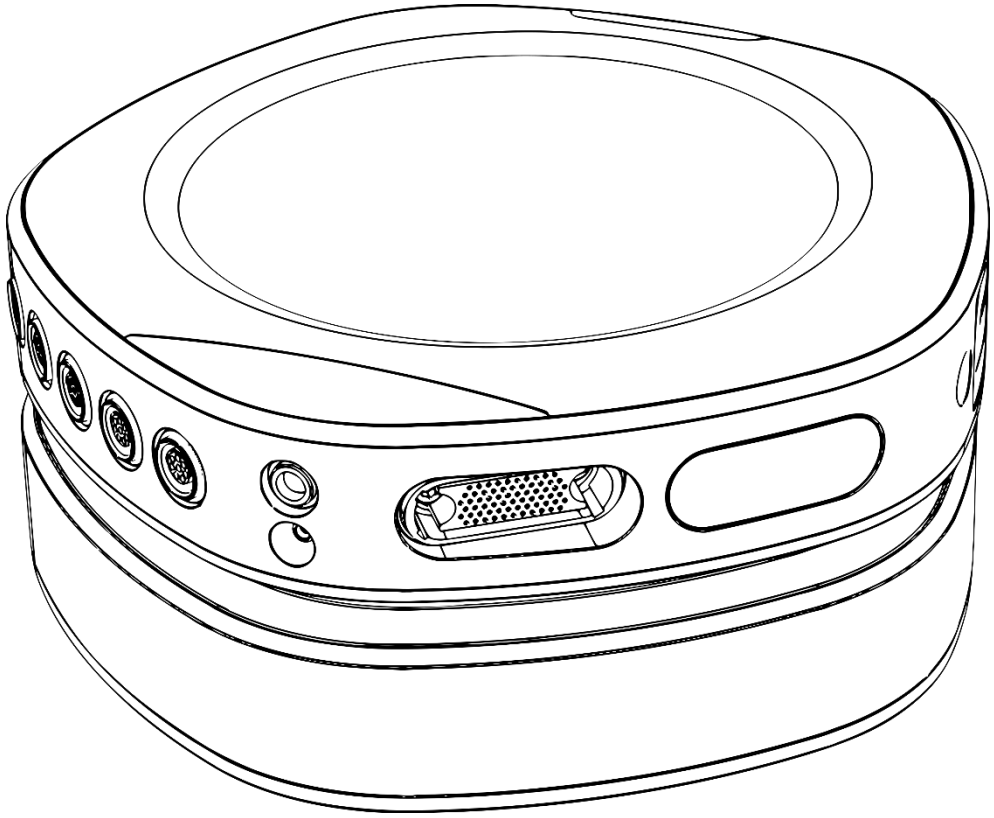
saga 32 64+



API DOCUMENTATION

TMSI SAGA INTERFACE FOR MATLAB

TMSi

REF: 93-2303-0004-EN-1-0

REV 1 **EN**

Last update: 2020-11-16

TABLE OF CONTENTS

The TMSi MATLAB Interface is distributed under MIT license, see license details for more information. This interface is part of the SoftPro Project that has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement No.688857

1 INTRODUCTION

1.1 About this document

This document describes the TMSi SAGA interface for MATLAB, what it does, how it should work and what you can and cannot expect from TMSi regarding this interface. Please read carefully through this document.

1.2 TMSi SAGA interface for MATLAB

The TMSi SAGA interface for MATLAB is a library for MATLAB written by TMSi to interface the SAGA Device Driver to MATLAB. This interface allows you to acquire data, write and read .poly5 files via MATLAB and/or export data to EEGLAB (http://sccn.ucsd.edu/eeglab/).

Functionality that is included in the TMSi SAGA interface for MATLAB is limited to the signal acquisition part of the driver and loading data from previously recorded data in TMSi Polybench or the TMSi SAGA interface for MATLAB. Data repair is supported to repair sample loss in wireless measurements. All functionality needed to perform on-board memory recordings (so-called ambulatory measurement) is **NOT** implemented.

The TMSi SAGA interface for MATLAB is mostly similar to the TMSi MATLAB interface for older TMSi devices. Differences between both interfaces can be found in chapter 4. Differences between the initial release of the TMSi SAGA interface for MATLAB (v1) and the current release of the TMSi SAGA interface for MATLAB (v2) can be found in chapter 5.

1.3 License and Support

The TMSi SAGA interface for MATLAB is **free of charge** and distributed under MIT License. In short this means that you can do what you want with the MATLAB interface. The complete text of the MIT license is included in the LICENSE.txt in the package, as well as in this document.

TMSi has tested the interface, but cannot guarantee that it works under every circumstance, let alone that it will function in the experimental setup that you have in mind.

1.4 TMSi Support

The TMSi SAGA interface for MATLAB can be downloaded from our TMSi website under ‘Support’ (www.tmsi.com). We cannot help you write MATLAB code, but any questions about (using) the TMSi SAGA interface for MATLAB can be asked via support@tmsi.com.

1.5 TMSi Polybench or the TMSi SAGA interface for MATLAB?

TMSi Polybench is a toolbox-based software package which we continue to support and sell. The TMSi SAGA interface for MATLAB does not replace TMSi Polybench.

Researchers who want to easily measure signals in a simple graphical environment, where you can create your own measurement configurations using symbolic programming without coding or scripting skills: TMSi Polybench is the way to go. TMSi Polybench is licensed software. Please contact sales@tmsi.com to obtain a quote.

The TMSi SAGA interface for MATLAB can be used by engineers and researchers who are familiar or learning to become familiar with use of MATLAB. The TMSi SAGA interface for MATLAB is easy to understand and documented with comments in the code. You probably do not have to change the more complex aspects of the implemented functions of the SAGA Device Driver. More information about scripting for SAGA can be found in the TMSi Device API User Manual.

2 DESCRIPTION OF THE TMSI SAGA INTERFACE FOR MATLAB

2.1 Features

The code in the TMSi SAGA interface for MATLAB is a MATLAB library for sampling TMSi SAGA devices. The goal of this library is to provide an easy and intuitive way of accessing the TMSi devices from MATLAB to use in your own experiments.

The library provides the following functionality:

- Sampling over USB or network.
- Sampling over electrical or optical interface
- Sampling over wireless connection and retrieving lost samples afterwards via Data Repair.
- (Limited) live plotting of sampled data.
- Sampling in impedance mode and (limited) live plotting of impedance values.
- Limited live plotting of an HD EMG heat map (if MATLAB's Signal Processing Toolbox is available).
- Directly saving sampled data to Poly5 file format.
- Offline export of .Poly5 data to EEGLAB.

The library has been tested on the following MATLAB version:

- MATLAB 2020b (Windows 10 / 64bit)

And should work for newer versions of MATLAB. There is no support for 32-bit systems and no support for earlier versions than MATLAB 2016a.

2.2 Installation requirements

The TMSi SAGA interface for MATLAB requires:

- A computer with Windows 10, 64-bit.
- SAGA Device Driver 2.0.0.
- MATLAB release 9.

Most functionalities of the TMSi SAGA interface for MATLAB do not require special toolboxes, although the Signal Processing Toolbox is highly recommended.

2.3 Installation

When using the MATLAB library, ensure that the TMSi SAGA Device Driver has been successfully installed. Before using the SDK, copy the `+TMSiSAGA/` folder to your working directory and reference the classes as `TMSiSAGA/<class>`.

2.4 Usage

To start using the SDK, you have to initiate the SDK by creating a `TMSiSAGA.Library()` object. Next you can retrieve a TMSi SAGA Device over a specific interface using `library.getFirstAvailableDevice`. From here on you will first have to call `device.connect()` to connect to the device, before you can start sampling with a call to `device.start()`. You will always need to call `device.stop()` to stop sampling and `device.disconnect()` to disconnect from the device. Any changes to the configuration of the device have to be done after the device is connected and before it is sampling.
If you run in a situation where there is a device still sampling or connected but you no longer have the device object, you can use the function `library.cleanUp()` to stop and disconnect all devices.

2.5 Recommendations

A few recommendations and things to keep in mind when using this code.

- Always enclose your code in a `try catch end` block, with a call to `library.cleanUp()` in the catch block, to ensure that all open and sampling devices are closed properly in case of an error.
- `TMSiSAGA.Data` object is not very efficient with memory, because it will append data and extend the array continuously. If you are measuring for extended period of times and you want to save the data, use the 'TMSiSAGA.Poly5' class.
- Limit the channels that are shown in a `RealTimePlot`
- Limit the window size of `RealTimePlot` to ~30 seconds.
- When changing data recorder interface, make sure you wait before reconnecting again. It might take a while for the device to properly setup the new interface connection.
- Make sure the window size of a `Visualisation` object is chosen sufficiently large to perform the online processing that is needed to create the HD EMG heat map.

- The HD EMG heat map should be used over either the electrical or optical interface. The applied high-pass filter is not able to cope with the situation when samples are missed over a wireless connection. When this happens, data collection continues, but the visualisation disappears.

2.6 Content

The library contains the following:

+TMSiSAGA/

Contains all the SDK files. Should be copied to the directory from where you want to use the TMSi SAGA MATLAB SDK.

+TMSiSAGA/Channel

Class that represents a channel of a SAGA device, contains information and transformation logic.

+TMSiSAGA/Data

Class that represents data retrieved from the devices. Class can be used as a storage for short measurements (~10 minutes) and is also used to read/write to and from Poly5 files. This class also provides means to export data to EEGLAB.

+TMSiSAGA/DataRecorderInfo

Class that contains information about the data recorder. Information as interface type, serial number, temperature, etc.

+TMSiSAGA/Device

Class that represents a single TMSiSAGA device. This class contains all functions to operate the device with, from configuration to sampling.

+TMSiSAGA/DeviceLib

Class that contains calls to the TMSiSagaDeviceLib.dll functions. It hides some of the specifics from the .dll functions in MATLAB. Internal class that should not be used directly.

+TMSiSAGA/DockingStationInfo

Class that contains information about the docking station like serial number, temperature, etc.

+TMSiSAGA/HiddenHandle

This file can be ignored as it only reduces clutter in documentation.

+TMSiSAGA/ImpedancePlot

Class that visualises the skin-electrode impedance measurement. This class offers both a grid lay-out and head lay-out visualisation of the impedance measurement.

+TMSiSAGA/Library

Class that is the entry point of the SDK. From here you can ask for a list of devices and it keeps track of all open and sampling devices.

+TMSiSAGA/Poly5

Class that provides means to stream data to and store the data in Poly5 files. It also allows offline reading of Poly5 files. Poly5 files can be made using TMSi Polybench or the TMSi SAGA Interface for MATLAB.

+TMSiSAGA/RealTimePlot

Class that allows a basic real-time plotting of sampled data. The `RealTimePlot` adds data and shows it in real-time. Closing the figure used for plotting will also close the connection to the device. The `RealTimePlot` allows user input when you press ‘a’ (AutoScale) or ‘q’ (Quit).

+TMSiSAGA/Repair

Class that contains logic to repair sampled data. This class contains functions to fill in the missing samples with the data retrieved from the device.

+TMSiSAGA/Sensor

Class that represents a TMSi Sensor. It keeps track of all sensors that are retrieved from the TMSi device. This class is not directly correlated to a sensor channel. It is for internal use.

+TMSiSAGA/SensorChannelDummy

Class that represents a dummy sensor channel. It performs no changes to the sampled data.

+TMSiSAGA/SensorChannelType0

Class that represents a Type0 sensor channel. It will perform a transformation of the sampled data automatically before returned by the `sample()` function.

+TMSiSAGA/TMSiUtils

Class that contains some utility functions, for internal use only.

+TMSiSAGA/Visualisation

Class that provides means to visualise an HD EMG measurement as a heat map of the muscle activation. This class also has signal processing functionalities to ensure the heat map can be visualised correctly. Use of this class requires access to MATLAB's Signal Processing Toolbox.

3 EXAMPLES

The easiest way to start using the interface is to follow the examples. With the examples you will be guided through the main features of the TMSi SAGA interface for MATLAB. Connect a TMSi SAGA device to the PC as you would normally do and follow the steps on screen. Please check the communication interfaces, most examples assume the USB and electrical interface.

SampleElectrical.m

Example that shows how to sample data from a device, after setting the device configuration and channel configuration. Furthermore, the example shows how to save data to a Poly5 file and stream data to a `RealTimePlot` object.

ChangeDataRecorderInterface.m

The connection between docking station and data recorder can be established over multiple interfaces: (1) Docked ('electrical'), (2) Fiber ('optical') or (3) Wireless ('wifi'). This example can be used to reconfigure the data recorder interface.

SampleWifi.m

Example that shows how to sample data from a device over WiFi interface, after setting the device configuration and channel configuration. The example assumes that the data recorder interface is configured for WiFi measurements (run `ChangeDataRecorderInterface.m` example first). The example shows how to enable repair logging and how to perform a repair after samples have been missed.

SampleImpedanceVisual.m

SAGA devices support impedance measurements to verify that the patient connection is such that a reliable and high-quality measurement can be made. This example shows how to visualise the impedance measurement. The impedance can be plotted in a head or grid layout. The .mat-file *EEGChannels32/64TMSi.mat* is called when the head layout is chosen.

EEG_Workflow_Example.m

Example that shows a workflow of an EEG measurement. The example starts with an impedance measurement. The impedance measurement is stopped when the figure window is closed. Next, the example continues with a measurement that is plotted using a `RealTimePlot` object. Closing the window terminates the measurement.

EMG_Workflow_Example.m

Example that shows a workflow of an HD EMG measurement. The example starts with an impedance check. The impedance measurement is stopped when the figure window is closed. Next, a heat map of the muscle activation is shown. Missing channels are marked red, as this could indicate that the connection between electrode and skin are not good.

Poly5toEEGlab.m

This example uses the Poly5 reader from the library to load a previously recorded file. After reading the data, it is converted into EEGLAB format and saved as EEGLAB readable dataset. The .mat-file *EEGChannels32/64TMSi.mat* is called to assign the electrode locations used by EEGLAB.

resetFactoryDefault.m

This example shows how to reset the configuration to the factory default configuration.

4 MIGRATE FROM TMSI TO TMSISAGA SDK

With the TMSi SAGA device, a new SDK is created to accompany it. This chapter describes the changes with respect to the previous version.

Removed Sample class

The sampler class has been removed from the SDK. Instead, all configurations are directly stored in the `device` object and have to be synched with the actual device. The sync of configuration is no longer done automatically but requires a call to `device.updateDeviceConfig()`.

Handling missing samples

New functionality has been added to the SDK to help with missing samples, as the SAGA devices are more versatile in their use and can be used in environments where not all samples are guaranteed to be received by the Data Recorder. When repair logging is turned on, all samples are stored on the SAGA device and can be retrieved after sampling by calling `device.getMissingSamples()`.

New sensor channels

SAGA devices support sensor identification and a special object is added to support this. The sensor channels are created for each individual channel that has a sensor connected to it. These channels transform the sample data before returning it from the `device.sample()` call. If a custom sensor is used, a custom sensor channel can be made and has to be added to the sensor creation function in the `Sensor.m` file.

Poly5 file counter channel

Due to the nature of the Poly5 file, the counter channel in a Poly5 file is reset every 2^23 samples.

Sample rate availability

The sample rate is not directly retrieved from the device, but has to be calculated based on the settings of the device. For this reason, the sample rate of a device can change depending on what divider is used for a channel type. The `device.sample_rate` property contains the current sample rate of the device and is updated after every `device.updateDeviceConfig()` call. The sample rate of the counter channel corresponds to the channel type with the highest sample rate.

Initialization of the Libary

The new library does not require an interface, however to retrieve devices you will now have to specify which interfaces have to be used. The possible interfaces are `'usb'` or `'network'` for connection between PC and docking station and `'electrical'`, `'optical'` or `'wifi'` for connection between docking station and data recorder.

Library cleanUp

It is now easier to recover and close all devices in case of an error during sampling or configuration. The function `library.cleanUp()` will stop and close all open devices that were created. This should reduce hanging of devices and allow for easier closing of devices in case of errors. A good practice is to do the following:

```
library = TMSiSAGA.Library();


try
    device = library.getFirstAvailableDevice();
    device.connect();
    device.start();
    % ==== your code here ====
    device.stop();
    device.disconnect();
catch e
    library.cleanUp();
end
```

4.1 Examples (TMSi to TMSiSAGA)

Initialize Library

You do not have to specify the interface for the library, but you do have to specify where to look for devices when retrieving devices. The list of devices returned are already Device objects can be used immediately without having to first get the device.

TMSi SDK	TMSiSAGA SDK

`library = TMSi.Library('usb');`
 	
`device_list = library.refreshDevices();`
 	
`device = library.getDevice(device_list(1).name);`
	`library = TMSiSAGA.Library();`
	
	`devices = library.getDevices('network', 'electrical');`
	
	`device = devices(1);`

Get first available device

To get the first available device, you now have to specify on which docking station and data recorder interface you want to look. However, you no longer have to refresh the devices to be able to get the first device.

TMSi SDK	TMSiSAGA SDK

`library = TMSi.Library('usb');`
`device_list = library.refreshDevices();`
`device = library.getFirstDevice();`
	`library = TMSiSAGA.Library();`
	
	`device = library.getFirstAvailableDevice('network', 'electrical');`

Create a sampler

Sampler no longer exists and everything is done in the device object. To save settings you now need to explicitly call the `device.setDeviceConfig()` function.

TMSi SDK	TMSiSAGA SDK

`sampler = device.createSampler();`
`sampler.setSampleRate(4000);`
`sampler.connect();`
`sampler.start();`
 	
`% rest of sampling code`
	`device = library.getFirstAvailableDevice();`
	`device.setDeviceConfig(struct('BaseSampleRate',4000));`
	`device.connect();`
	`device.start();`
	
	`% rest of sampling code`

Sampling from device

Sampler has been removed and now you sample directly with the Device object.

TMSi SDK	TMSiSAGA SDK

`sampler.start();`
`samples = sampler.sample();`
`sampler.stop();`
	`device.start();`
	`samples = device.sample();`
	`device.stop();`

Error handling during sampling

The number of try catch statements required to properly handle errors during sampling is reduced.

TMSi SDK	TMSiSAGA SDK

try
    sampler = device.createSampler();
    try
        sampler.connect();
        sampler.start();
        % Error occurred
    catch e
        % log
    end
    sampler.stop();
    sampler.disconnect();
catch e
    % log
end
library.destroy();
	try
	    device.connect();
	    device.start();
	    % Error occurred
	    device.stop();
	    device.disconnect()
	catch e
	    library.cleanUp();
	end

5 MIGRATE FROM TMSISAGA SDK V1 TO TMSISAGA SDK V2

With the release of the new version of the TMSiSAGA interface for MATLAB, several changes are implemented that can affect your custom-made code. Most changes from TMSiSAGA SDK v1 to TMSiSAGA SDK v2 are done to improve stability and to introduce new features. The function `device.resetDeviceConfig()` is present in most examples of the TMSiSAGA SDK v1 release. However, as this function is intended as an error recovery mode, we strongly recommend to omit the use of this function in your custom-made code and have removed this function from the updated examples.

Improved WiFi stability

`pause` statements are removed from the code when a connection to a device is made over WiFi interface. This is done by adding a `numOfRetries` parameter to the `library.getFirstAvailableDevice()` function where you can state how many tries the device may take to find a device over WiFi.

Support for multi sample rate measurements

With the release of the SAGA v2 firmware, measuring at multiple sample rates for different channel types is supported. Therefore, dividers per channel type can now be configured.

Set the device configuration

Changes to the configuration of the device are made differently. A single `device.setDeviceConfig()` function is available that provides functionality to update the device configuration with one function call, instead of calling multiple functions that each change a single configuration setting. Every call to `device.setDeviceConfig()` includes a call to `device.updateDeviceConfig()` so that the device configuration is synced.

Set the channel configuration

Changing the active channel configuration can be done per type of channel. As a result, it is not necessary to use the channel number that is available in the device. Every call to `device.setChannelConfig()` includes a call to `device.updateDeviceConfig()` so that the channel configuration is synced.

Calculate the used bandwidth of the configuration

A function is added that supports the calculation of the bandwidth that is in use with the set device configuration. When the configured bandwidth exceeds the maximum bandwidth of the interface, a warning is given to the user. After updating the device configuration, the configured bandwidth is automatically calculated. Furthermore, the `device.getCurrentBandwidth()` function can be called directly.

Visualisation of an impedance measurement

The measured impedance values can be visualised online, allowing for more intuitive and clearer information on the skin-electrode connection. Both a head layout and grid layout can be chosen for the visualisation.

Visualisation of an HD EMG heat map

An HD EMG specific visualisation is made available to display the muscle activation online. This provides a spatiotemporal plotting functionality, as opposed to the `RealTimePlot` object that offers a temporal plotting functionality.

Export to EEGLAB

The export from .Poly5 files to EEGLAB compatible datasets is extended. The topographic channel locations of the measured channels are added when the .Poly5 file is exported. With the V2 firmware release of the SAGA, trigger support is available to the user. The single 16-bit Triggers channel is split into 16 event channels when exporting your .Poly5 file to an EEGLAB compatible dataset.

Data repair with triggers

With the V2 firmware release of the SAGA, trigger support is available to the user. The repair functionality retrieves samples stored on the Data Recorder's onboard memory. As triggers are recorded on the Docking Station, the repair functionality has been adapted to ensure that the Triggers channel is not overwritten during data repair.

5.1 Examples (TMSiSAGA SDK v1 to TMSiSAGA SDK v2)

Get first available device

The optional parameter `numOfRetries` can be added, which is needed by the device to find a connection over WiFi interface without using a `pause` statement.

TMSiSAGA SDK v1	TMSiSAGA SDK v2

`lib = TMSiSAGA.Library();`
 	
`device = lib.getFirstAvailableDevice( 'usb', 'wifi');`
 	
`disp('Wait for the device to be connected to wifi.');`
 	
`pause`
 	
`device.connect();`
 	
`% Rest of sampling code`
	`lib = TMSiSAGA.Library();`
	
	`device = lib.getFirstAvailableDevice( 'usb', 'wifi', 5);`
	
	`device.connect();`
	
	`% Rest of sampling code`

Setting device configuration

A single function to set the device configuration is available to the user, instead of separate functions to tune individual device configuration settings. With the v2 release of the SAGA interface for MATLAB, dividers can be set per channel type as opposed to setting all dividers to the last configured divider. After setting the new device configuration, an update is automatically performed.

TMSiSAGA SDK v1	TMSiSAGA SDK v2

`device.connect();`
 	
`device.setBaseSampleRate(4000);`
 	
`device.setDividers('uni', 1);`
`device.setDividers('bip', 2);`
 	
`device.setReferenceMethod('aver age');`
 	
`device.updateDeviceConfig();`
 	
`device.start();`
 	
`% Rest of sampling code`
	`device.connect();`
	
	`config = struct('BaseSampleRate', 4000, 'Dividers', {{'uni', 1; 'bip', 2}}, … 'ReferenceMethod', 'average');`
	
	`device.setDeviceConfig(config);`
	
	`device.start();`
	
	`% Rest of sampling code`

Setting channel configuration

A single function to enable a given (sub)set of channels is available to the user, which does not require numerical input based on the channel's number in the device.

TMSiSAGA SDK v1	TMSiSAGA SDK v2

`device.connect();`
`% Enable UNI 1-4, BIP 1-2 and AUX 1-1 – 1-3`
`device.disableChannels( device.channels);`
`device.enableChannels([2:5 34:35, 38:40]);`
`device.updateDeviceConfig();`
`device.start();`
`% Rest of sampling code`
	`device.connect();`
	`% Enable UNI 1-4, BIP 1-2 and AUX 1-1 – 1-3`
	`channelConfig = struct( 'uni', [1:4],'bip', [1 2], 'aux', [1:3]);`
	`device.setChannelConfig( channelConfig);`
	`device.start();`
	`% Rest of sampling code`

6 FAQ

Does the TMSi SAGA interface for MATLAB work in Octave?

A: No. Unfortunately, most of the TMSi SAGA interface for MATLAB does not work in Octave. You can read and write Poly5-files in Octave and do the data processing but you can not use Octave to perform measurements. The 'Library' call that imports the TMSi driver into a MATLAB environment is not supported by Octave. We would encourage and support you where we can, in case you are willing to find a workaround for this in Octave. We are not pursuing the compatibility in Octave unless a 'Library' or similar function is implemented in Octave. If you know a workaround, please let us know.

I manually stopped the script from by pressing 'Ctrl+C' or 'Quit Debugging'. Is there anything I should do before I can start a new measurement?

A: Pressing 'Ctrl+C' stops the script, but does not stop the device from sampling. You have to do a library `cleanUp` to stop and disconnect the device. After the `cleanUp` you can start a new measurement. It is good practice to add a check on whether a Library exists and perform a `cleanUp` before creating a new Library object.

Where can I find the units of my data?

A: The units of the data can be found in the device channel properties.

Why do I have to set a divider for a different sample rate?

A: The sample rate of SAGA device can only be set by the base sample rate divided by a power of 2. For example, with a base sample rate of 4000, the possible sample rates are 4000, 2000, 1000, 500. To get these you will have to set the divider to respectively 0, 1, 2, 3.

`Sample_rate=(base_sample_rate/(2^n))` with `n=[0,1,2,3]`

Can I use the TMSi SAGA interface for MATLAB to program my device to record to a flash card?

A: No, we did not implement direct control for recording to flash disk. This requires TMSi Polybench. Contact TMSi Sales (sales@tmsi.com) for more information.

What do I need to do if I get "docking station response timeout" after an error?

A: First thing to do is to repower both docking station and data recorder and try again. If this does not work, restart MATLAB, else restart computer. Also make sure you have followed the steps mentioned in the User Manual of your device for correct installation.

How can I identify loss of samples?

A: You can check loss of samples through inspection of the STATUS channel. See the User Manual for more information about the STATUS channel. Furthermore, a message is displayed in the command window if samples are lost, which includes the number of missing samples.

Am I allowed to distribute the interface to my colleagues?

A: Yes you are.

I want to control configurations that are not shown in the examples. How do I know how to control them?

A. Not all functionality of the interface is shown in the examples. More information can be found in the inline documentation of the interface. More information about general control of the SAGA device can be found in the SAGA Device API.

How much processing can I do before the TMSi SAGA interface for MATLAB crashes?

A: We do not know, it will depend on your PC, memory, other processes etc.

7 MIT LICENSE

8 CHANGELOG

November 16th, 2020 – Release for V2 SAGA Interface for MATLAB