - None
- Results Properties:
  - *State:* the current state of the CHA, one of the following:
    - DONE: the exam has completed
    - OFF: the CHA has not started the exam yet; this is usually an error condition

        - `RUNNING`: data are being collected
        - `PROCESSING`: data are being processed
    - *`NumBlocksAccepted:`* number of blocks were accepted for use in ensemble-averaged FFT
    - *`NumBlocksRejected:`* number of blocks rejected by the algorithm
    - *`FftFreqs:`* array of frequency values for the raw FFT, Hz
    - *`RawFFT:`* array of raw FFT magnitudes, dB SPL
    - *`NoiseFloor:`* array of calculated noise floor values, dB SPL
    - *`NormalizedFFT:`* FFT in SNR units, dB SPL
    - *`Leqs:`* LEQ of each processed block
    - *`BlockFOMs:`* figure of merit for each processed block
    - *`OverallFOMs:`* figures of merit for the ensemble
    - *`Peaks:`* a structure containing information on detected SOAEs with the following fields
        - *`Freqs:`* array of frequencies where SOAEs were detected
        - *`NormalizedMag:`* array of SOAE SNR values, dB
        - *`RawMag:`* array of SOAE magnitudes, dB SPL
- Example Scripts:
    - *SOAE_example.m*

2.3.14 thirdoctavebands_exam (6.0.2)

This exam class is used to measure the SPL in third octave bands recorded on the ‘MICR0’ channel.

- Exam Properties:
  - *`BufferLength:`* minimum number of samples to consider for third octave result
  - *`InputChannel:`* 1 x 4 cell array specifying the input channel; see **setup_pr** for a list of valid inputs; only one channel may be assigned
- Submission Inputs:
  - None
- Results Properties:
  - *`Frequencies:`* a vector of frequencies for which bands were computed, Hz
  - *`Leq:`* the equivalent sound level in each band, dB SPL
- Example Scripts:
  - *ThirdOctaveBands_example.m*

2.3.15 threedigit_exam (9.3.0)

This class defines a three-digit exam. It is a speech test where the target stimuli is three-digit triplets such as 3-5-9, and there may or may not be background noise.

- Exam Properties:
  - *nPresentations:* number of presentations
  - *warmupN:* number of presentations during the warm-up period
  - *targetType:* type of target material; either 'filtered' (default), 'timeCompressed', 'H3CamFiltered', 'TFS', or 'Swahili'
  - *warmupMasker:* type of masker material during the warm-up period; either 'none', 'positivePhase' (default), or 'negativePhase'
  - *initialSNR*: signal to noise ratio of the first presentation, dB
  - *fixedLevel*: level of either the target or the masked, whichever is fixed, dB~~A~~ SPL
  - *fixedMaterial*: determines whether the target or the masker is presented at the *fixedLevel*; the other is adjusted to achieve the desired SNR
  - *correctStep*: SNR step size for each correct digit in the previous response, using the *warmupMasker* masker type
  - *incorrectStep*: SNR step size for each incorrect digit in the previous response, using the *warmupMasker* masker type maxSNR: maximum decibels the target can exceed the masker
  - *ear*: the media is played in this channel only; either 'left', 'right', or 'both'
  - *maxLevel*: maximum output level, dB
- Submission Inputs:
  - this_exam.createSubmission(*nCorrect*)
    - *nCorrect*: number of digits correctly identified
- Results Properties:
  - *currentPresentation:* filename of the current target presentation
  - *currentDigits:* the correct response for the current target presentation

- *currentSNR:* the SNR of the current target presentation, in dB
- *presentationCount:* the current presentation number
- *currentMasker:* the masker of the current presentation; either 'positivePhase' or 'negativePhase'
- *State*: current state of the exam, one of the following:
  - *PLAYING*: the exam is playing
  - *WAITING_FOR_RESPONSE*: the CHA is waiting for an input submission
  - *DONE*: the exam has completed
  - *LOADING*: the exam is loading
- *digitScore*: percent of total number of digits correctly identified
- *presentationScore*: percent of presentations where all three digits were correctly identified
- *maskerLevel*: level of the masker, dB SPL
- *targetLevel*: level of the target, dB SPL
- *targetType*: the same as the *targetType* input parameter
- *warmupDigitScore*: percent of total number of digits correctly identified while the masker was the *warmupMasker*
- *warmupPresentationScore*: percent of presentations where all three digits were correctly identified while the masker was the *warmupMasker*
- *ear*: the same as the *ear* input parameter

- Example Scripts:
  - *ThreeDigit_example.m*

2.4 OTHER INCLUDED FUNCTIONS

The CHAMI SDK also includes two functions for performing what may be common tasks: changing the maximum MCL knob level setting and querying a probe for its RETSPL values. These are discussed below.

2.4.1 changeMaxMCL

During playback, the CHA corrects for the non-spectrally flat response of a playback device by filtering the data in real time using a 16-bit FIR filter. For playback at a fixed level, this filter is built to take full advantage of all 16 bits.

When the MCL is used for real-time level control (e.g., **setup_pr** with multitrack set to true), the filter is scaled relative to a filter built for the "maxMcl" setting to adjust the level of the signal.

Because the filter has only 16 bits of resolution, and some target material (e.g., speech) has a low RMS-to-peak ratio, playback at exceptionally low output levels (with the default "maxMcl" setting) may run into bit resolution issues, degrading the signal quality and calibration accuracy. This function can be used to quickly and easily change the maximum setting prior to testing. Likewise, testing at exceptionally high levels will require the maximum setting increased upwards prior to testing.

- Function Calls:
  - `changeMaxMCL(`*`cha_device, newMaxMCL`*`)`
- Inputs:
  - *`cha_device:`* open instance of class chami
  - *`newMaxMCL:`* the new maximum MCL setting (see Section 2.5 for a note on calibration)

2.4.2 get_RETSPL

Output devices used for audiometry have a parameter known as the RETSPL. The RETSPL is sound pressure level (SPL) measured on an artificial ear or in an ear simulator at the same input voltage level that corresponds to 0 dB HL for a "normal" human subject at a given frequency. As with other calibration-related data, the RETSPL for CHA output devices is stored on the probe-side connector. If building an audiometry exam from scratch (the **bekesylike_exam** and **hughsonwestlake_exam** classes already correct for this), it is necessary to know the RETSPL. This function provides an easy way for querying this information. The function uses nearest-neighbor interpolation for frequencies outside of the range stored on the probe; a warning is returned if this occurs.

- Function Calls:
  - *RETSPL* = get_RETSPL(*cha_device*, *frequency*)
- Inputs:
  - *cha_device:* open instance of class chami
  - *frequency*: array of frequencies to return the RETSPL at
- Outputs:
  - *RETSPL:* array of RETSPL values for the output device attached to the CHA at the frequencies specified by *frequency*

2.5 AN IMPORTANT NOTE ON CALIBRATED PLAYBACK

For low-level methods and classes (e.g., **setup_pr, playsound_exam**), data are streamed to the CHA for immediate playback. The CHA has no means of knowing what the RMS value of the entire stream will be. This RMS level is needed to determine the scaling for playback at the proper equivalent sound pressure level (LEQ). The CHA therefore assumes that the input stream has the same RMS level as a sine wave spanning values from -1.0 to 1.0—an RMS of 0.707. When configuring the playback level on the CHA, the user must account for this; in most cases the requested LEQ will have to be adjusted higher than the desired level:

```
requested_level = desired_Level + 20*log10(0.707/RMS_value)
```

Likewise, the level setting of the MCL knob returned by the CHA also needs to be corrected for this. In this case the MCL knob most likely will return a value greater than the actual value:

```
actual_level = MCL_setting + 20*log10(RMS_value/0.707)
```

Higher level methods (e.g., **play_wrt_reference, play_with_noise**) first process the data in MATLAB before passing it to the CHA and make the necessary corrections automatically.

3 APPENDIX A—CREARE WIRELESS AUDIOMETRIC HEADSET ATTENUATION