g.tec medical engineering GmbH
4521 Schiedlberg, Sierningstrasse 14, Austria
Tel.: (43)-7251-22240-0
Fax: (43)-7251-22240-39
office@gtec.at, http://www.gtec.at

g.NEEDaccess

NETWORK ENABLED EASY DATA ACCESS INTERFACE

MATLAB API User Manual

V1.16.00

CONTENT:

TO THE READER

Welcome to the medical and electrical engineering world of g.tec!
Discover the only professional biomedical signal processing platform under MATLAB and Simulink. Your ingenuity finds the appropriate tools in the g.tec elements and systems. Choose and combine flexibly the elements for biosignal amplification, signal processing and stimulation to perform even real-time feedback.

Our team is prepared to find the better solution for your needs.

Take advantage of our experience!

Dr. Christoph Guger Dr. Guenter Edlinger

Researcher and Developer

Reduce development time for sophisticated real-time applications from month to hours.
Integrate g.tec's open platform seamlessly into your processing system.
g.tec's rapid prototyping environment encourages your creativity.

Scientist

Open new research fields with amazing feedback experiments.
Process your EEG/ECG/EMG/EOG data with g.tec's biosignal analyzing tools.
Concentrate on your core problems when relying on g.tec's new software features like ICA, AAR or online Hjorth's source derivation.

Study design and data analysis

You are planning an experimental study in the field of brain or life sciences? We can offer consultation in experimental planning, hardware and software selection and can even do the measurements for you. If you have already collected EEG/ECG/EMG/EOG, g.tec can analyze the data starting from artifact control, do feature extraction and prepare the results ready for publication.

Related Products

g.tec provides several biosignal analysis elements that are especially relevant to the kinds of tasks you perform with g.NEEDaccess MATLAB API.

For more detailed information on any of our elements, up-dates or new extensions please visit our homepage www.gtec.at or just send us an email to office@gtec.at

Conventions

Item	Format	Example
MATLAB code	`Courier`	to start Simulink, type `simulink`
String variables	*Courier italics*	`set(P_C,'PropertyName',...µ)`
Menu items	**Boldface**	Select **Save** from the **File** menu

ATTENTION

NOTE

Installation and Configuration

This chapter includes the following sections:

Hardware and Software Requirements

Hardware Requirements

g.NEEDaccess MATLAB API requires a PC compatible desktop, notebook workstation running Microsoft Windows.

The table below lists optimal settings:

Hardware	Properties
CPU	Intel Core i5 working at 3 GHz or faster
Harddisk	256 GB SSD
RAM	8 GB
USB 2.0 high speed port	One free USB port (EHCI) for any g.tec device

Software Requirements

g.NEEDaccess MATLAB API requires the installation of g.NEEDaccess Server, MATLAB and Microsoft .NET Framework. Make sure that the MATLAB installation works correctly before installing the g.NEEDaccess MATLAB API software. Depending on your Windows operating system administrator rights might be necessary for the installation.

Software	Version
g.NEEDaccess Server	1.16.01
MATLAB	Release 2015a
Microsoft .NET Framework	4.6.1
Windows	Windows 10 Pro, English, 64-bit
Acrobat Reader	DC 2015

ATTENTION

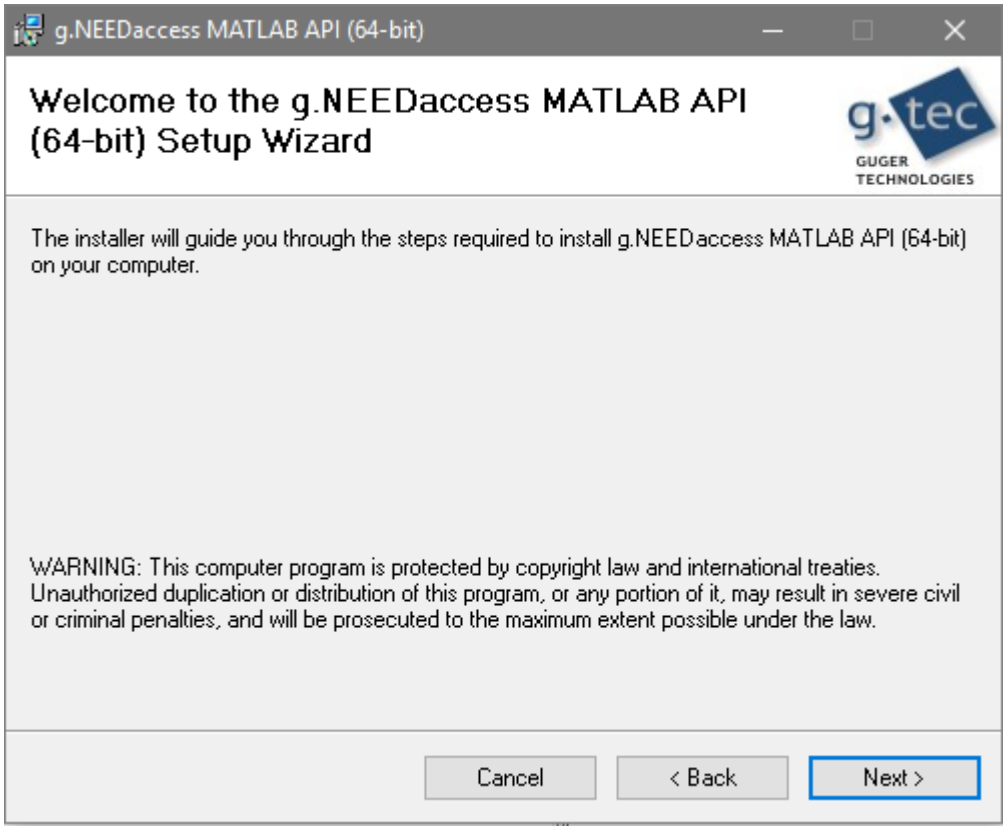
It is essential to use the correct Microsoft Windows, MATLAB and .NET Framework versions! The package is not tested with other Microsoft or MATLAB versions and might not work correctly if other versions are used.

Installation from a CD

The installation consists of three steps:

1. Installation of g.NEEDaccess MATLAB API

   Insert the g.tec product CD into the CD-drive and change to the `g.NEEDaccess MATLAB API` directory of your CD-drive and double-click on the `Setup.exe` file. The installation starts and displays the welcome message. Follow the instructions on the screen.



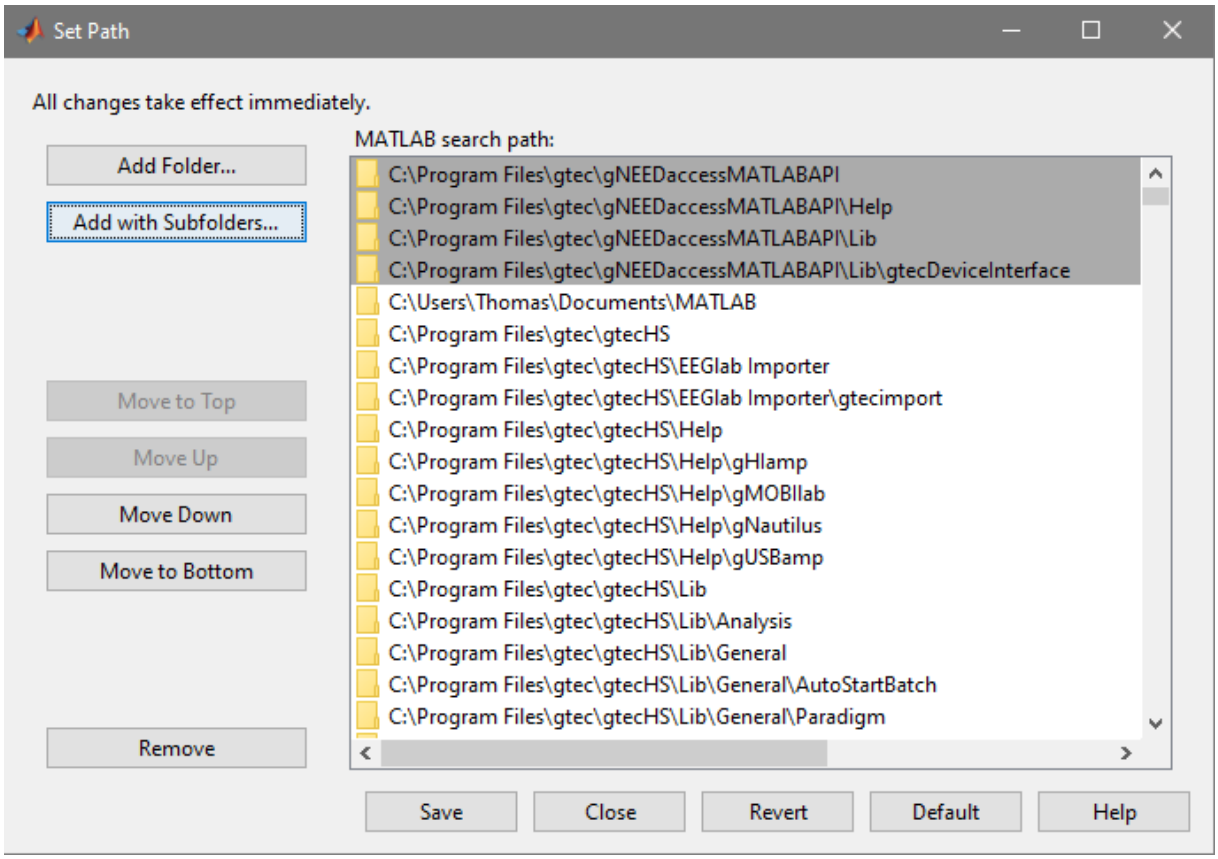
   Please read the License Agreement for g.NEEDaccess MATLAB API and if you agree with the terms, click **I Agree** and **Next**.

2. Set MATLAB path

   To make the path settings start MATLAB and open the **Set Path** window from the **File** menu. Then click on the **Add with Subfolders** button and select

   ```
   C:\Program Files\gtec\gNEEDaccessMATLABAPI
   ```

   to add all subdirectories:



   To add the folder containing the examples to the MATLAB paths, click **Add with Subfolders** and select

   ```
   C:\Users\<user_name>\Documents\gtec\gNEEDaccessMATLABAPI\Examples
   ```

   The g.NEEDaccess Server root installation folder has also to be added to the MATLAB paths. Click **Add Folder…** and browse to the folder

   ```
   C:\Program Files\gtec\gNEEDaccess
   ```

   and click **OK**.

   Click **Save** and **Close** to finish the installation.

Files on your Computer

g.NEEDaccess MATLAB API files - are stored under (it is assumed that the default path setting is used)

```
C:\Program Files\gtec\gNEEDaccessMATLABAPI
```

Example code MATLAB files - are stored in the subdirectory for each amplifier

```
C:\Users\<user_name>\Documents\gtec\gNEEDaccessMATLABAPI\Examples
```

Help files - are stored under

```
C:\Program Files\gtec\gNEEDaccessMATLABAPI\Help
```

g.tec Device Interface Class Properties and Methods

The g.tec device interface MATLAB class `gtecDeviceInterface` provides an interface to all applicable settings for g.NEEDaccess and the g.tec devices g.HIamp, g.Nautilus or g.USBamp in their corresponding device classes, as well as all methods provided by GDS to get the corresponding amplifier capabilities.

Common GDS Properties

`IPAddressHost`	Define IP address of server where the g.tec device is connected as string
`HostPort`	Port of host connection as integer
`IPAddressLocal`	Define IP address of local computer as string
`LocalPort`	Port of local connection as integer
`DeviceConfigurations`	Holding the device specific configurations for g.HIamp, g.Nautilus or g.USBamp.

Properties available during data acquisition

`SamplesAvailable`	Holds the number of samples currently available. Read only, valid only when data acquisition is running.
`isRunning`	Information that data acquisition is running. Read only, true when data is acquired, false otherwise.

Device specific classes and their properties

g.HIamp (gHIampDeviceConfiguration)

Amplifier Settings

`Name`	String holding the serial number of g.HIamp (e.g. `HA-2014.01.02`)
`DeviceType`	String representing the device type (G`HiAmp`) (predefined, must not be changed)
`SamplingRate`	Specify the sampling frequency of g.Hiamp in Hz as unsigned integer
`NumberOfScans`	Specify the buffering block size as unsigned short, possible values depend on sampling rate, use function GetSupportedSamplingRates to get recommended values.
`Channels`	Array of g.HIamp channel configurations (`gHIampChannels`) holding properties for each analog channel

Options

`CounterEnabled`	Show a counter on first recorded channel which is incremented with every block transmitted to the PC. Overruns at 1000000.
`TriggerLinesEnabled`	Scan the digital trigger channel with the analog inputs
`HoldEnabled`	enable signal hold
`InternalSignalGenerator`	Holds a `gHIampInternalSignalGenerator` object to enable and configure internal signal generator of g.HIamp

Internal Signal Generator (`gHIampInternalSignalGenerator`)

`Enabled`	Apply internal test signal to all inputs (requires shortcut of all analog channels to ground)
`Amplitude`	Amplitude of the test signal (7.62283 mV fix)
`Offset`	Offset of the test signal (-7.62283 mV fix)
`Frequency`	Specify the frequency of the test signal

Channel Settings (`gHIampChannels`)

`ChannelNumber`	Unsigned integer holding the channel number of the analog channel
`Available`	Bool value indicating availability of an analog channel (read only)
`Acquire`	Bool value selecting the channel for data acquisition
`BandpassFilterIndex`	Perform a digital bandpass filtering of the input channels. Use GetAvailableFilters to get filter indices.
`NotchFilterIndex`	Perform a bandstop filtering to suppress the power line frequency of 50 Hz or 60 Hz. Use GetAvailableFilters to get filter indices.
`ReferenceChannel`	Select a channel number as reference channel for an analog channel

g.Nautilus (gNautilusDeviceConfiguration)

Amplifier Settings

`Name`	String holding the serial number of g.Nautilus (e.g. `NA-2014.07.67`)
`DeviceType`	String representing the device type (`GNautilus`)
`SamplingRate`	Specify the sampling frequency of g.Nautilus in Hz as unsigned integer
`NumberOfScans`	Specify the buffering block size as unsigned short, possible values depend on sampling rate, use function GetSupportedSamplingRates to get recommended values.
`InputSignal`	holding type of input signal, can be 1 (Electrode), 2 (Shortcut) or 6 (TestSignal)
`Channels`	Array of g.Nautilus channel configurations (`gNautilusChannels`) holding properties for each analog channel

Options

`NoiseReduction`	Bool value enabling noise reduction for g.Nautilus
`CAR`	Bool value enabling common average calculation for g.Nautilus
`AccelerationData`	Bool value enabling acquisition of acceleration data from g.Nautilus head stage, adds 3 additional channels to the data acquisition for x, y, and z direction
`Counter`	show a counter as an additional channel
`LinkQualityInformation`	Bool value enabling additional channel informing about link quality between head stage and base station
`BatteryLevel`	Bool to enable acquisition of additional channel holding information about remaining battery capacity
`DigitalIOs`	scan the digital channels with the analog inputs and add it as additional channel acquired
`ValidationIndicator`	Enables the additional channel validation indicator, informing about the liability of the data recorded
`NetworkChannel`	Unsigned integer value representing the network channel used between head stage and base station

Specify Channel Settings (`gNautilusChannels`)

`ChannelNumber`	Unsigned integer holding the channel number of the analog channel
`Available`	Bool value indicating availability of an analog channel
`Acquire`	Bool value selecting the channel for data acquisition
`Sensitivity`	Double value representing the sensitivity of the specified channel
`UsedForNoiseReduction`	Bool value indicating if channel should be used for noise reduction
`UsedForCAR`	Bool value indicating if channel should be used for common average calculation
`BandpassFilterIndex`	Perform a digital bandpass filtering of the input channels. Use GetAvailableFilters to get filter indices.
`NotchFilterIndex`	Perform a bandstop filtering to suppress the power line frequency of 50 Hz or 60 Hz. Use GetAvailableFilters to get filter indices.
`BipolarChannel`	Select a zero based channel index as reference channel for an analog channel

g.USBamp (gUSBampDeviceConfiguration)

Amplifier Settings

`Name`	String holding the serial number of g.USBamp (e.g. `UB-2014.05.23`)
`DeviceType`	String representing the device type (`GUsbAmp`)
`SamplingRate`	Specify the sampling frequency of g.HIamp in Hz as unsigned integer
`NumberOfScans`	Specify the buffering block size as unsigned short, possible values depend on sampling rate, use function GetSupportedSamplingRates to get recommended values.
`CommonGround`	Array of 4 bool elements to enable or disable common ground
`CommonReference`	Array of 4 bool values to enable or disable common reference
`Channels`	Array of g.USBamp channel configurations (`gUSBampChannels`) holding properties for each analog channel

Options

`ShortCutEnabled`	Bool enabling or disabling g.USBamp shortcut
`CounterEnabled`	Show a counter on first recorded channel which is incremented with every block transmitted to the PC. Overruns at 1000000.
`TriggerEnabled`	scan the digital trigger channel with the analog inputs
`InternalSignalGenerator`	Holds a `gUSBampInternalSignalGenerator` object to enable and configure internal signal generator of g.USBamp

Internal Signal Generator (`gUSBampInternalSignalGenerator`)

`Enabled`	Apply internal test signal to all inputs
`WaveShape`	Unsigned integer representing the wave shape of the internal test signal. Can be 1 (square), 2 (saw tooth), 3 (sine) 4 (DRL) or 5 (noise)
`Amplitude`	the amplitude of the test signal (can be -250 to 250 mV)
`Offset`	the offset of the test signal (can be -200 to 200 mV)
`Frequency`	specify the frequency of the test signal (can be 1 to 100 Hz)

Specify Channel Settings (`gUSBampChannels`)

`ChannelNumber`	Unsigned integer holding the channel number of the analog channel
`Available`	Bool value indicating availability of an analog channel (read only)
`Acquire`	Bool value selecting the channel for data acquisition
`BandpassFilterIndex`	Perform a digital bandpass filtering of the input channels. Use GetAvailableFilters to get filter indices.
`NotchFilterIndex`	Perform a bandstop filtering to suppress the power line frequency of 50 Hz or 60 Hz. Use GetAvailableFilters to get filter indices.
`BipolarChannel`	Select a channel number as reference channel for an analog channel

Supported sampling rates and corresponding recommended number of scans

For the relation between number of scans and sampling rate It is generally recommended to have a ratio of minimum 30 ms (see formula below)

$$30\ ms\ \leq \frac{\text{Number of Scans}}{\text{Sampling Rate}} * 1000$$

For the standard recommended settings see table below.

g.HIamp		g.USBamp		g.Nautilus	
Sampling Rate [Hz]	Recommended Number of Scans	Sampling Rate [Hz]	Recommended Number of Scans	Sampling Rate [Hz]	Recommended Number of Scans
256	8	32	1	250	8
512	16	64	2	500	15
600	32	128	4		
1200	64	256	8		
2400	128	512	16		
4800	256	600	32		
9600	256	1200	64		
19200	256	2400	128		
38400	256	4800	256		
		9600	512		
		19200	512		
		38400	512		

Settings below 30 ms are likely to lose data during data acquisition.

Get devices currently connected to the computer

This function returns all g.tec devices available at a defined endpoint (IP address and port). Host and local IP and corresponding ports are required to use this function (see common GDS properties section).

GetConnectedDevices

Gets a structure with all devices currently connected to the computer at the defined IP address.

Call with:

```
connected_devices = gds_interface.GetConnectedDevices();
```

Type	Name	Description
Input	No input	
Output	`connected_devices`	Structure with one element for each g.tec amplifier found containing three fields each
Output fields	`DeviceType`	String with type of connected device (`GNautilus`, `GHiAmp` or `GUsbAmp`)
	`Name`	String with serial number of connected device (e.g. `NA-2014.07.67`)
	`InUse`	Bool value indicating if device connected is currently used for a data acquisition

Measure Impedance

g.NEEDaccess allows the measurement of channel impedances for g.Nautilus, g.HIamp and g.USBamp.

GetImpedance

Returns the impedances for a selected range of channels, for a single channel or for all channels available for the selected device is no input is provided. If g.USBamp is connected, the serial number of the device has to be provided.

g.HIamp distinguishes between active and passive electrodes, it is not possible to measure both though (will throw an exception in this case). If g.HIamp is used with impedance measurement, please use an optional string input either with the string active or passive. See calling example below

For g.USBamp call with:

```
impedances = gds_interface.GetImpedance(gusbamp_config.Name,
channels_selected);
```

For g.HIamp call for passive electrodes with
```
impedances = gds_interface.GetImpedance(channels_selected, 'passive');
```
Or for active electrodes with
```
impedances = gds_interface.GetImpedance(channels_selected, 'active');
```

For g.Nautilus and g.HIamp call with:

```
impedances = gds_interface.GetImpedance(channels_selected);
```

Type	Name	Description
Input	`gusbamp_config.Name`	Serial number of g.USBamp, only valid if g.USBamp is used.
	`channels_selected`	Array with 1 x m elements containing the channels whose impedance should be measured. Optional, impedances for all channels of the connected device are returned if unused.
		String input with `'active'` or `'passive'`, depending on the electrode type used.
Output	`impedances`	Array with 2 x m elements containing the channel number and the corresponding impedance value

ATTENTION

Function GetImpedance might throw error message when no impedance can be measured for the device currently connected, mainly when ground and/or reference electrodes are not connected.

Common Methods to get device capabilities

All methods require a device configuration holding at least the device serial (`Name` property in `gNautilusDeviceConfiguration`, `gHIampDeviceConfiguration` or `gUSBampDeviceConfiguration`)

Note: If g.USBamp is used and multiple amplifiers are synchronized, each of the functions in this section requires the serial number of the g.USBamp to return the property for this device. Each function will fail otherwise.

GetAvailableChannels

Returns a bool array with as many elements as the maximum possible channel number of a g.tec device (64 channels for g.Nautilus, 256 channels for g.HIamp or 16 channels for g.USBamp), where each element having the value true (1) represents an available analog channel of the device currently connected. False (0) represents an unavailable channel.

For g.USBamp call with:

```
available_channels =
gds_interface.GetAvailableChannels(gusbamp_config.Name);
```

For g.Nautilus and g.HIamp call with:

```
available_channels = gds_interface.GetAvailableChannels();
```

Type	Name	Description
Input	`gusbamp_config.Name`	Serial number of g.USBamp, only valid if g.USBamp is used.
Output	`available_channels`	Array of bool with 16 (g.USBamp), 64 (g.Nautilus) or 256 (g.HIamp) elements, where true represents an hardware available channel, false otherwise.

GetAvailableFilters

Gets a structure containing all bandpass and notch filters available for the current device.

For g.USBamp call with:

```
available_filterss = gds_interface.GetAvailableFilters(gusbamp_config.Name,
sampling_rate);
```

For g.Nautilus and g.HIamp call with:

```
available_filters = gds_interface.GetAvailableFilters(sampling_rate);
```

Type	Name	Description
Input	`gusbamp_config.Name`	Serial number of g.USBamp, only valid if g.USBamp is used.
	`sampling_rate`	Sampling rate to return filters for, optional
Output	`available_filters`	Structure with two sub-structures, one for available bandpass filters and one for available notch filters.
Output fields	`BandpassFilters`	Structure with one element for each bandpass filter available containing five fields each
	`NotchFilters`	Structure with one element for each notch filter available containing five fields each
	`FilterIndex`	Integer value holding the index for the device setting in the analog channel configuration of an amplifier either BandpassFilterIndex or NotchFilterIndex (please see gNautilusChannels, gHIampChannels or gUSBampChannels definition for further information)
	`SamplingRate`	Double value holding the sampling rate for which the filter is valid
	`Order`	Unsigned integer holding filter order
	`LowerCutOffFrequency`	Double representing lower cutoff frequency of the filter
	`UpperCutOffFrequency`	Double representing upper cutoff frequency of the filter
	`TypeId`	String representing type of filter

GetSupportedSamplingRates

Returns an array of unsigned integers holding the sampling rates supported by the connected device.

For g.USBamp call with:

```
supported_fs =
gds_interface.GetSupportedSamplingRates(gusbamp_config.Name);
```

For g.Nautilus and g.HIamp call with:

```
supported_fs = gds_interface.GetSupportedSamplingRates();
```

Type	Name	Description
Input	`gusbamp_config.Name`	Serial number of g.USBamp, only valid if g.USBamp is used.
Output	`supported_fs`	Array of unsigned integers with the dimensions 2 x m, where one represents the available sampling rates and the other one the corresponding recommended number of scans.

GetDeviceInfo

Return a string with the complete device information of the device currently connected.

For g.USBamp call with:

```
device_info = gds_interface.GetDeviceInfo(gusbamp_config.Name);
```

For g.Nautilus and g.HIamp call with:

```
device_info = gds_interface.GetDeviceInfo();
```

Type	Name	Description
Input	`gusbamp_config.Name`	Serial number of g.USBamp, only valid if g.USBamp is used.
Output	`device_info`	String holding the complete hardware information of the connected device.

GetScaling

This function gets a structure with arrays for scaling factor and offset for each analog channel of the connected device.

For g.USBamp call with:

```
device_scaling = gds_interface.GetScaling(gusbamp_config.Name);
```

For g.Nautilus and g.HIamp call with:

```
device_scaling = gds_interface.GetScaling();
```

Type	Name	Description
Input	`gusbamp_config.Name`	Serial number of g.USBamp, only valid if g.USBamp is used.
Output	`device_scaling`	Structure with two elements, scaling factor and offset
Output fields	`Factor`	Array holding single type values with scaling factor for each analog channel.
	`Offset`	Array holding single type values with offset for each analog channel.

Device specific methods

g.Nautilus methods

GetSupportedSensitivities

Retrieves the supported sensitivity values for a g.Nautilus device.

Call with:

```
supported_sensitivities = gds_interface.GetSupportedSensitivities();
```

Type	Name	Description
Input	No Input	-
Output	`supported_ sensitivities`	Array of double values each one holding a supported sensitivity value for g.Nautilus currently connected.

GetSupportedInputSources

Retrieves the available input sources a g.Nautilus device can use to acquire signals.

Call with:

```
supported_input_sources = gds_interface.GetSupportedInputSources();
```

Type	Name	Description
Input	No input	-
Output	`supported_ input_sources`	Structure holding name and value for each input source available.
Output fields	`Name`	String describing the input source (e.g. `Electrode` for the EEG input of g.Nautilus)
	`Value`	Unsigned integer holding the number of the corresponding input source (e.g. `1` for `Electrode`)

GetChannelNames

Gets the names of all available channels depending on the mounted grid.

Call with:

```
channel_names = gds_interface.GetChannelNames();
```

Type	Name	Description
Input	No input	-
Output	`channel_names`	Cell array holding an element with a channel name for each analog channel available.

GetSupportedNetworkChannels

Retrieves the available network channels a g.Nautilus device can use to pair its headset and base station.

Call with:

```
supported_network_channels = gds_interface.GetSupportedNetworkChannels();
```

Type	Name	Description
Input	No input	-
Output	`supported_network_channels`	Array of unsigned integer values, each one representing a possible network channel to be used for g.Nautilus.

GetNetworkChannel

Retrieves the radio channel currently used for wireless communication between headset and base station.

Call with:

```
network_channel = gds_interface.GetNetworkChannel();
```

Type	Name	Description
Input	No input	-
Output	`network_channel`	Network channel currently used for communication between g.Nautilus base station and headset.

SetNetworkChannel

Sets the radio channel used for wireless communication between the headset and base station.

Call with:

```
gds_interface.SetNetworkChannel(network_channel);
```

Type	Name	Description
Input	`network_channel`	Network channel to be used for communication. Use `GetSupportedNetworkChannels` to retrieve possible network channels.
Output	No output	-

GetNumberOfMountedModules

Gets the number of the actually mounted modules of the currently connected g.Nautilus device.

Call with:

```
mounted_modules = gds_interface.GetNumberOfMountedModules();
```

Type	Name	Description
Input	No input	-
Output	`mounted_modules`	Number of mounted modules on g.Nautilus headstage (corresponds to the number of channels the device offers).

GetAvailableDIOs

Gets information about digital I/O channels available of the specified g.Nautilus device.

Call with:

```
available_dios = gds_interface.GetAvailableDIOs();
```

Type	Name	Description
Input	No input	-
Output	`available_dios`	Structure containing one element for each digital IO available.
Output fields	`ChannelNumber`	Unsigned integer representing the digital IO number.
	`Direction`	String representing the direction of the digital channel (`In` or `Out`)

g.USBamp Methods

GetAsyncDigitalIOs

Use this function to get the asynchronous digital in- or outputs available for g.USBamp currently used for the data acquisition session. This function requires one input argument and has one output argument.

Call with:

```
async_dios = gds_interface.GetAsyncDigitalIOs(gusbamp_config.Name);
```

Type	Name	Description
Input	`gusbamp_config.Name`	Serial number of g.USBamp
Output	`async_dios`	Structure with as many elements as asynchronous digital IOs are available, each elements contains three fields
Output fields	`ChannelNumber`	Integer value representing the digital channel number
	`Direction`	String holding the digital channel direction (`In` or `Out`)
	`Value`	Current value of the digital channel (`true` or `false`).

SetAsyncDigitalOutputs

Use this function to set the asynchronous digital outputs available for the g.USBamp currently used for the data acquisition session. This function requires two input arguments and has no output argument.

Call with:

```
gds_interface.SetAsyncDigitalIOs(gusbamp_config.Name, async_dios);
```

Type	Name	Description
Input	`gusbamp_config.Name`	Serial number of g.USBamp
	`async_dios`	Structure with as many elements as asynchronous digital IOs are available, each elements contains three fields
Input fields	`ChannelNumber`	Integer value representing the digital channel number
	`Direction`	String holding the digital channel direction (`In` or `Out`)
	`Value`	Value to be applied for the digital channel (`true` or `false`).
Output	-	-

`Value`: Value to be applied for the digital channel to set (`true` or `false`).

Methods for data acquisition

SetConfiguration

Sets the configuration currently available in the property `DeviceConfigurations` of `gtecDeviceInterface`.

Input: none.

Output: none.

StartDataAcquisition

Starts the data acquisition with the settings provided.

Input: none.

Output: none.

StopDataAcquisition

Stops the data acquisition currently running and closes the connection.

Input: none.

Output: none.

GetData

Retrieves data during the data acquisition is running.

Call with:

```
[scans_received, data] = gds_interface.GetData(number_of_scans);
```

Type	Name	Description
Input	`number_of_scans`	Can either be empty to retrieve the number of scans currently available (0 as input does the same) or if an input is provided should be defined as equal to or a multiple of the number of scans in the device configuration (`NumberOfScans` property in `gNautilusDeviceInterface`, `gHIampDeviceInterface` or `gUSBampDeviceInterface`). Maximum value for this input is the sampling rate selected for the current data acquisition.
Output	`scans_received`	Actual amount of acquired scans
	`data`	Acquired data in the form scans x channels, data type returned is single

NOTE

All biosignal data returned by the GetData function is in µV.

ATTENTION

For g.HIamp and g.USBamp it is strongly recommended to use 0 or an empty input for the GetData function, to return all scans currently available (especially for sampling rates above 2400 Hz). The first value returned by GetData will hold the number of actually returned scans.

ATTENTION

It is strongly advised to use the values for number of scans recommended by GetSupportedSamplingRates for the device configuration and the same value or multiples of it for GetData function to avoid data loss.

Quick Start

g.Nautilus

NOTE: The example contained in the g.Nautilus examples installation folder (default `C:\Users\<user_name>\Documents\gtec\gNEEDaccessMATLABAPI\Examples\gNauailus`) is using a 32 channel g.Nautilus device. If another device is used (8, 16 or 64 channels), the code has to adapted correspondingly (see comments in the example code).

To test g.NEEDaccess MATLAB API with g.Nautilus on your system, please perform the following example:

1. Create a `gtecDeviceInterface` object with

   ```
   gds_interface = gtecDeviceInterface;
   ```

2. Set the IP and ports for the host and the client. If used on a local computer, set the following parameters

   ```
   gds_interface.IPAddressHost = '127.0.0.1';
   gds_interface.IPAddressLocal = '127.0.0.1';
   gds_interface.HostPort = 50223;
   gds_interface.LocalPort = 50224;
   ```

3. Get the device(s) currently connected

   ```
   connected_devices = GetConnectedDevices();
   ```

4. Use `gNautilusDeviceConfiguration` to create a g.Nautilus device configuration and, as there should only be one g.Nautilus device available, set the device serial as name

   ```
   gnautilus_config = gNautilusDeviceConfiguration();
   gnautilus_config.Name = connected_devices(1,1).Name;
   ```

5. To use functions requiring a device serial later, set this newly created device configuration to the `gtecDeviceInterface` object

   ```
   gds_interface.DeviceConfigurations = gnautilus_config;
   ```

6. Get available channels, supported sensitivities and supported input sources

   ```
   available_channels = gds_interface.GetAvailableChannels();
   supported_sensitivities = gds_interface.GetSupportedSensitivities();
   supported_input_sources = gds_interface.GetSupportedInputSources();
   ```

7. Set the sampling rate to `250` Hz and number of scans to `4`

```
gnautilus_config.SamplingRate = 250;
gnautilus_config.NumberOfScans = 4;
```

8. Set input source to record the internal test signal of g.Nautilus, a square wave applied to all analog channels with a frequency of 10 Hz and an amplitude of 15 mV

```
gnautilus_config.InputSource = supported_input_source(3).Value;
```

9. Enable additional channels counter and validation indicator

```
gnautilus_config.Counter = true;
gnautilus_config.ValidationIndicator = true;
```

10. Record all channels available

```
for i=1:size(gNautilusConfig.Channels,2)
    if (available_channels(1,i))
        gnautilus_config.Channels(1,i).Available = true;
        gnautilus_config.Channels(1,i).Acquire = true;
        % set sensitivity to 187.5 mV
        gnautilus_config.Channels(1,i).Sensitivity =...
            supported_sensitivities(6);
        % do not use channel for CAR and noise reduction
        gnautilus_config.Channels(1,i).UsedForNoiseReduction = false;
        gnautilus_config.Channels(1,i).UsedForCAR = false;
        % do not use filters
        gnautilus_config.Channels(1,i).BandpassFilterIndex = -1;
        gnautilus_config.Channels(1,i).NotchFilterIndex = -1;
        % do not use a bipolar channel
        gnautilus_config.Channels(1,i).BipolarChannel = -1;
    end
end
```

11. Set device configuration to `gtecDeviceInterface`

```
gds_interface.DeviceConfigurations = gnautilus_config;
```

12. Configure g.Nautilus with the current settings

```
gds_interface.SetConfiguration();
```

13. Start data acquisition and streaming from server

```
gds_interface.StartDataAcquisition();
```

14. Get data acquired from g.Nautilus, 2500 samples for 10 seconds of data

```
samples_acquired = 0;
while (samples_acquired < 2500)
    [scans_received, data] = gds_interface.GetData(4);
    samples_acquired = samples_acquired + scans_received;
end
```

15. End streaming and stop data acquisition and delete and clear `gtecDeviceInterface`

```
gds_interface.StopDataAcquisition();
delete(gds_interface);
clear gds_interface;
```

This example is also available as MATLAB script `gNautilusAPIDemo1.m` in the folder

`C:\Users\<user_name>\Documents\gtec\gNEEDaccessMATLABAPI\Examples\gNautilus`

g.HIamp

To test g.NEEDaccess MATLAB API with g.HIamp, please perform the following example:

1. Create a `gtecDeviceInterface` object with

   ```
   gds_interface = gtecDeviceInterface;
   ```

2. Set the IP and ports for the host and the client. If used on a local computer, set the following parameters

   ```
   gds_interface.IPAddressHost = '127.0.0.1';
   gds_interface.IPAddressLocal = '127.0.0.1';
   gds_interface.HostPort = 50223;
   gds_interface.LocalPort = 50224;
   ```

3. Get the device(s) currently connected

   ```
   connected_devices = GetConnectedDevices();
   ```

4. Use `gHIampDeviceConfiguration` to create a g.HIamp device configuration and, as there should only be one g.HIamp device available, set the device serial as name

   ```
   ghiamp_config = gHIampDeviceConfiguration();
   ghiamp_config.Name = connected_devices(1,1).Name;
   ```

5. To use functions requiring a device serial later, set this newly created device configuration to the `gtecDeviceInterface` object

   ```
   gds_interface.DeviceConfigurations = ghiamp_config;
   ```

6. Get available channels

   ```
   available_channels = gds_interface.GetAvailableChannels();
   ```

7. Set the sampling rate to `256` Hz and number of scans to `4`

   ```
   ghiamp_config.SamplingRate = 256;
   ghiamp_config.NumberOfScans = 4;
   ```

8. Enable counter

   ```
   ghiamp_config.CounterEnabled = true;
   ```

9. Record all channels available

```
for i=1:size(ghiamp_config.Channels,2)
    if (available_channels(1,i))
        ghiamp_config.Channels(1,i).Available = true;
        ghiamp_config.Channels(1,i).Acquire = true;
        % do not use filters
        ghiamp_config.Channels(1,i).BandpassFilterIndex = -1;
        ghiamp_config.Channels(1,i).NotchFilterIndex = -1;
        % do not use a bipolar channel
        ghiamp_config.Channels(1,i).ReferenceChannel = 0;
    end
end
```

10. Set device configuration to `gtecDeviceInterface`

```
gds_interface.DeviceConfigurations = ghiamp_config;
```

11. Configure g.HIamp with the current settings

```
gds_interface.SetConfiguration();
```

12. Start data acquisition and streaming from server

```
gds_interface.StartDataAcquisition();
```

13. Get data acquired from g.HIamp, 2560 samples for 10 seconds of data

```
samples_acquired = 0;
while (samples_acquired < 2560)
    [scans_received, data] = gds_interface.GetData(0);
    samples_acquired = samples_acquired + scans_received;
end
```

`scans_received` holds the number of actual scans returned by **GetData**

14. End streaming and stop data acquisition and delete and clear `gtecDeviceInterface`

```
gds_interface.StopDataAcquisition();
delete(gds_interface);
clear gds_interface;
```

This example is also available as MATLAB script `gHIampAPIDemo1.m` in the folder

`C:\Users\<user_name>\Documents\gtec\gNEEDaccessMATLABAPI\Examples\gHIamp`

g.USBamp

To test g.NEEDaccess MATLAB API with g.USBamp, please perform the following example:

1. Create a `gtecDeviceInterface` object with

   ```
   gds_interface = gtecDeviceInterface;
   ```

2. Set the IP and ports for the host and the client. If used on a local computer, set the following parameters

   ```
   gds_interface.IPAddressHost = '127.0.0.1';
   gds_interface.IPAddressLocal = '127.0.0.1';
   gds_interface.HostPort = 50223;
   gds_interface.LocalPort = 50224;
   ```

3. Get the device(s) currently connected

   ```
   connected_devices = GetConnectedDevices();
   ```

4. Use `gUSBampDeviceConfiguration` to create a g.USBamp device configuration and, as there should only be one g.USBamp device available, set the device serial as name

   ```
   gusbamp_config = gUSBampDeviceConfiguration();
   gusbamp_config.Name = connected_devices(1,1).Name;
   ```

5. To use functions requiring a device serial later, set this newly created device configuration to the `gtecDeviceInterface` object

   ```
   gds_interface.DeviceConfigurations = ghiamp_config;
   ```

6. Get available channels

   ```
   available_channels = gds_interface.GetAvailableChannels();
   ```

7. Set the sampling rate to `256` Hz and number of scans to `8`

   ```
   gusbamp_config.SamplingRate = 256;
   gusbamp_config.NumberOfScans = 8;
   ```

8. Enable and configure internal signal generator to acquire g.USBamps internal test signal

   ```
   gusbamp_siggen = gUSBampInternalSignalGenerator();
   gusbamp_siggen.Enabled = true;
   gusbamp_siggen.Frequency = 10;
   gusbamp_siggen.WaveShape = 3;
   gusbamp_siggen.Amplitude = 200;
   gusbamp_siggen.Offset = 0;
   gusbamp_config.InternalSignalGenerator = gusbamp_siggen;
   ```

9. Record all channels available

```
for i=1:size(gusbamp_config.Channels,2)
    if (available_channels(1,i))
        gusbamp_config.Channels(1,i).Available = true;
        gusbamp_config.Channels(1,i).Acquire = true;
        % do not use filters
        gusbamp_config.Channels(1,i).BandpassFilterIndex = -1;
        gusbamp_config.Channels(1,i).NotchFilterIndex = -1;
        % do not use a bipolar channel
        gusbamp_config.Channels(1,i).BipolarChannel = 0;
    end
end
```

10. Set device configuration to `gtecDeviceInterface`

```
gds_interface.DeviceConfiguration = gusbamp_config;
```

11. Configure g.HIamp with the current settings

```
gds_interface.SetConfiguration();
```

12. Start data acquisition and streaming from server

```
gds_interface.StartDataAcquisition();
```

13. Get data acquired from g.USBamp, `2560` samples for 10 seconds of data

```
samples_acquired = 0;
while (samples_acquired < 2560)
    [scans_received, data] = gds_interface.GetData(0);
    samples_acquired = samples_acquired + scans_received;
end
```

`scans_received` holds the number of actual scans returned by **GetData**

14. End streaming and stop data acquisition and delete and clear `gtecDeviceInterface`

```
gds_interface.StopDataAcquisition();
delete(gds_interface);
clear gds_interface;
```

This example is also available as MATLAB script `gUSBampAPIDemo1.m` in the folder

`C:\Users\<user_name>\Documents\gtec\gNEEDaccessMATLABAPI\Examples\gUSBamp`

Digital Inputs

g.Nautilus

NOTE: The example contained in the g.Nautilus examples installation folder (default `C:\Users\<user_name>\Documents\gtec\gNEEDaccessMATLABAPI\Examples\gNauailus`) is using a 32 channel g.Nautilus device. If another device is used (8, 16 or 64 channels), the code has to adapted correspondingly (see comments in the example code).

When enabled, digital inputs of g.Nautilus are recorded as additional channel ranging from 0 to 255. To record analog channels and digital inputs of g.Nautilus please perform the following steps:

1. Create a `gtecDeviceInterface` object with

   ```
   gds_interface = gtecDeviceInterface;
   ```

2. Set the IP and ports for the host and the client. If used on a local computer, set the following parameters

   ```
   gds_interface.IPAddressHost = '127.0.0.1';
   gds_interface.IPAddressLocal = '127.0.0.1';
   gds_interface.HostPort = 50223;
   gds_interface.LocalPort = 50224;
   ```

3. Get the device(s) currently connected

   ```
   connected_devices = GetConnectedDevices();
   ```

4. Use `gNautilusDeviceConfiguration` to create a g.Nautilus device configuration and, as there should only be one g.Nautilus device available, set the device serial as name

   ```
   gnautilus_config = gNautilusDeviceConfiguration();
   gnautilus_config.Name = connected_devices(1,1).Name;
   ```

5. To use functions requiring a device serial later, set this newly created device configuration to the `gtecDeviceInterface` object

   ```
   gds_interface.DeviceConfigurations = gnautilus_config;
   ```

6. Get available channels, supported sensitivities and supported input sources

   ```
   available_channels = gds_interface.GetAvailableChannels();
   supported_sensitivities = gds_interface.GetSupportedSensitivities();
   supported_input_sources = gds_interface.GetSupportedInputSources();
   ```

7. Set the sampling rate to `250` Hz and number of scans to `4`

```
gnautilus_config.SamplingRate = 250;
gnautilus_config.NumberOfScans = 4;
```

8. Set input source to record the internal test signal of g.Nautilus, a square wave applied to all analog channels with a frequency of `10` Hz and an amplitude of `15` mV

```
gnautilus_config.InputSource = supported_input_source(3).Value;
```

9. Enable digital inputs for recording

```
gnautilus_config.DigitalIOs = true;
```

10. Enable additional channels counter and validation indicator

```
gnautilus_config.Counter = true;
gnautilus_config.ValidationIndicator = true;
```

11. Record all channels available

```
for i=1:size(gNautilusConfig.Channels,2)
    if (available_channels(1,i))
        gnautilus_config.Channels(1,i).Available = true;
        gnautilus_config.Channels(1,i).Acquire = true;
        % set sensitivity to 187.5 mV
        gnautilus_config.Channels(1,i).Sensitivity = ...
              supported_sensitivities(6);
        % do not use channel for CAR and noise reduction
        gnautilus_config.Channels(1,i).UsedForNoiseReduction = false;
        gnautilus_config.Channels(1,i).UsedForCAR = false;
        % do not use filters
        gnautilus_config.Channels(1,i).BandpassFilterIndex = -1;
        gnautilus_config.Channels(1,i).NotchFilterIndex = -1;
        % do not use a bipolar channel
        gnautilus_config.Channels(1,i).BipolarChannel = -1;
    end
end
```

12. Set device configuration to `gtecDeviceInterface`

```
gds_interface.DeviceConfigurations = gnautilus_config;
```

13. Configure g.Nautilus with the current settings

```
gds_interface.SetConfiguration();
```

14. Start data acquisition and streaming from server

```
gds_interface.StartDataAcquisition();
```

15. Get data acquired from g.Nautilus, `2500` samples for 10 seconds of data

```
samples_acquired = 0;
while (samples_acquired < 2500)
    [scans_received, data] = gds_interface.GetData(4);
    samples_acquired = samples_acquired + scans_received;
end
```

16. End streaming and stop data acquisition and delete and clear `gtecDeviceInterface`

```
gds_interface.StopDataAcquisition();
delete(gds_interface);
clear gds_interface;
```

This example is also available as MATLAB script `gNautilusAPIDIDemo.m` in the folder

`C:\Users\<user_name>\Documents\gtec\gNEEDaccessMATLABAPI\Examples\gNautilus`

g.HIamp

When enabled, digital inputs of g.HIamp are recorded as additional channel ranging from 0 to 65535. To record analog channels and digital inputs of g.HIamp please perform the following steps:

1. Create a `gtecDeviceInterface` object with

   ```
   gds_interface = gtecDeviceInterface;
   ```

2. Set the IP and ports for the host and the client. If used on a local computer, set the following parameters

   ```
   gds_interface.IPAddressHost = '127.0.0.1';
   gds_interface.IPAddressLocal = '127.0.0.1';
   gds_interface.HostPort = 50223;
   gds_interface.LocalPort = 50224;
   ```

3. Get the device(s) currently connected

   ```
   connected_devices = GetConnectedDevices();
   ```

4. Use `gHIampDeviceConfiguration` to create a g.HIamp device configuration and, as there should only be one g.HIamp device available, set the device serial as name

   ```
   ghiamp_config = gHIampDeviceConfiguration();
   ghiamp_config.Name = connected_devices(1,1).Name;
   ```

5. To use functions requiring a device serial later, set this newly created device configuration to the `gtecDeviceInterface` object

   ```
   gds_interface.DeviceConfigurations = ghiamp_config;
   ```

6. Get available channels, supported sensitivities and supported input sources

   ```
   available_channels = gds_interface.GetAvailableChannels();
   ```

7. Set the sampling rate to `256` Hz and number of scans to `4`

   ```
   ghiamp_config.SamplingRate = 256;
   ghiamp_config.NumberOfScans = 4;
   ```

8. Enable digital inputs for recording

   ```
   ghiamp_config.TriggerLinesEnabled = true;
   ```

9. Enable counter

   ```
   ghiamp_config.CounterEnabled = true;
   ```

10. Record all channels available

```
for i=1:size(ghiamp_config.Channels,2)
    if (available_channels(1,i))
        ghiamp_config.Channels(1,i).Available = true;
        ghiamp_config.Channels(1,i).Acquire = true;
        % do not use filters
        ghiamp_config.Channels(1,i).BandpassFilterIndex = -1;
        ghiamp_config.Channels(1,i).NotchFilterIndex = -1;
        % do not use a bipolar channel
        ghiamp_config.Channels(1,i).ReferenceChannel = 0;
    end
end
```

11. Set device configuration to `gtecDeviceInterface`

```
gds_interface.DeviceConfigurations = ghiamp_config;
```

12. Configure g.HIamp with the current settings

```
gds_interface.SetConfiguration();
```

13. Start data acquisition and streaming from server

```
gds_interface.StartDataAcquisition();
```

14. Get data acquired from g.HIamp, `2560` samples for 10 seconds of data

```
samples_acquired = 0;
while (samples_acquired < 2560)
    [scans_received, data] = gds_interface.GetData(4);
    samples_acquired = samples_acquired + scans_received;
end
```

15. End streaming and stop data acquisition and delete and clear `gtecDeviceInterface`

```
gds_interface.StopDataAcquisition();
delete(gds_interface);
clear gds_interface;
```

This example is also available as MATLAB script `gHIampAPIDIDemo.m` in the folder

`C:\Users\<user_name>\Documents\gtec\gNEEDaccessMATLABAPI\Examples\gHIamp`

g.USBamp

When enabled, digital inputs of g.USBamp are recorded as additional channel ranging from 0 to 255. To record analog channels and digital inputs of g.USBamp please perform the following steps:

1. Create a `gtecDeviceInterface` object with

   ```
   gds_interface = gtecDeviceInterface;
   ```

2. Set the IP and ports for the host and the client. If used on a local computer, set the following parameters

   ```
   gds_interface.IPAddressHost = '127.0.0.1';
   gds_interface.IPAddressLocal = '127.0.0.1';
   gds_interface.HostPort = 50223;
   gds_interface.LocalPort = 50224;
   ```

3. Get the device(s) currently connected

   ```
   connected_devices = GetConnectedDevices();
   ```

4. Use `gUSBampDeviceConfiguration` to create a g.USBamp device configuration and, as there should only be one g.USBamp device available, set the device serial as name

   ```
   gusbamp_config = gUSBampDeviceConfiguration();
   gusbamp_config.Name = connected_devices(1,1).Name;
   ```

5. To use functions requiring a device serial later, set this newly created device configuration to the `gtecDeviceInterface` object

   ```
   gds_interface.DeviceConfigurations = gusbamp_config;
   ```

6. Get available channels

   ```
   available_channels = gds_interface.GetAvailableChannels();
   ```

7. Set the sampling rate to `256` Hz and number of scans to `8`

   ```
   gusbamp_config.SamplingRate = 256;
   gusbamp_config.NumberOfScans = 8;
   ```

8. Enable digital inputs for recording

   ```
   gusbamp_config.TriggerEnabled = true;
   ```

9. Record all channels available

```
for i=1:size(gusbamp_config.Channels,2)
    if (available_channels(1,i))
        gusbamp_config.Channels(1,i).Available = true;
        gusbamp_config.Channels(1,i).Acquire = true;
        % do not use filters
        gusbamp_config.Channels(1,i).BandpassFilterIndex = -1;
        gusbamp_config.Channels(1,i).NotchFilterIndex = -1;
        % do not use a bipolar channel
        gusbamp_config.Channels(1,i).ReferenceChannel = 0;
    end
end
```

10. Set device configuration to `gtecDeviceInterface`

```
gds_interface.DeviceConfigurations = gusbamp_config;
```

11. Configure g.USBamp with the current settings

```
gds_interface.SetConfiguration();
```

12. Start data acquisition and streaming from server

```
gds_interface.StartDataAcquisition();
```

13. Get data acquired from g.USBamp, `2560` samples for 10 seconds of data

```
samples_acquired = 0;
while (samples_acquired < 2560)
    [scans_received, data] = gds_interface.GetData(4);
    samples_acquired = samples_acquired + scans_received;
end
```

14. End streaming and stop data acquisition and delete and clear `gtecDeviceInterface`

```
gds_interface.StopDataAcquisition();
delete(gds_interface);
clear gds_interface;
```

This example is also available as MATLAB script `gUSBampAPIDIDemo.m` in the folder

`C:\Users\<user_name>\Documents\gtec\gNEEDaccessMATLABAPI\Examples\gUSBamp`

Display data

MATLAB's DSP System Toolbox provides a time scope, which can be used to display acquired data right during data acquisition.

g.Nautilus

NOTE: The example contained in the g.Nautilus examples installation folder (default `C:\Users\<user_name>\Documents\gtec\gNEEDaccessMATLABAPI\Examples\gNauailus`) is using a 32 channel g.Nautilus device. If another device is used (8, 16 or 64 channels), the code has to adapted correspondingly (see comments in the example code).

To use g.tec MATLAB API with g.Nautilus and the time scope please perform the following steps:

1. Create a time scope for 3 input channels, 250Hz sampling frequency, a buffer length of 10 seconds (2500 samples per channel) and an amplitude range from 0 to 16 mV.

```
scope_handle = dsp.TimeScope(3,250, 'BufferLength', 2500,
'TimeAxisLabels', 'Bottom',...
'YLimits', [0 16000], 'TimeSpan', 10, 'LayoutDimensions', [3,1],...
'ReduceUpdates',true, 'YLabel','Amplitude [µV]');
set(scope_handle, 'ActiveDisplay',2, 'YLimits', [0 2510],
'YLabel','Counter');
set(scope_handle, 'ActiveDisplay',3, 'YLimits', [0 1],
'YLabel','Valid');
```

2. Create a `gtecDeviceInterface` object with

```
gds_interface = gtecDeviceInterface;
```

3. Set the IP and ports for the host and the client. If used on a local computer, set the following parameters

```
gds_interface.IPAddressHost = '127.0.0.1';
gds_interface.IPAddressLocal = '127.0.0.1';
gds_interface.HostPort = 50223;
gds_interface.LocalPort = 50224;
```

4. Get the device(s) currently connected

```
connected_devices = GetConnectedDevices();
```

5. Use `gNautilusDeviceConfiguration` to create a g.Nautilus device configuration and, as there should only be one g.Nautilus device available, set the device serial as name

```
gnautilus_config = gNautilusDeviceConfiguration();
gnautilus_config.Name = connected_devices(1,1).Name;
```

6. To use functions requiring a device serial later, set this newly created device configuration to the `gtecDeviceInterface` object

```
gds_interface.DeviceConfigurations = gnautilus_config;
```

7. Get available channels, supported sensitivities and supported input sources

```
available_channels = gds_interface.GetAvailableChannels();
supported_sensitivities = gds_interface.GetSupportedSensitivities();
supported_input_sources = gds_interface.GetSupportedInputSources();
```

8. Set the sampling rate to `250` Hz and number of scans to `4`

```
gnautilus_config.SamplingRate = 250;
gnautilus_config.NumberOfScans = 4;
```

9. Set input source to record the internal test signal of g.Nautilus, a square wave applied to all analog channels with a frequency of 10 Hz and an amplitude of 15 mV

```
gnautilus_config.InputSource = supported_input_source(3).Value;
```

10. Enable additional channels counter and validation indicator

```
gnautilus_config.Counter = true;
gnautilus_config.ValidationIndicator = true;
```

11. Record all channels available

```
for i=1:size(gNautilusConfig.Channels,2)
    if (available_channels(1,i))
        gnautilus_config.Channels(1,i).Available = true;
        gnautilus_config.Channels(1,i).Acquire = true;
        % set sensitivity to 187.5 mV
        gnautilus_config.Channels(1,i).Sensitivity = ...
              supported_sensitivities(6);
        % do not use channel for CAR and noise reduction
        gnautilus_config.Channels(1,i).UsedForNoiseReduction = false;
        gnautilus_config.Channels(1,i).UsedForCAR = false;
        % do not use filters
        gnautilus_config.Channels(1,i).BandpassFilterIndex = -1;
        gnautilus_config.Channels(1,i).NotchFilterIndex = -1;
        % do not use a bipolar channel
        gnautilus_config.Channels(1,i).BipolarChannel = -1;
    end
end
```

12. Set device configuration to `gtecDeviceInterface`

```
gds_interface.DeviceConfigurations = gnautilus_config;
```

13. Configure g.Nautilus with the current settings

```
gds_interface.SetConfiguration();
```

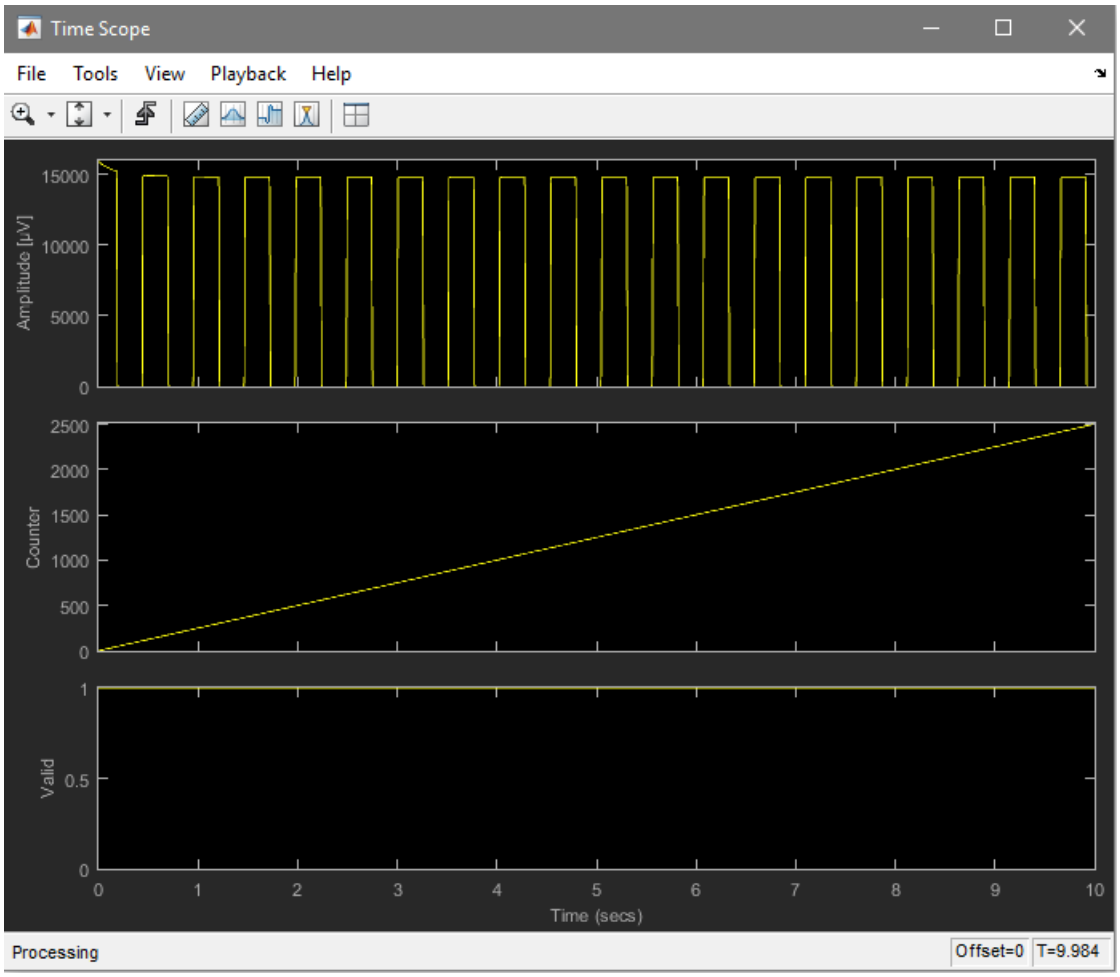
14. Start data acquisition and streaming from server

```
gds_interface.StartDataAcquisition();
```

15. Get data acquired from g.Nautilus, `2500` samples for 10 seconds of data and display data in the time scope

```
samples_acquired = 0;
while (samples_acquired < 2500)
    [scans_received, data] = gds_interface.GetData(4);
    step(scope_handle, data(:,1),data(:,9),data(:,10));
    samples_acquired = samples_acquired + scans_received;
end
```

16. DSP time scope displays the first analog channel, counter and the validation indicator



17. End streaming and stop data acquisition and delete and clear `gtecDeviceInterface` as well as the time scope handle

```
gds_interface.StopDataAcquisition();

delete(gds_interface);
scope_handle.hide;
```

```
clear gds_interface;
clear scope_handle;
```

This example is also available as MATLAB script `gNautilusAPIScope.m` in the folder

`C:\Users\<user_name>\Documents\gtec\gNEEDaccessMATLABAPI\Examples\gNautilus`

g.HIamp

To use g.tec MATLAB API with g.HIamp and the time scope please perform the following steps:

1. Create a time scope for 2 input channels, 256Hz sampling frequency, a buffer length of 10 seconds (2560 samples per channel) and an amplitude range from 0 to 16 mV.

```
scope_handle = dsp.TimeScope(2, 256, 'BufferLength', 2560,...
    'TimeAxisLabels', 'Bottom', 'YLimits', [0 2560], 'TimeSpan',
10,...
    'LayoutDimensions', [2,1], 'ReduceUpdates', true, 'YLabel',
'Counter');
```

2. Create a `gtecDeviceInterface` object with

```
gds_interface = gtecDeviceInterface;
```

3. Set the IP and ports for the host and the client. If used on a local computer, set the following parameters

```
gds_interface.IPAddressHost = '127.0.0.1';
gds_interface.IPAddressLocal = '127.0.0.1';
gds_interface.HostPort = 50223;
gds_interface.LocalPort = 50224;
```

4. Get the device(s) currently connected

```
connected_devices = GetConnectedDevices();
```

5. Create a `gHIampDeviceConfiguration` and, as there should only be one g.HIamp device available, set the device serial as name

```
ghiamp_config = gHIampDeviceConfiguration();
ghiamp_config.Name = connected_devices(1,1).Name;
```

6. To use functions requiring a device serial later, set this newly created device configuration to the `gtecDeviceInterface` object

```
gds_interface.DeviceConfigurations = ghiamp_config;
```

7. Get available channels

```
available_channels = gds_interface.GetAvailableChannels();
```

8. Set the sampling rate to `256` Hz and number of scans to `4`

```
ghiamp_config.SamplingRate = 256;
ghiamp_config.NumberOfScans = 4;
```

9. Activate the internal test signal of g.HIamp, a square wave applied to all analog channels with a frequency of 10 Hz and amplitude of 15 mV. Shortcut analog channel 1 to ground.

```
ghiamp_siggen = gHIampInternalSignalGenerator();
ghiamp_siggen.Enabled = true;
ghiamp_siggen.Frequency = 10;
ghiamp_config.InternalSignalGenerator = ghiamp_siggen;
```

10. Enable counter

```
ghiamp_config.CounterEnabled = true;
```

11. Record all channels available

```
for i=1:size(ghiamp_config.Channels,2)
    if (available_channels(1,i))
        ghiamp_config.Channels(1,i).Available = true;
        if (i <= 80)
            ghiamp_config.Channels(1,i).Acquire = true;
        else
            ghiamp_config.Channels(1,i).Acquire = false;
        end
        % do not use filters
        ghiamp_config.Channels(1,i).BandpassFilterIndex = -1;
        ghiamp_config.Channels(1,i).NotchFilterIndex = -1;
        % do not use a bipolar channel
        ghiamp_config.Channels(1,i).ReferenceChannel = 0;
    end
end
```

12. Set device configuration to `gtecDeviceInterface`

```
gds_interface.DeviceConfigurations = ghiamp_config;
```

13. Configure g.HIamp with the current settings

```
gds_interface.SetConfiguration();
```

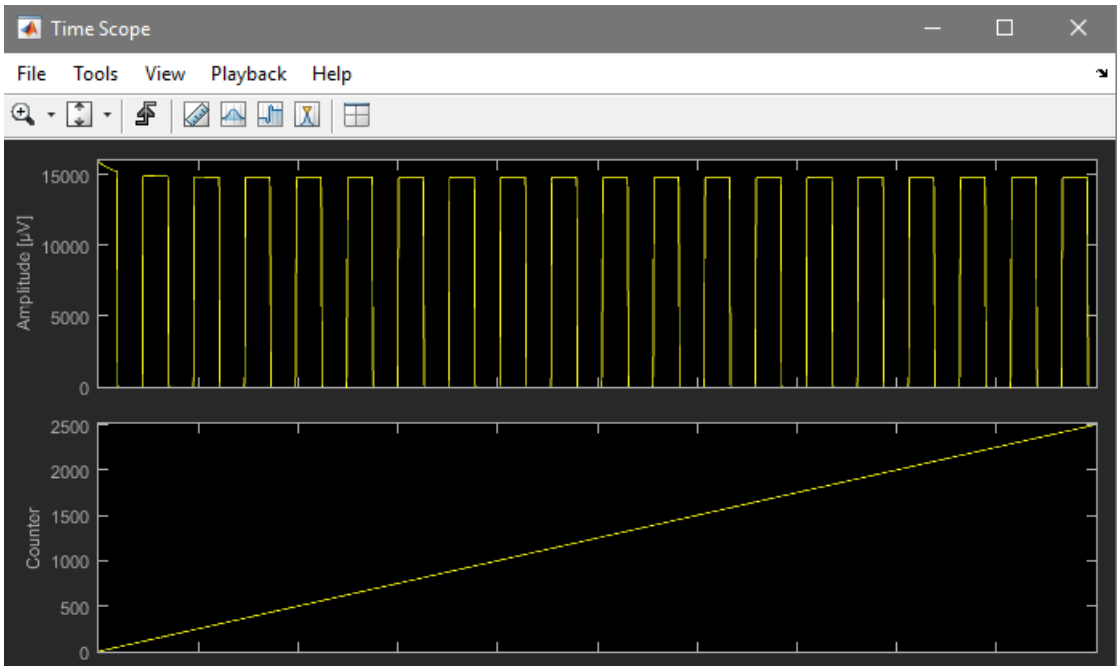
14. Start data acquisition and streaming from server

```
gds_interface.StartDataAcquisition();
```

15. Get data acquired from g.HIamp, `2560` samples for 10 seconds of data and display data in the time scope

```
samples_acquired = 0;
while (samples_acquired < 2560)
    [scans_received, data] = gds_interface.GetData(4);
    step(scope_handle, data(:,2),data(:,1));
    samples_acquired = samples_acquired + scans_received;
end
```

16. DSP time scope displays the analog channel and the counter



17. End streaming and stop data acquisition and delete and clear `gtecDeviceInterface` as well as the time scope handle

```
gds_interface.StopDataAcquisition();

delete(gds_interface);
scope_handle.hide;

clear gds_interface;
clear scope_handle;
```

This example is also available as MATLAB script `gHIampAPIScope.m` in the folder

`C:\Users\<user_name>\Documents\gtec\gNEEDaccessMATLABAPI\Examples\gHIamp`

g.USBamp

To use g.tec MATLAB API with g.USBamp and the time scope please perform the following steps:

1. Create a time scope for 2 input channels, 256Hz sampling frequency, a buffer length of 10 seconds (2560 samples per channel) and an amplitude range from -200 to 200 mV.

```
scope_handle = dsp.TimeScope(2, 256, 'BufferLength', 2560,...
    'TimeAxisLabels', 'Bottom', 'YLimits', [-200000 100000],...
    'TimeSpan', 10, 'LayoutDimensions', [2,1], 'ReduceUpdates',
true,...
    'YLabel', 'Amplitude [µV]');
```

2. Create a `gtecDeviceInterface` object with

```
gds_interface = gtecDeviceInterface;
```

3. Set the IP and ports for the host and the client. If used on a local computer, set the following parameters

```
gds_interface.IPAddressHost = '127.0.0.1';
gds_interface.IPAddressLocal = '127.0.0.1';
gds_interface.HostPort = 50223;
gds_interface.LocalPort = 50224;
```

4. Get the device(s) currently connected

```
connected_devices = GetConnectedDevices();
```

5. Create a `gUSBampDeviceConfiguration` and, as there should only be one g.USBamp device available, set the device serial as name

```
gusbamp_config = gUSBampDeviceConfiguration();
gusbamp_config.Name = connected_devices(1,1).Name;
```

6. To use functions requiring a device serial later, set this newly created device configuration to the `gtecDeviceInterface` object

```
gds_interface.DeviceConfigurations = gusbamp_config;
```

7. Get available channels

```
available_channels = gds_interface.GetAvailableChannels();
```

8. Set the sampling rate to `256` Hz and number of scans to `4`

```
gusbamp_config.SamplingRate = 256;
gusbamp_config.NumberOfScans = 4;
```

9. Activate the internal test signal of g.USBamp, a square wave applied to all analog channels with a frequency of 10 Hz and amplitude of 200 mV. Shortcut analog channel 1 to ground.

```
gusbamp_siggen = gUSBampInternalSignalGenerator();
gusbamp_siggen.Enabled = true;
gusbamp_siggen.Frequency = 10;
gusbamp_siggen.WaveShape = 3;
gusbamp_siggen.Amplitude = 200;
gusbamp_siggen.Offset = 0;
gusbamp_config.InternalSignalGenerator = gusbamp_siggen;
```

10. Enable counter

```
gusbamp_config.CounterEnabled = true;
```

11. Record all channels available

```
for i=1:size(gusbamp_config.Channels,2)
    if (available_channels(1,i))
        gusbamp_config.Channels(1,i).Available = true;
        gusbamp_config.Channels(1,i).Acquire = true;
        % do not use filters
        gusbamp_config.Channels(1,i).BandpassFilterIndex = -1;
        gusbamp_config.Channels(1,i).NotchFilterIndex = -1;
        % do not use a bipolar channel
        gusbamp_config.Channels(1,i).BipolarChannel = 0;
    end
end
```

12. Set device configuration to `gtecDeviceInterface`

```
gds_interface.DeviceConfiguration = gusbamp_config;
```

13. Configure g.USBamp with the current settings

```
gds_interface.SetConfiguration();
```

14. Start data acquisition and streaming from server

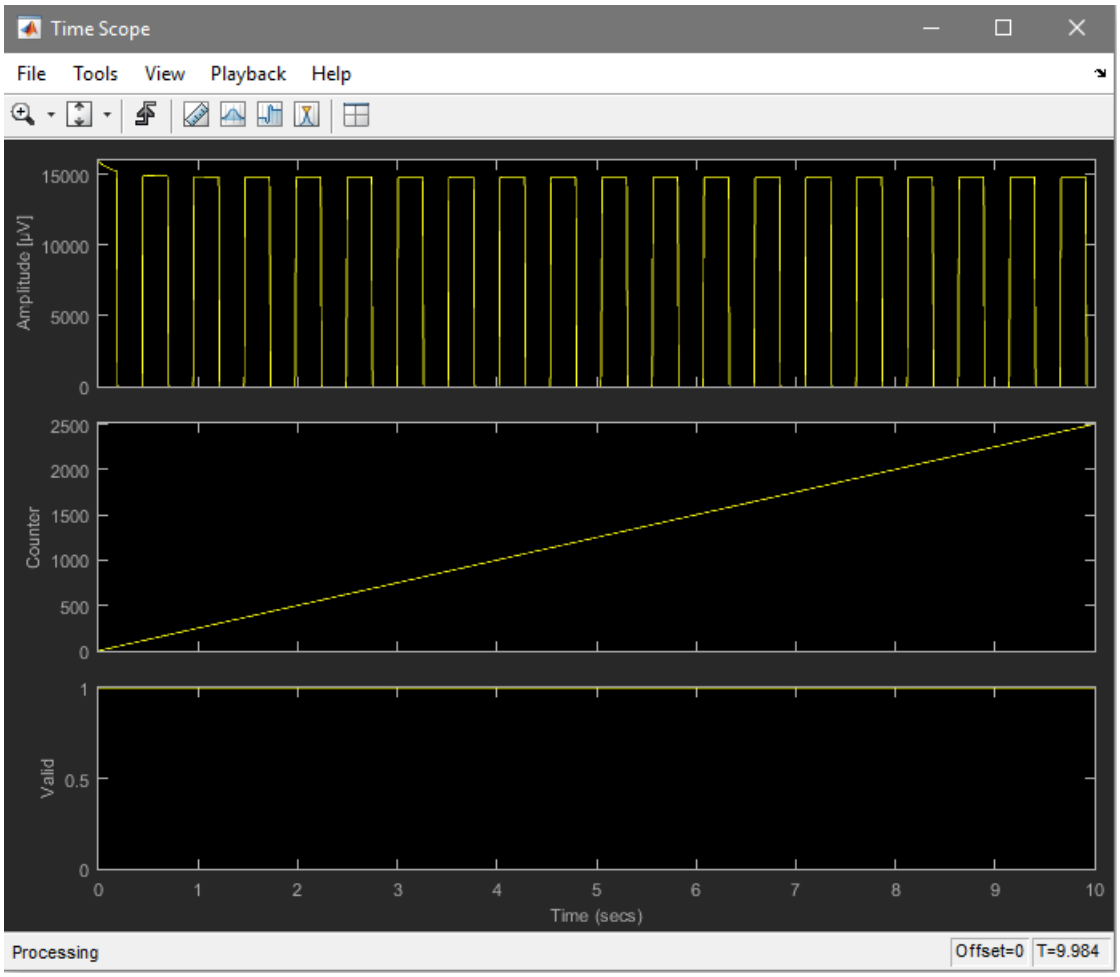
```
gds_interface.StartDataAcquisition();
```

15. Get data acquired from g.USBamp, `2560` samples for 10 seconds of data and display data in the time scope

```
samples_acquired = 0;
while (samples_acquired < 2560)
    [scans_received, data] = gds_interface.GetData(4);

    step(scope_handle, data(:,1), data(:,16));
    samples_acquired = samples_acquired + scans_received;
end
```

16. DSP time scope displays the first analog channel, counter and the validation indicator



17. End streaming and stop data acquisition and delete and clear `gtecDeviceInterface` as well as the time scope handle

```
gds_interface.StopDataAcquisition();

delete(gds_interface);
scope_handle.hide;

clear gds_interface;
clear scope_handle;
```

This example is also available as MATLAB script `gUSBampAPIScope.m` in the folder

`C:\Users\<user_name>\Documents\gtec\gNEEDaccessMATLABAPI\Examples\gUSBamp`

Synchronizing g.USBamp

This demo shows how to synchronize two g.USBamps with the g.NEEDaccess MATLAB API and display data in the DSP time scope. It requires two amplifiers and a cable to synchronize the two devices.

NOTE: As the `GetConnectedDevices` function of the `gtecDeviceInterface` object returns the connected g.USBamps as they are recognized by GDS it might be useful to check the order in the returned object (steps 2 to 4 in this section) separately in the MATLAB command window before running the demo. Adapt the code described in step 5 afterwards accordingly.

The following steps are required to execute this demo:

1. Create the DSP time scope with two axes for displaying data during acquisition

```
scope_handle = dsp.TimeScope(2,256, 'BufferLength', 2560,...
'YLimits', [-200000 200000], 'TimeSpan', 5, 'LayoutDimensions',
[2,1],...
'ReduceUpdates',true, 'YLabel','Amplitude [µV]');
set(scope_handle, 'ActiveDisplay',2, 'YLimits', [-200000 200000],
'YLabel','Amplitude [µV]');
```

2. Create a `gtecDeviceInterface` object with

```
gds_interface = gtecDeviceInterface;
```

3. Set the IP and ports for the host and the client. If used on a local computer, set the following parameters

```
gds_interface.IPAddressHost = '127.0.0.1';
gds_interface.IPAddressLocal = '127.0.0.1';
gds_interface.HostPort = 50223;
gds_interface.LocalPort = 50224;
```

4. Get the device(s) currently connected

```
connected_devices = gds_interface.GetConnectedDevices();
```

5. Use `gUSBampDeviceConfiguration` to create an array of two g.USBamp device configurations and, as there should only be two g.USBamp devices available, set the device serials as names. Make sure that first device is master, second device must be slave.

```
gusbamp_configs(1,1:2) = gUSBampDeviceConfiguration();
gusbamp_configs(1,1).Name = connected_devices(1,1).Name;
gusbamp_configs(1,2).Name = connected_devices(1,2).Name;
```

6. To use functions requiring a device serial later, set this newly created device configuration to the `gtecDeviceInterface` object

```
gds_interface.DeviceConfigurations = gusbamp_configs;
```

7. Get available channels

```
available_channels_master =
gds_interface.GetAvailableChannels(connected_devices(1,1).Name);
available_channels_slave =
gds_interface.GetAvailableChannels(connected_devices(1,2).Name);
```

8. Edit both configurations to have the same sampling rate (`256` Hz) and number of scans (`8` scans) (will not work if configured differently), enable and configure internal test signal as shown below

```
for i=1:size(gusbamp_configs,2)
    gusbamp_configs(1,i).SamplingRate = 256;
    gusbamp_configs(1,i).NumberOfScans = 8;
    gusbamp_siggen = gUSBampInternalSignalGenerator();
    gusbamp_siggen.Enabled = true;
    gusbamp_siggen.Frequency = 10;
    gusbamp_siggen.WaveShape = 3;
    gusbamp_siggen.Amplitude = 200;
    gusbamp_siggen.Offset = 0;
    gusbamp_configs(1,i).InternalSignalGenerator = gusbamp_siggen;
    for j=1:size(gusbamp_configs(1,i).Channels,2)
        if (available_channels_master(1,j))
            gusbamp_configs(1,i).Channels(1,j).Available = true;
            gusbamp_configs(1,i).Channels(1,j).Acquire = true;
            % do not use filters
            gusbamp_configs(1,i).Channels(1,j).BandpassFilterIndex = -1;
            gusbamp_configs(1,i).Channels(1,j).NotchFilterIndex = -1;
            % do not use a bipolar channel
            gusbamp_configs(1,i).Channels(1,j).BipolarChannel = 0;
        end
    end
end
```

9. Set device configuration to `gtecDeviceInterface`

```
gds_interface.DeviceConfigurations = gusbamp_configs;
```

10. Configure both amplifiers with the current settings

```
gds_interface.SetConfiguration();
```

11. Start data acquisition and streaming from server

```
gds_interface.StartDataAcquisition();
```

12. Get data acquired from both devices, `2560` samples for 10 seconds of data and display data in the time scope

```
samples_acquired = 0;
while (samples_acquired < 2560)
    [scans_received, data] = gds_interface.GetData(4);
    % display first analog channel for each device in time scope
    step(scope_handle, data(:,1), data(:,17));
end
```

13. End streaming and stop data acquisition and delete and clear `gtecDeviceInterface` as well as the time scope handle

```
gds_interface.StopDataAcquisition();

delete(gds_interface);
scope_handle.hide;

clear gds_interface;
clear scope_handle;
```

This example is also available as MATLAB script `gUSBampAPISynchDemo.m` in the folder

`C:\Users\<user_name>\Documents\gtec\gNEEDaccessMATLABAPI\Examples\gUSBamp`

Help

g.NEEDaccess MATLAB API provides a printable documentation.

The printable documentation is stored under

`C:\Program Files\gtec\gNEEDaccessMATLABAPI\Help`

as `gNEEDaccessMATLABAPI.pdf`. Use Acrobat Reader to view the documentation.

Product Page

Please visit our homepage www.gtec.at for

- Update announcements
- Downloads
- Troubleshooting
- Additional demonstrations


g.tec
GUGER
TECHNOLOGIES
contact information
g.tec medical engineering GmbH
Sierningstrasse 14
4521 Schiedlberg
Austria
tel. +43 7251 22240
fax. +43 7251 22240 39
web: www.gtec.at
e-mail: office@gtec.at