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

g.tec® GUGER TECHNOLOGIES

g.® USBamp USB BIOSIGNAL AMPLIFIER

MATLAB API User Manual

V3.12.00

How to contact g.tec:

☎	++43-7251-22240-0	**Phone**
🖷	++43-7251-22240-39	**Fax**
🖂	g.tec medical engineering GmbH Sierningstrasse 14, 4521 Schiedlberg, Austria	**Mail**
🖧	http://www.gtec.at	**Web**
@	office@gtec.at	**e-mail**
	AT/CA01/RC000989-00	**ÖBIG Reg. number**

CONTENT:

The intended function of the equipment

Measuring, recording and analysis of electrical activity of the brain and/or through the attachment of multiple electrodes at various locations to aid in monitoring and diagnosis as routinely found in clinical settings for EEG.

The device **must not** be used for patient monitoring. The device **must not** be used for the determination of brain death. Additional examinations are needed for diagnosis and no diagnosis may be done only based on using this device.

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, can do feature extraction and can 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.USBamp 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'*,...)
Menu items	**Boldface**	Select **Save** from the **File** menu

Installation and Configuration

This chapter includes the following sections:

Hardware and Software Requirements

Hardware Requirements

g.USBamp MATLAB API requires a PC compatible desktop or notebook workstation running Microsoft Windows.

The table below lists optimal settings:

Hardware	Properties
CPU	Pentium working at 2000 MHz
Harddisk	20-30 GB
RAM	1024 MB
USB 2.0 high speed port	One free USB port for each g.USBamp

Software Requirements

g.USBamp MATLAB API requires the installation of the g.USBamp driver which is shipped with the device, MATLAB and of the Data Acquisition Toolbox. Make sure that the MATLAB installation works correctly before installing the g.USBamp software. Depending on your Windows operating system administrator rights might be necessary for the installation.

NOTE: It is highly recommended to turn off the User Account Control of Windows 7 operating system when using g.USBamp MATLAB API!

Software	Version
g.USBamp driver	3.12.00
MATLAB	R2012a
Data Acquisition Toolbox	R2012a
Windows	Windows 7 Professional English Win32
Acrobat Reader	10.1.3

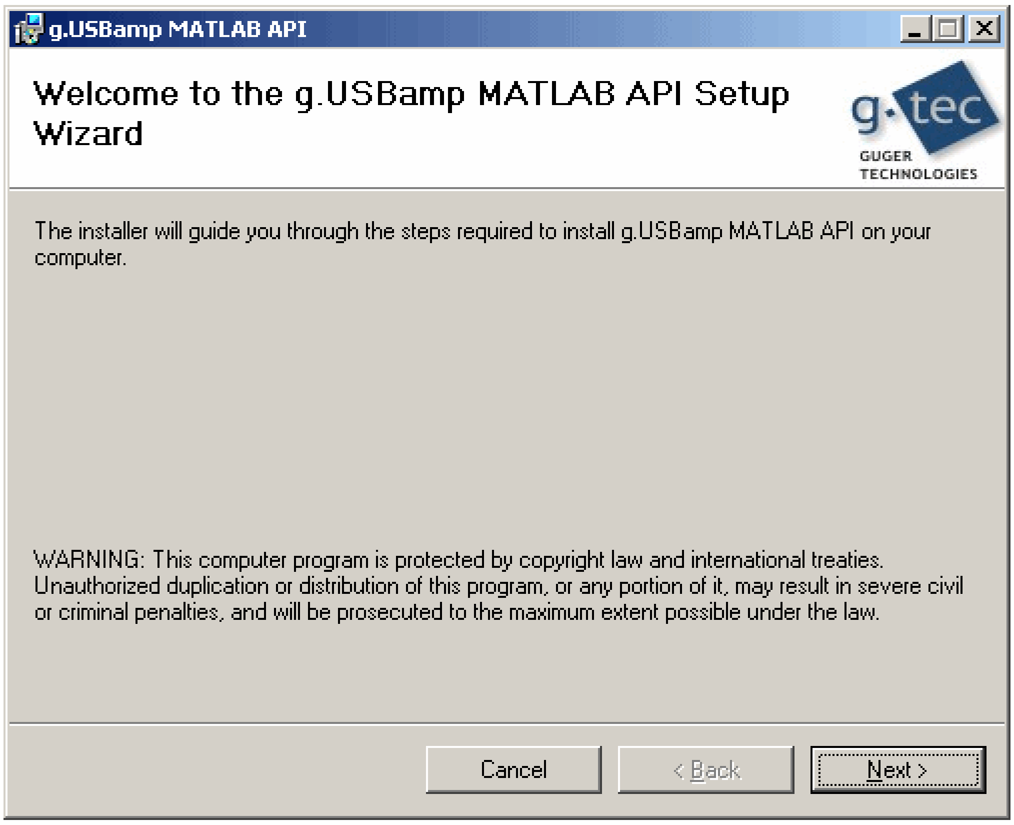
Installation from a CD

Note: Please make sure that g.USBamp Driver is installed before g.USBamp MATLAB API is installed!

The installation of g.USbamp MATLAB API consists of two steps:

1. Insert the g.tec product CD into the CD-drive and change to the `g.USBamp\g.USBamp MATLAB API` directory of your CD-drive and double-click the `setup.exe` file and follow the instructions on the screen to install the g.USBamp MATLAB API.

   Please read the License Agreement for g.USBamp MATLAB API and if you agree with the terms, click **Next**. After installation, you can view the license agreement in the file `license.rtf` located in the `gtec\gUSBampMatlabAPI` directory.



The installer will guide you through the steps required to install g.USBamp MATLAB API on your computer.

WARNING: This computer program is protected by copyright law and international treaties. Unauthorized duplication or distribution of this program, or any portion of it, may result in severe civil or criminal penalties, and will be prosecuted to the maximum extent possible under the law.

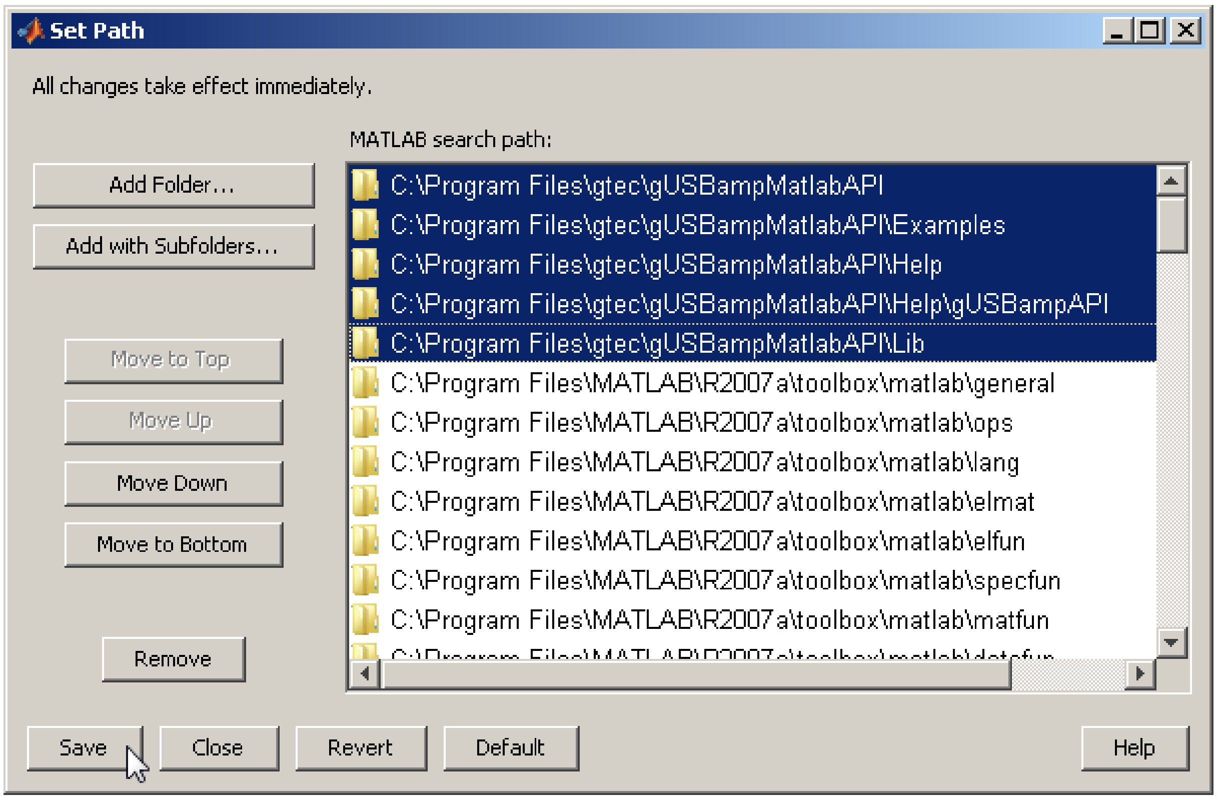
Then just follow the instruction on the screen.

2. When the `setup.exe` has finished you need to set the 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

   `your Installation Folder\gtec\gUSBampMatlabAPI`

   to add all subdirectories:



```
C:\Program Files\gtec\gUSBampMatlabAPI
C:\Program Files\gtec\gUSBampMatlabAPI\Examples
C:\Program Files\gtec\gUSBampMatlabAPI\Help
C:\Program Files\gtec\gUSBampMatlabAPI\Help\gUSBampAPI
C:\Program Files\gtec\gUSBampMatlabAPI\Lib
C:\Program Files\MATLAB\R2007a\toolbox\matlab\general
C:\Program Files\MATLAB\R2007a\toolbox\matlab\ops
C:\Program Files\MATLAB\R2007a\toolbox\matlab\lang
C:\Program Files\MATLAB\R2007a\toolbox\matlab\elmat
C:\Program Files\MATLAB\R2007a\toolbox\matlab\elfun
C:\Program Files\MATLAB\R2007a\toolbox\matlab\specfun
C:\Program Files\MATLAB\R2007a\toolbox\matlab\matfun
```

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

Files on your Computer

g.USBamp files - are stored under

```
Your Installation Folder\gtec\gUSBampMatlabAPI
```

Example files - are stored in the subdirectory

```
Your Installation Folder\gtec\gUSBampMatlabAPI\Examples
```

Help files - are stored under

```
Your Installation Folder\gtec\gUSBampMatlabAPI\Help
```

Quick Start

To test the g.USBamp MATLAB API configuration on your system please perform the following example:

1. Start the MATLAB command window. See your MATLAB documentation if you are not sure how to do this.

2. Type `h=daqhwinfo` into the MATLAB command line. This command searches for all installed data acquisition devices on your PC.

   ```
   h=daqhwinfo

   h =

             ToolboxName: 'Data Acquisition Toolbox'
          ToolboxVersion: '2.18 (R2011a)'
           MATLABVersion: '7.12 (R2011a)'
       InstalledAdaptors: {4x1 cell}
   ```

3. `InstalledAdaptors` contains the adaptor names of the installed data acquisition devices

   ```
   h.InstalledAdaptors

      ans =

          'guadaq'
          'nidaq'
          'parallel'
          'winsound'
   ```

   `guadaq` represents the g.USBamp data acquisition adaptor.

4. If the `guadaq` adaptor is not found on your system register the g.USBamp adaptor with

   ```
   daqregister('guadaq')

   ans =

   'guadaq.dll' successfully registered.
   ```

   This command writes the adaptor into the registry and therefore it is not necessary to perform the command the next time.

5. Check how many g.USBamps are connected to your PC
   `daqhwinfo('guadaq')`

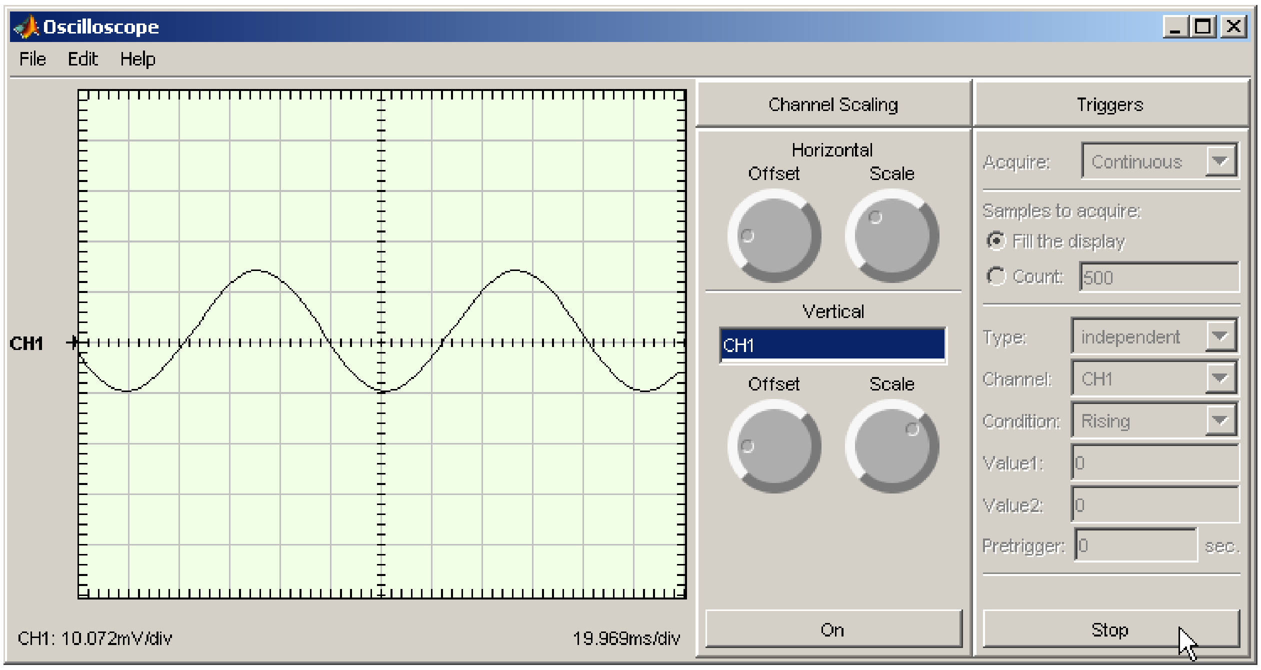
```
   ans =

   AdaptorDllName: 'C:\Program Files
                    \gtec\gUSBampMatlabAPI\Lib\guadaq.dll'
           AdaptorDllVersion: '5.11'
                 AdaptorName: 'guadaq'
                  BoardNames: {'g.USBamp'}
           InstalledBoardIds: {'1'}
       ObjectConstructorName: {'analoginput('guadaq',1)'   '' ,…
                               'digitalio('guadaq',1)'}
```

   `InstalledBoardIds` returns the Ids of the installed g.USBamps. In this case one amplifier with Id `1` is connected. `AdaptorDllVersion` is the version number of the g.USBamp adaptor. `daqhwinfo` registers the g.USBamp with this Id into your system. The next time when the g.USBamp with this serial number is connected to your system, the Id will be the same.

6. Configure g.USBamp to acquire 1 channel in calibration mode. Therefore an analog input object with the adaptor `guadaq` of device 1 must be created.

```
   ai = analoginput('guadaq',1);
```



CH1: 10.072mV/div
19.969ms/div
Acquire: Continuous
Samples to acquire:
Fill the display
Count: 500
Type: independent
Channel: CH1
Condition: Rising
Value1: 0
Value2: 0
Pretrigger: 0 sec.

   Add the first channel

```
   addchannel (ai,1)
```

   and set g.USBamp to calibration mode

```
   set(ai,'Mode','Calibration');
```

   Start the Data Acquisition Oscilloscope

```
   softscope(ai);
```

7. Press the **Trigger** button to start g.USBamp. The data window displays a calibration signal in form of a sine wave.

8. Use the **Vertical Scale** and **Horizontal Scale** buttons to adjust the scope settings

9. Press the **Stop** button and close the window.

Data Acquisition

A data acquisition session consists of the following steps:

1. Create a device object – a device object is created by using the `analoginput` function. Device objects are used to access the g.USBamp device.

2. Add channels – after the g.USBamp device object is created, analog input channels must be added. The channel number defines which analog input should be acquired.

3. Configure properties – use the `set` function to assign g.USBamp properties like sample rate, filter settings,…

4. Acquire data – use the `start` function to start the biosignal data acquisition

5. Clean up – use the `delete` function to remove the device object from memory and the clear function to remove it from MATLAB workspace

Example: Data Acquisition of 16 channels

This example demonstrates the steps for acquiring data with g.USBamp. Run this example by typing `gUSBampAPIDemo1` into the MATLAB command line.

1. Create device object – create an analog input object for g.USBamp.

   The connected g.USBamps are found with the following command:

   ```
      daqhwinfo('guadaq')

      ans =

              AdaptorDllName: 'C:\Program Files
                    \gtec\gUSBampMatlabAPI\Lib\guadaq.dll'
           AdaptorDllVersion: '5.09a'
                 AdaptorName: 'guadaq'
                  BoardNames: {'g.USBamp'}
           InstalledBoardIds: {'1'}
       ObjectConstructorName: {'analoginput('guadaq',1)'   '' ,…
                                    'digitalio('guadaq',1)'}
   ```

   Create the analog input object of g.USBamp with Id 1:

   ```
   ai = analoginput('guadaq',1);
   ```

2. Add channels – add 16 channels to the analog input object `ai`

   ```
   addchannel(ai,1:16);
   ```

3. Configure property values – set the sampling frequency of g.USBamp to `256` Hz and acquire data for `2` second

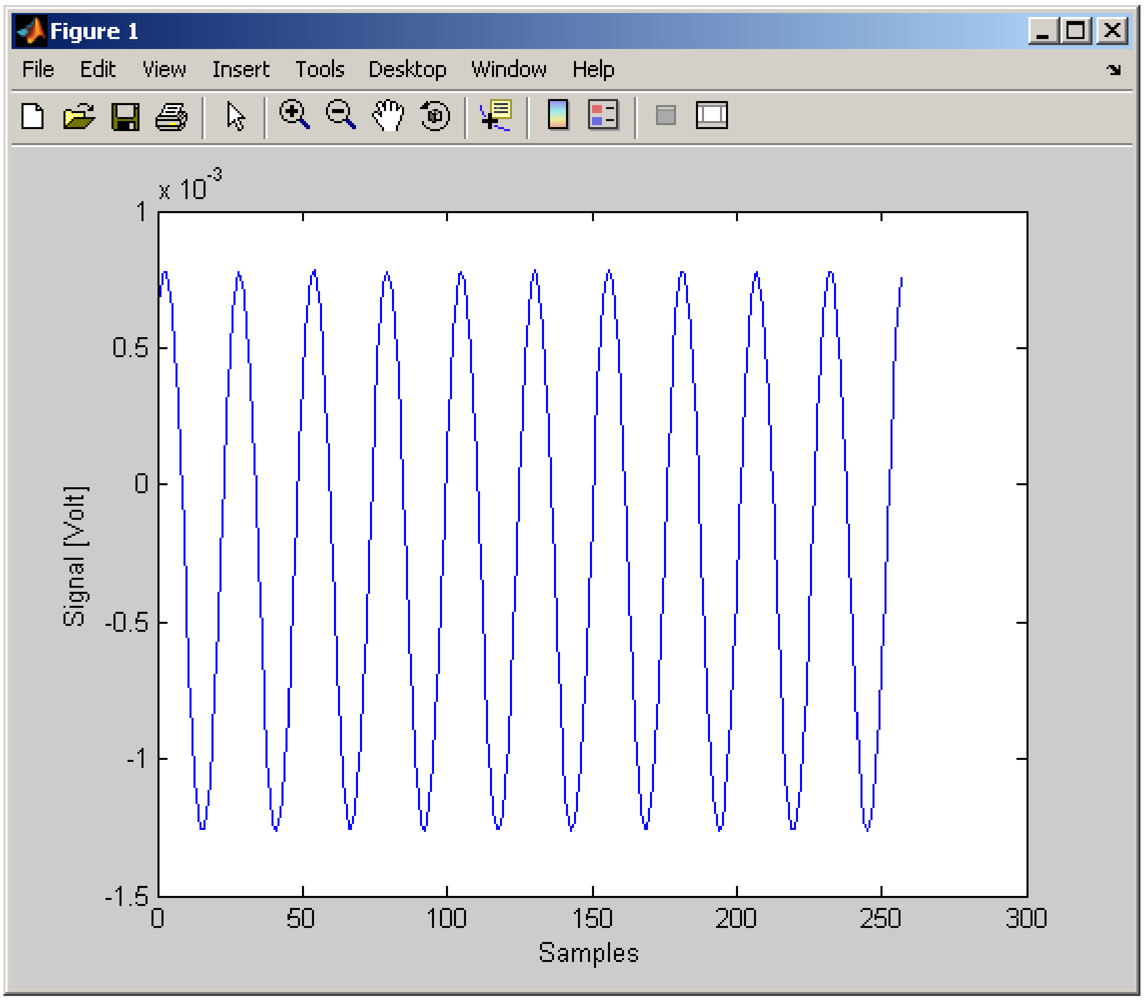
   ```
   set(ai,'SampleRate',256,'SamplesPerTrigger',512);
   ```

4. Acquire data – Start `ai`, wait for 512 samples and extract the data with `getdata`. Before starting connect a sine wave generator to the first channel of g.USBamp.

```
start(ai);
while strcmp(ai.running, 'On')==1
end
data = getdata(ai,ai.SamplesAvailable);
```

5. Plot the first channel and last second of the acquired data and label the figure.

```
plot(data(256:end,1));
xlabel('Samples');
ylabel('Signal [Volts]');
```



6. Clean up – remove the data acquisition object from memory and MATLAB workspace

```
delete(ai)
clear ai
```

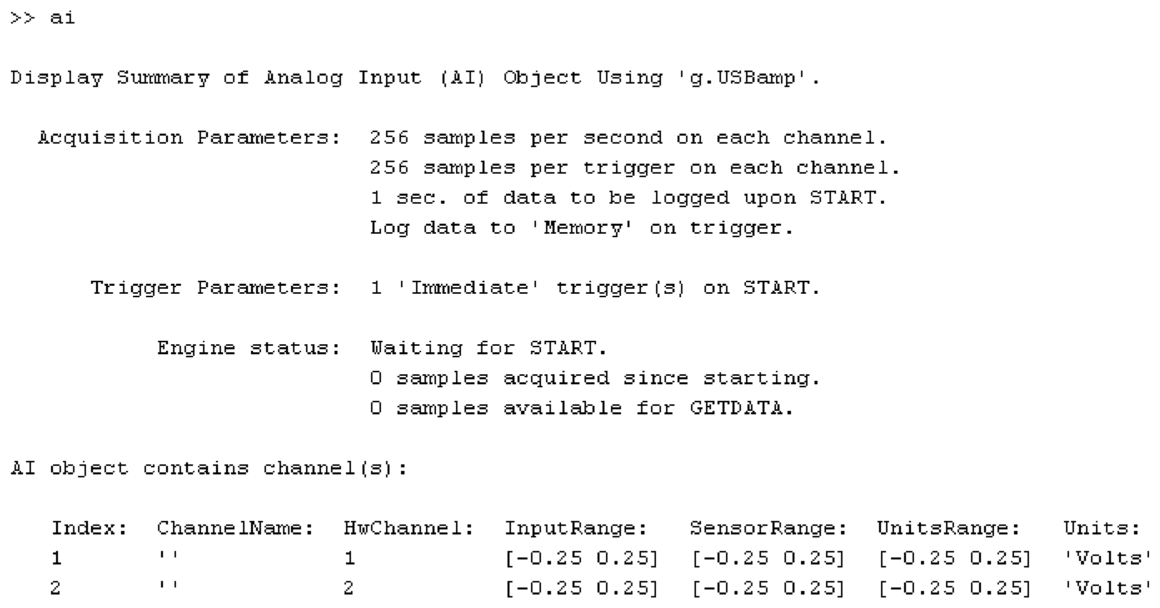
Channel Settings

Input Range and Units

1. Define an analog input object for g.USBamp with Id 1 and add 2 analog input channels:

```
ai = analoginput('guadaq',1);
addchannel(ai,1:2);
```

2. Type `ai` into the MATLAB command window to investigate the channel settings



```
>> ai

Display Summary of Analog Input (AI) Object Using 'g.USBamp'.

  Acquisition Parameters:  256 samples per second on each channel.
                           256 samples per trigger on each channel.
                           1 sec. of data to be logged upon START.
                           Log data to 'Memory' on trigger.

      Trigger Parameters:  1 'Immediate' trigger(s) on START.

           Engine status:  Waiting for START.
                           0 samples acquired since starting.
                           0 samples available for GETDATA.

AI object contains channel(s):

   Index:  ChannelName:  HwChannel:  InputRange:   SensorRange:  UnitsRange:   Units:
   1       ''            1           [-0.25 0.25]  [-0.25 0.25]  [-0.25 0.25]  'Volts'
   2       ''            2           [-0.25 0.25]  [-0.25 0.25]  [-0.25 0.25]  'Volts'
```

Currently 2 channels of g.USBamp are defined. The Input range of the device is ± `250` mV.

Bandpass Filter Settings

Each biosignal channel can be bandpass filtered on g.USBamp. The g.USBamp driver comes with predefined filters which can be shown with the command `gUSBampShowFilter` for a specific sampling frequency.

```
gUSBampShowFilter(256);
```

```
   Valid Bandpass Filters for 256 Hz:
   Filter:  HP:   LP:         Order:      Type:
   ___________________________________________
   32       0.10  0.00        8           butter
   33       1.00  0.00        8           butter
   34       2.00  0.00        8           butter
   35       5.00  0.00        8           butter
   36       0.00  30.00       8           butter
   37       0.00  60.00       8           butter
   38       0.00  100.00      8           butter
   39       0.01  30.00       6           butter
   40       0.01  60.00       8           butter
   41       0.01  100.00      8           butter
   42       0.10  30.00       8           butter
   43       0.10  60.00       8           butter
   44       0.10  100.00      8           butter
   45       0.50  30.00       8           butter
   46       0.50  60.00       8           butter
   47       0.50  100.00      8           butter
   48       2.00  30.00       8           butter
   49       2.00  60.00       8           butter
   50       2.00  100.00      8           butter
   51       5.00  30.00       8           butter
   52       5.00  60.00       8           butter
   53       5.00  100.00      8           butter
```

To define the filter with number 48 for channel 1 type in the MATLAB command line the following code:

```
set(ai.Channel(1),'BPIndex',48);
```

and for channel 2

```
set(ai.Channel(2),'BPIndex',48);
```

This will apply a butterworth bandpass with a lower cut-off frequency of 2 Hz and a upper cut-off frequency of 30 Hz with an order of 2 to channels 1 and 2.

Notch Filter Settings

A Notch filter with 50 Hz or 60 Hz can be applied on each amplifier channel. Use `gUSBampShowFilter` to investigate the possible filter versions.

For 256 Hz the following filters are available:

```
   Valid Notch Filters for 256 Hz:
   Filter:  HP:   LP:         Order:      Type:
   ___________________________________________
   2        48.00 52.00       4           butter
   3        58.00 62.00       4           butter
```

Enable the 50 Hz Notch filter for channel1

```
set(ai.Channel(1),'NotchIndex',2)
```

and show the settings

```
get(ai.Channel(1))
        ChannelName =
        HwChannel = 1
        Index = 1
        InputRange = [-0.25 0.25]
        NativeOffset = 1.21708e-005
        NativeScaling = 1
        Parent = [1x1 analoginput]
        SensorRange = [-0.25 0.25]
        Type = Channel
        Units = Volts
        UnitsRange = [-0.25 0.25]

        GUADAQ specific properties:
        BipolarChannel = 0
        BPHigh = 2.00
        BPIndex = 48
        BPLow = 30.00
        Notch = 50
        NotchIndex = 2
```

`BPHigh` and `BPLow` show the cut-off frequencies of 2 and 30 Hz of the bandpass filters. `Notch` shows that the filter is suppressing the power line frequency at 50 Hz.

On-line Data Acquisition

Use program `gUSBampScope.m` to perform the example below.

1. Define the analog input object for g.USBamp

```
ai = analoginput('guadaq',1);
addchannel(ai,[1]);
```

2. Set the sampling rate to 256 Hz and acquire 10 seconds of data

```
set(ai,'SampleRate',256,'SamplesPerTrigger',10*256);

preview=256;
p = plot(zeros(preview,1)); grid on
start(ai)
```

3. Wait for one second to have at least 256 samples for the visualization

```
while ai.SamplesAcquired < preview
end
```

4. Show the acquired data in the figure

```
while ai.SamplesAcquired < 10 * 256
    data = peekdata(ai,preview);
    set(p,'ydata',data);
    drawnow;
end
```

5. Extract the whole 10 second data segment and plot it

```
data = getdata(ai);
plot(data), grid on
```

6. Clear the analog input object

```
delete(ai);
clear ai
```

Bipolar Derivation

g.USBamp allows to perform a bipolar derivation of two input channels. Use the m-file `gUSBampAPIBipolar.m` to perform the example shown below.

1. Define the analog input object for g.USBamp

   ```
   ai = analoginput('guadaq',1);
   addchannel(ai,[1,2]);
   ```

2. Set the sampling frequency to 256 Hz

   ```
   set(ai,'SampleRate',256,'SamplesPerTrigger',512);
   ```

3. Use bandpass filter with Id 48 for channels 1 and 2

   ```
   set(ai.Channel(1),'BPIndex',48);
   set(ai.Channel(2),'BPIndex',48);
   ```

4. Perform a bipolar derivation of channel 1 by subtracting channel 2. Perform no bipolar derivation of channel 2. Before starting the data acquisition apply a sine wave signal on channels 1 and 2.

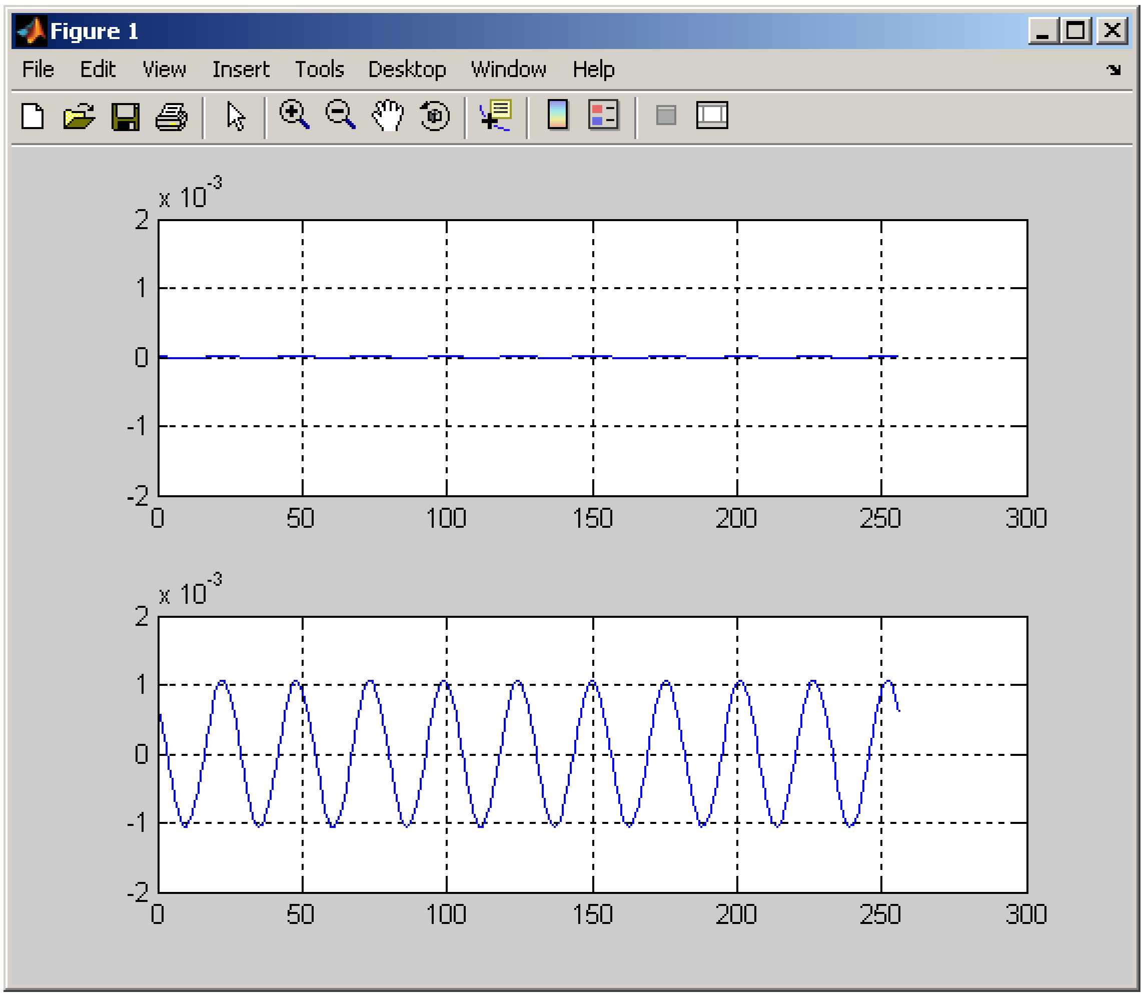
   ```
   set(ai.Channel(1),'BipolarChannel',2);
   set(ai.Channel(2),'BipolarChannel',0);
   start(ai)

   while strcmp(ai.running,'On')==1
   end
   ```

5. Extract the data and plot channels 1 and 2.

   ```
   data = getdata(ai);
   subplot(211)
   plot(data(257:end,1))
   set(gca,'YLim',[-0.002 0.002]), grid on
   subplot(212)
   plot(data(257:end,2));
   set(gca,'YLim',[-0.002 0.002]), grid on
   ```

6. The top plot shows the bipolar derivation of channel 1. The signal is almost zero because channel 2 was subtracted. On the bottom plot the input sine wave signal of channel 2 is visible.



7. Clear the analog input object

```
delete(ai);
clear ai
```

Common Ground and Reference

g.USBamp has 4 potential separated groups with 4 channels each. The ground and reference potentials can be connected to a common ground and common reference. This is useful if e.g. 16 EEG channels should be acquired. On the other hand it is useful to disconnect the groups for acquiring 4 EEG channels on the first group, 4 EMG channels on the second group,…

You can see that in the example the properties GroupAToCommonGround and GroupAToCommonReference are enabled. This connects ground sockets and reference sockets of group A to the common ground and reference. Perform the same commands for groups B, C and D. Additionally the property Mode is set to Calibration in order to generate an internal sine wave calibration signal. See program `gUSBampAPIDemo2.m`.

1. Define the analog input object for g.USBamp

   ```
   ai = analoginput('guadaq',1);
   addchannel(ai,[1:16]);
   ```

2. Set the sampling frequency to 256 Hz

   ```
   set(ai,'SampleRate',256,'SamplesPerTrigger',256);
   ```

3. Connect the ground sockets and reference sockets of groups A, B, C and D to common ground and reference.

   ```
   set(ai,'GroupAToCommonGround','Enabled',…
                  'GroupAToCommonReference','Enabled');
   set(ai,'GroupBToCommonGround','Enabled',…
                  'GroupBToCommonReference','Enabled');
   set(ai,'GroupCToCommonGround','Enabled',…
                  'GroupCToCommonReference','Enabled');
   set(ai,'GroupDToCommonGround','Enabled',…
                  'GroupDToCommonReference','Enabled');
   ```

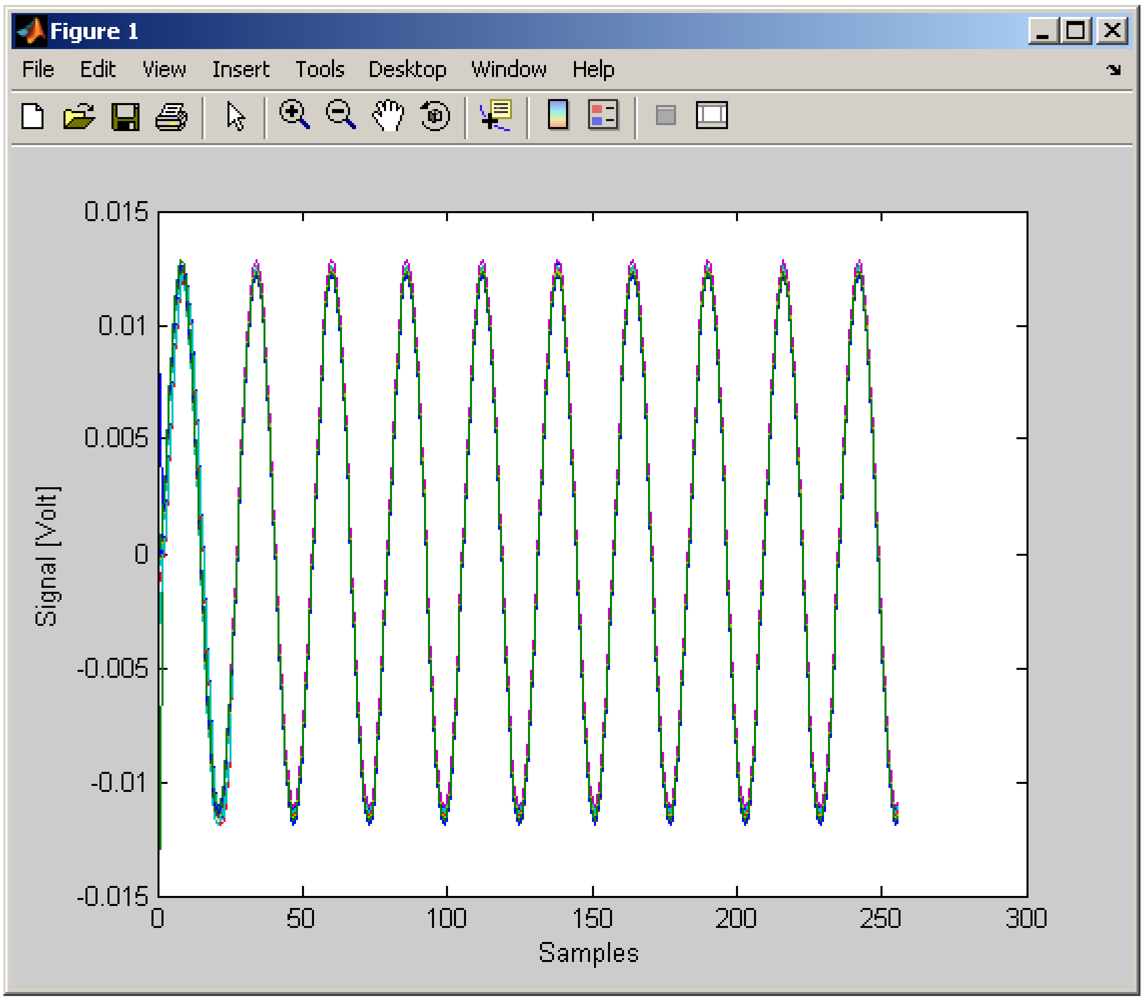
4. Generate a sine wave signal in calibration mode

   ```
   set(ai,'Mode','Calibration');
   ```

5. Start the data acquisition and acquire 1 second of data of all 16 channels. Plot the 16 channels.

   ```
   start(ai)
   while strcmp(ai.running,'On')==1
   end
   data=getdata(ai,ai.SamplesAvailable);
   plot(data);
   xlabel('Samples');
   ylabel('Signal [Volt]');
   ```

Figure 1 displays the 16 acquired sine wave calibration signals overlaying each other.



6. Clear the analog input object

```
delete(ai);
clear ai
```

Test Signal

g.USBamp can generate different test signals. Perform example `gUSBampTestSignal.m` to generate a sawtooth signal.

1. Open the g.USBamp device and add one channel

   ```
   ai = analoginput('guadaq',1);
   addchannel(ai,[1]);
   ```

2. Set the sampling rate to 256 Hz and acquire one second

   ```
   set(ai,'SampleRate',256,'SamplesPerTrigger',256);
   ```

3. Set the calibration amplitude to `2000`, the frequency of the signal to `10` Hz, the offset to `2047` and the waveshape to `Sawtooth`.

   ```
   set(ai,'AOAmplitude',2000,'AOFrequency',10,...
           'AOOffset',2047,'AOWaveShape','Sawtooth');
   ```

4. Disable the bandpass filter to see the sawtooth unfiltered

   ```
   set(ai.Channel(1),'BPIndex',-1);
   ```

5. Disable also the Notch filter

   ```
   set(ai.Channel(1),'NotchIndex',-1);
   ```

6. Set the mode to `calibration` to read in the signal from the calibration unit on channel 1

   ```
   set(ai,'Mode','Calibration');
   ```

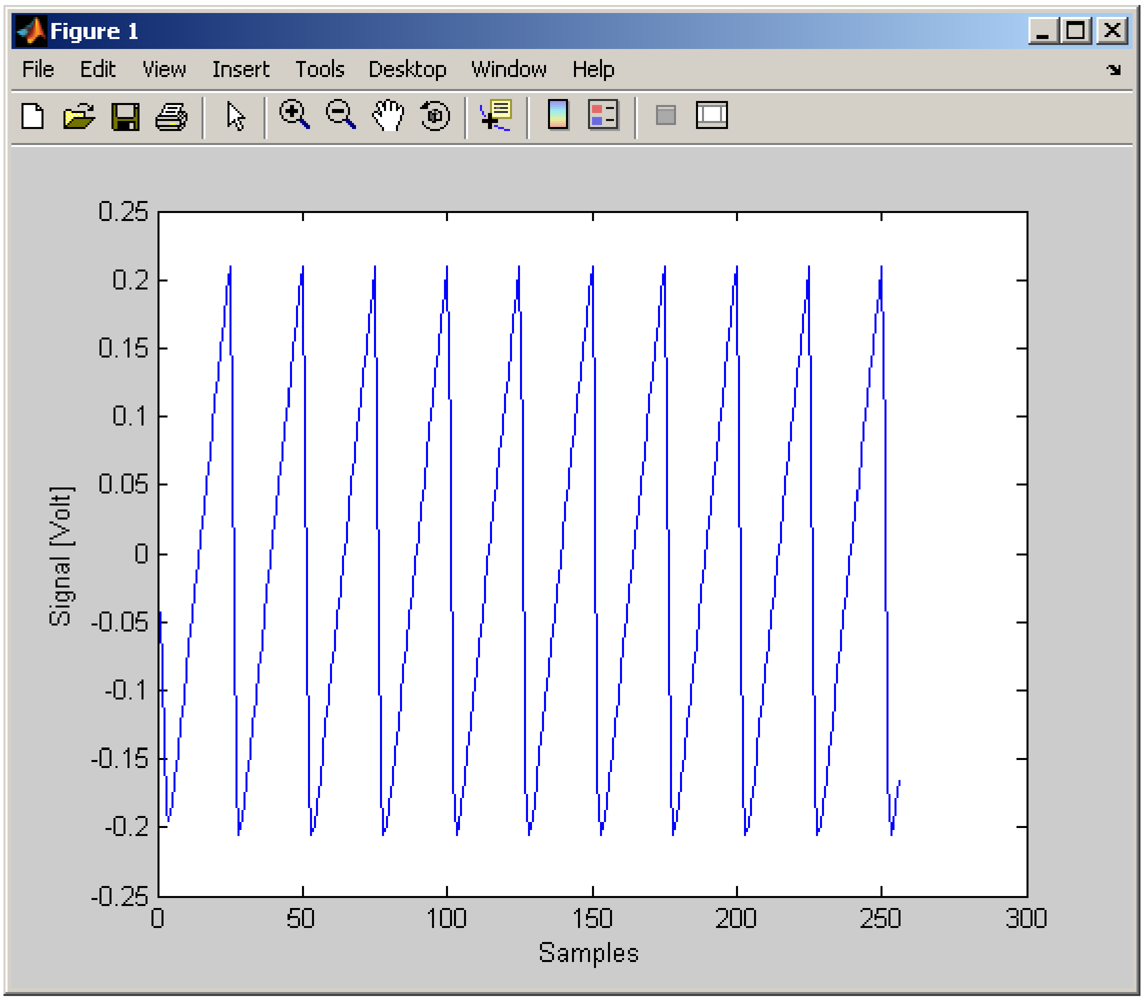
7. Start the acquisition and acquire one second of data

   ```
   start(ai)
   while strcmp(ai.running,'On')==1
   end
   ```

8. Finally plot the data

   ```
   data=getdata(ai,ai.SamplesAvailable);
   plot(data);
   xlabel('Samples');
   ylabel('Signal [Volt]');
   delete(ai);
   clear ai;
   ```

The Figure displays exactly 10 periods of a sawtooth generated by the internal calibration unit of g.USBamp.



Data Storage

g.USBamp allows to store data in MATLAB daq format (see the MATLAB Data Acquisition Toolbox manual). Perform the example `gUSBampDataStorage.m` to store one channel acquired with g.USBamp.

1. Open the g.USBamp device and add one channel

   ```
   ai = analoginput('guadaq',1);
   addchannel(ai,[1]);
   ```

2. Set the sampling rate to 256 Hz and acquire one second

   ```
   set(ai,'SampleRate',256,'SamplesPerTrigger',256);
   ```

3. Set the mode to calibration

   ```
   set(ai,'Mode','Calibration');
   ```

4. Define the filename for logging the data to harddisk. LoggingMode is set to Disk in order to stream the data to the harddisk. Use `Disk&Memory` to stream the data to harddisk and to memory. `LogToDiskMode` is set to `Overwrite` to replace an existing file with the same name.

   ```
   set(ai,'LogFileName','test','LoggingMode','Disk',…
             'LogToDiskMode','Overwrite');
   ```

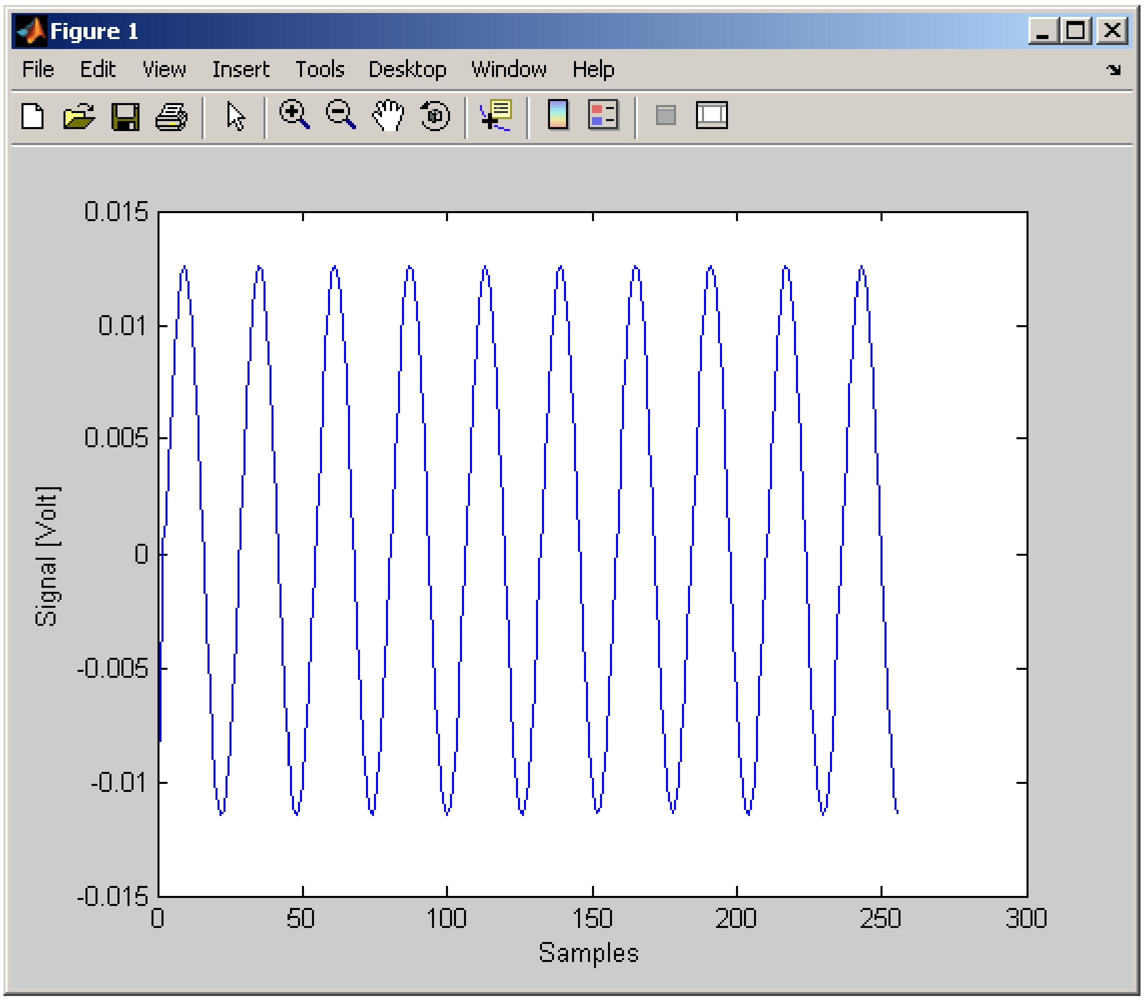
5. Start the acquisition and log one second of data. Then clear the analog input object.

   ```
   start(ai)
   while strcmp(ai.running,'On')==1
   end
   delete(ai);
   clear ai
   ```

6. Load the acquired data from file test.mat into the MATLAB workspace and plot the data.

   ```
   data=daqread('test');
   plot(data);
   xlabel('Samples');
   ylabel('Signal [Volt]');
   ```

The Figure shows the calibration signal that was streamed to the harddisk.



7. Read in the data acquisition file and return a structure which contains the g.USBamp settings.

   ```
   data=daqread('test','info');
   ```

   To obtain the properties of the channels from the file `test.daq` use

   ```
   channelinfo=data.ObjInfo.Channel
   ```

Calibration

The g.USBamp calibration function adjusts the magnification and offset of each channel.

Perform the following command and pass in the g.USBamp serial number

```
[offset, scaling] = gUSBampCalibration('UA-2006.01.05',false);
```

Please wait until the command returns `offset` and `scaling` which contain the calibration vectors.

Use `gUSBampSaveCalibration(offset, scaling, 'UA-2006.01.05')` to save the calibration vectors for this amplifier. The values are stored in permanent memory of g.USBamp.

Impedance Measurement

g.USBamp has an internal impedance measurement unit. The impedance of each electrode is measured against the ground electrode.

Call the function from the MATLAB command line with the appropriate g.USBamp serial number.

```
[z] = gUSBampImpedance('UA-2006.01.05');
```

`z` returns the impedance values. The first 16 values correspond to the analog input channels 1-16 of g.USBamp. The last 4 values correspond to the reference electrodes.

Type z into MATLAB to see the impedance values in kOhm. The example shows on channel 13 a 15 kOhm impedance.

```
z =

       85723
       85507
       84640
       83995
       85771
       84207
       83969
       85579
       85580
       84680
       85332
       85484
          15
       86212
       86751
       86886
       87515
       71464
       96323
      103038
```

Trigger, Digital Inputs and Outputs

g.USBamp version 2.0 has one trigger line which is sampled synchronously with the analog channels and can be acquired as 17th analog channel. g.USBamp 2.0 also has two asynchronous digital inputs and two digital outputs which can be used in MATLAB. See program `gUSBampDIO.m`.

g.USBamp version 3.0 has eight trigger lines which are sampled synchronously with the analog channels and can be acquired as 17th analog channel (coded into the analog channel). g.USBamp 3.0 also has 4 digital outputs, which can be used similar to g.USBamp version 2.0. To use the digital outputs replace all ‘in’ in `hwlines = addline(dio,[0 1 2 3],{'out';'out';'in';'in'});` with ‘out’.

1. Define a digital I/O object with g.USBamp Id 1

   ```
   dio = digitalio('guadaq',1);
   ```

2. Define 2 digital channels as outputs and 2 digital channels as inputs

   ```
   hwlines = addline(dio,[0 1 2 3],{'out';'out';'in';'in'});
   ```

3. Use `daqhwinfo` to investigate the digital I/O object parameters

   ```
   hwinfo = daqhwinfo(dio)

   hwinfo =

                AdaptorName: 'guadaq'
                 DeviceName: 'g.USBamp'
                         ID: '1'
                       Port: [1x2 struct]
              SubsystemType: 'DigitalIO'
                 TotalLines: 4
    VendorDriverDescription: 'g.USBamp Hardware Driver'
        VendorDriverVersion: 'Version 3.00'
   ```

4. Create a loop and use `putvalue` to toggle the digital outputs every second. Use `getvalue` to read in the digital inputs.

   ```
      for i=1:10
          putvalue(dio.Line(1),0);
          putvalue(dio.Line(2),1);
          in1=getvalue(dio.Line(3))
          in2=getvalue(dio.Line(4))
          pause(1),
          putvalue(dio.Line(1),1);
          putvalue(dio.Line(2),0);
          pause(1)
      end
   ```

5. Delete the digital I/O object

   ```
   delete(dio);
   clear dio
   ```

Synchronization of Two g.USBamps

The g.USBamp API allows to operate multiple g.USBamps synchron. For this reason one g.USBamp must be the master device and all other g.USBamps are slave devices. See example `gUSBampAPISync.m` for an example to synchronize two g.USBamps.

1. To check the number of connected amplifiers perform the following command:

   ```
   daqhwinfo('guadaq')
   ans =
              AdaptorDllName: 'C:\Program
   Files\gtec\gUSBampMatlabAPI\Lib\guadaq.dll'
           AdaptorDllVersion: '5.09a'
                 AdaptorName: 'guadaq'
                  BoardNames: {'g.USBamp'  'g.USBamp'}
           InstalledBoardIds: {'1'  '2')
       ObjectConstructorName: {2x3 cell}
   ```

2. Connect the synchronization cable to SYNC OUT of the master device and to the SYNC IN of the slave device.

3. Define an analog input object of device 1 and for device 2

   ```
   a1 = analoginput('guadaq',1);
   a2 = analoginput('guadaq',2);
   ```

4. Extract the serial number of both devices in order to identify the master

   ```
   Serial1 = a1.DeviceSerial
   Serial2 = a2.DeviceSerial
   ```

5. Set the device where the cable is connected to SYNC IN to slave mode

   ```
   set(a2,'SlaveMode','on');
   ```

6. Add 16 analog input channels to each device, set the sampling rate to 256 Hz and acquire 2 seconds of data. Set the devices to calibration mode in order to apply a sine wave signal to all inputs.

   ```
   addchannel(a1,[1:16]);
   addchannel(a2,[1:16]);
   set(a1,'SampleRate',256,'SamplesPerTrigger',512);
   set(a2,'SampleRate',256,'SamplesPerTrigger',512);
   set(a1,'Mode','Calibration');
   set(a2,'Mode','Calibration');
   ```

7. Set the bandpass filters for all 16 channels of both devices and start the data acquisition

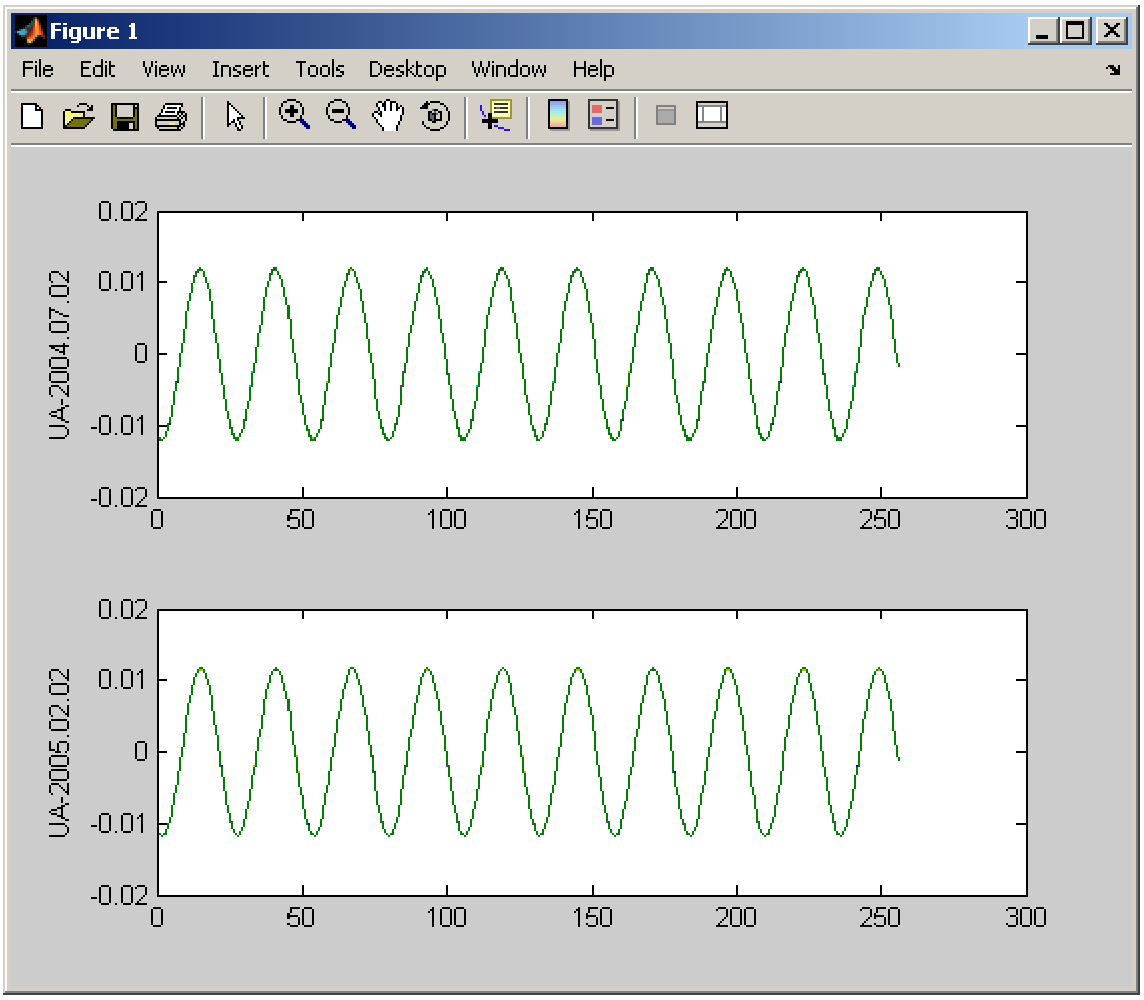
   ```
   for i=1:16
       set(a1.Channel(i),'BPIndex',48);
       set(a2.Channel(i),'BPIndex',48);
   end
   start(a2)
   start(a1)

   while strcmp(a1.running,'On')==1
   end
   ```

8. Extract the data from both devices and plot it

```
data1 = getdata(a1);
data2 = getdata(a2);

subplot(211);
plot(data1(257:end,:));
ylabel(Serial1)
subplot(212);
plot(data2(257:end,:));
ylabel(Serial2)
```



The top row shows the acquired data of g.USBamp with serial number UA-2004.07.02 and the bottom row of device UA-2005.02.02.

9. Delete the analog input object of both devices

```
delete(a1);
clear a1
delete(a2);
clear a2
```

Synchronization of Four g.USBamps

The preceding example showed the g.USBamp MATLAB API working with two amplifiers. In this example the synchronization of four g.USBamps is shown. See example `gUSBampAPISync4amps.m` for an example to synchronize four g.USBamps.

1. Type `daqhwinfo('guadaq')` to see the actual number gUSBamps connected to the PC.

   ```
   daqhwinfo('guadaq')
   ans =
              AdaptorDllName: 'C:\Program
   Files\gtec\gUSBampMatlabAPI\Lib\guadaq.dll'
           AdaptorDllVersion: '5.09a'
                 AdaptorName: 'guadaq'
                  BoardNames: {'g.USBamp'  'g.USBamp'  'g.USBamp'
   'g.USBamp'}
           InstalledBoardIds: {'11'  '12'  '13'  '14')
       ObjectConstructorName: {2x3 cell}
   ```

2. The example `gUSBampAPISync4amps` is split in different sub functions. To initialize the hardware type `gUSBampAPISync4amps('init')` into the MATLAB command window. This function sets the analog input objects needed for each amplifier, the sampling frequency, filter and defines one device as master and three slave devices. It displays the serial numbers of the connected amplifiers and the master device in the MATLAB command window.

   Data will be stored for each g.USBamp in the preset DAQ-files 'mydatafile1.daq' to 'mydatafile4.daq'. Use MATLAB function `daqread` to view the files in MATLAB.

   Sampling frequency is set to `600` Hz, because for higher frequencies the `Calibration` signal cannot be used.

3. To start data acquisition type `gUSBampAPISync4amps('start')` and acquired data will be plotted in a figure.

4. Stop the data acquisition with `gUSBampAPISync4amps('stop')` and clean the workspace with `gUSBampAPISync4amps('cleanup').`

g.USBamp Commands

`gUSBampCalibration`	calibrate g.USBamp
`gUSBampImpedance`	measure the electrode impedance
`gUSBampSaveCalibration`	save the calibration values
`gUSBampShowFilter`	show possible bandpass and notch filters

gUSBampCalibration

DESCRIPTION: Get Offset and Scaling values for gUSBamp.

Call it with: `[offset, scaling] = gUSBampCalibration(ID,verbose)`

INPUT:
`ID`...amplifiers serial ID can be retrieved from device top label (string)

`verbose`...display progress information in command line (boolean)

OUTPUT:
`offset`...vector of offset voltages in µV (16x1)
`scaling`...vector of scaling factors (16x1)

NOTE: Values are not stored. Use `gUSBampSaveCalibration` to store values.

EXAMPLE:

```
verbose = true;
ID ='UA-2006.01.05';
[offset, scaling] = gUSBampCalibration(ID,verbose);
```

gUSBampImpedance

DESCRIPTION: Measure the electrode impedances connected to gUSBamp.

Call it with: `impedance = gUSBampImpedance(ID,verbose)`

INPUT:
`ID`...amplifiers serial ID can be retrieved from device top label (string)
`verbose`...display progress information in command line (boolean)

OUTPUT:
`impedance`...vector of impedances

NOTE: `impedance(1:16)` are the impedance values for channels 1 to 16
`impedance(17:20)` are the impedance values for references A, B, C, D

EXAMPLE:

```
verbose = true;
ID ='UA-2006.01.05';
impedance = gUSBampImpedance(ID,verbose);
```

gUSBampSaveCalibration

DESCRIPTION: Store Offset and Scaling values for gUSBamp.

Call it with: `gUSBampSaveCalibration(offset,scaling,ID)`

INPUT:
`offset`...offset voltages in µV (16x1)
`scaling`...scaling factors (16x1)
`ID`...amplifiers serial ID can be retrieved from device top label (string)

EXAMPLE:

```
offset = zeros(16,1);
scaling = ones(16,1);
ID ='UA-2006.01.05';
gUSBampSaveCalibration(offset,scaling,ID);
```

gUSBampShowFilter

DESCRIPTION: Show possible bandpass and notch filters for a specific sampling frequency.
Possible sampling frequencies are: 32, 64, 128, 256, 512, 600, 1200, 2400, 4800

Call it with: `gUSBampShowFilter(samplingfrequency)`

INPUT:
`samplingfrequency`...specify the sampling rate

EXAMPLE:

```
gUSBampShowFilter(256);
```

Analog Input Properties

SampleRate

Sets the sampling frequency of g.USBamp. The sampling frequency is common for all analog input channels. The sampling frequency value must correspond to a value defined in the table below. The oversampling rate depends directly on the selected sampling frequency. A small sampling frequency will result in a higher oversampling rate.

The possible sampling rates and the corresponding oversampling rates are shown in the following table:

`SampleRate` [Hz]	Oversampling Rate [Scalar]
32	1200
64	600
128	300
256	150
512	75
600	64
1200	32
2400	16
4800	8

Example:

Set the sampling frequency to 256 Hz of analog input object `a1`.

```
set(a1,'SampleRate',256);
```

Calibration

Define the settings of the internal calibration unit.

`AOAmplitude`	max: 2000 (250mV), min: 0, default : 1000	
`AOFrequency`	frequency in Hz, default: 10 Hz	
`AOOffset`	no offset: 2047, min: 0, max: 4096, default: 2047	
`AOWaveShape`	[`Square | Sawtooth | {Sine} | Noise`]	
	`Square`	generate square wave signal
	`Sawtooth`	generate sawtooth signal
	`Sine`	generate sine wave
	`Noise`	generate white noise

Note that `Mode` must be set to `Calibration` to generate the calibration signal. The electrodes on the amplifier input are disconnected internally in calibration mode.

Property `AOFrequency` is not required for the `AOWaveShape Noise`.

Example:

Set the calibration amplitude to `2000`, the frequency of the signal to `10` Hz, the offset to `2047` and the waveshape to `sawtooth`. This will result in a calibration signal of ± 250 mV.

```
set(ai,'AOAmplitude',2000,'AOFrequency',10,...
          'AOOffset',2047,'AOWaveShape','Sawtooth');

set(ai,'Mode','Calibration');
```

Counter

The property counter allows to generate on channel 16 a counter signal. Whenever g.USBamp is sending a data package to the PC the counter is increased by one. This function can be used to check if the USB connection between g.USBamp and your PC is working properly.

`Counter`	[`On`│`Off`]	enable or disable the counter

Note that channel 16 can not be used as input channel if it is used as counter channel.

Example:

Enable the `Counter` to generate on channel 16 the counter signal

```
set(ai,'Counter','on');
```

Device Serial

Read the serial number of the g.USBamp. Each g.USBamp has a unique serial number which can be used to identify the device. This is important when multiple devices are used. If one g.USBamp is the first time connected to your PC it is registered in the operating system and gets a specific Id in MATLAB. If the device is re-connected to your PC it will receive the same Id in MATLAB.

Example

Read the serial number of g.USBamp with Id 1.

```
ai=analoginput('guadaq',1)
get(ai,'DeviceSerial')
```

returns

```
UA-2004.07.02
```

Bipolar

Define the hardware channels for a bipolar derivation. The bipolar derivation is performed on g.USBamp before bandpass filtering the channels.

Example

To perform a bipolar derivation between channel 1 and 2 use the following command.

```
set(ai.Channel(1),'Bipolar',2);
```

This will subtract channel 2 from channel 1.

Common Ground and Reference

Connect or disconnect the grounds and references of the 4 groups (A, B, C, D).
If the property is enabled the ground or reference is connected to the common ground or reference.
Each combination is allowed.

```
GroupAToCommonGround [ Enabled | {Disabled} ]
GroupBToCommonGround [ Enabled | {Disabled} ]
GroupCToCommonGround [ Enabled | {Disabled} ]
GroupDToCommonGround [ Enabled | {Disabled} ]

GroupAToCommonReference [ Enabled | {Disabled} ]
GroupBToCommonReference [ Enabled | {Disabled} ]
GroupCToCommonReference [ Enabled | {Disabled} ]
GroupDToCommonReference [ Enabled | {Disabled} ]
```

Example:

Connect group A and B to common ground and reference.

```
set(ai,'GroupAToCommonGround','Enabled',…
                  'GroupAToCommonReference','Enabled');
set(ai,'GroupBToCommonGround','Enabled',…
                  'GroupBToCommonReference','Enabled');
```

Mode

g.USBamp has three operation modes:

`Normal`	measure biosignals with electrodes on all 16 input channels
`Impedance`	measure the impedance of the connected electrodes
`Calibration`	generate a calibration signal on all input channels

Note that in `Calibration` mode the electrodes are internally disconnected from g.USBamp.

```
Mode         [ {Normal} | Impedance | Calibration]
```

Example

```
set(ai,'Mode','Calibration');
```

Shortcut

g.USBamp has a shortcut connector at the rear side. While a TTL High is applied to this input the analog inputs of g.USBamp are connected to ground and the electrodes are disconnected from the inputs. A LOW connects the input sockets to the amplifier channels. To enable or disable this functionality use the shortcut property.

```
Shortcut    [ Enabled | {Disabled} ]
```

Example

Enable the Shortcut feature.

```
set(ai,'Shortcut','Enable');
```

Synchronization

Set the amplifier to slave/master mode. If multiple amplifiers are used one amplifier must be the master device.

```
SlaveMode         [ On | {Off} ]
```

Example:

The master device must be set to

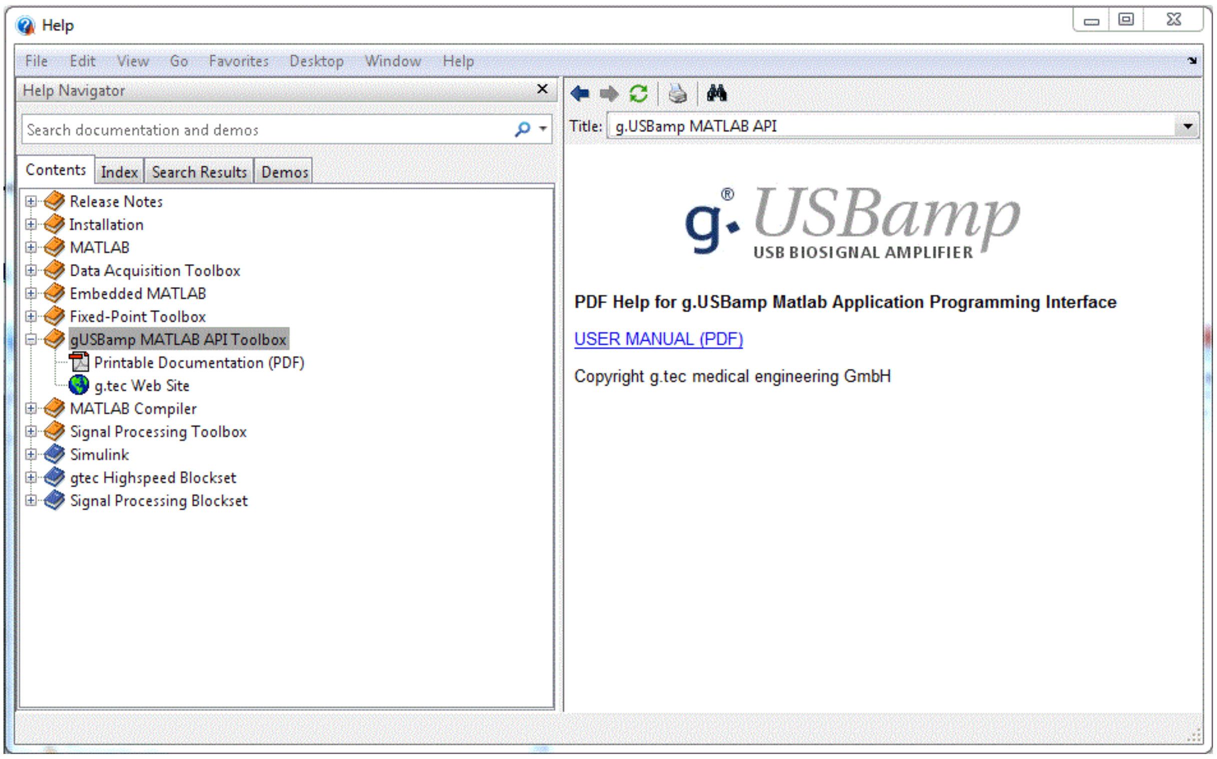
```
set(ai,'SlaveMode','Off');
```

all other devices must be set to

```
set(ai,'SlaveMode','On');
```

Help

g.USBamp MATLAB API provides a printable documentation.


Help
File Edit View Go Favorites Desktop Window Help
Help Navigator
Search documentation and demos
Contents Index Search Results Demos
Release Notes
Installation
MATLAB
Data Acquisition Toolbox
Embedded MATLAB
Fixed-Point Toolbox
gUSBamp MATLAB API Toolbox
Printable Documentation (PDF)
g.tec Web Site
MATLAB Compiler
Signal Processing Toolbox
Simulink
gtec Highspeed Blockset
Signal Processing Blockset
Title: g.USBamp MATLAB API
g.® USBamp
USB BIOSIGNAL AMPLIFIER

PDF Help for g.USBamp Matlab Application Programming Interface

USER MANUAL (PDF)

Copyright g.tec medical engineering GmbH

To access the help click on **Product Help** in the **Help** menu of MATLAB. To access the help from command line type:

```
doc
```

The printable documentation is stored under

```
Your Installation Folder\gtec\gUSBampMatlabAPI\Help
```

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

Product Page

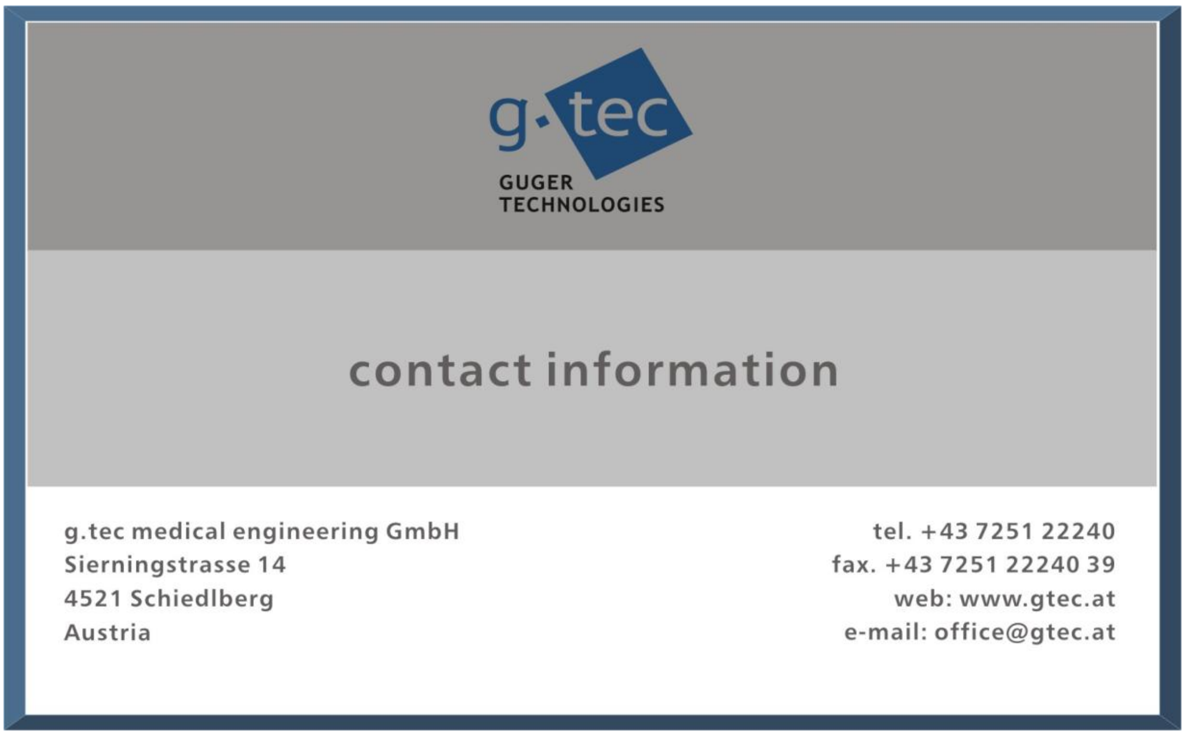
Please visit our homepage www.gtec.at for

• Update announcements

• Downloads

• Troubleshooting

• Additional demonstrations



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