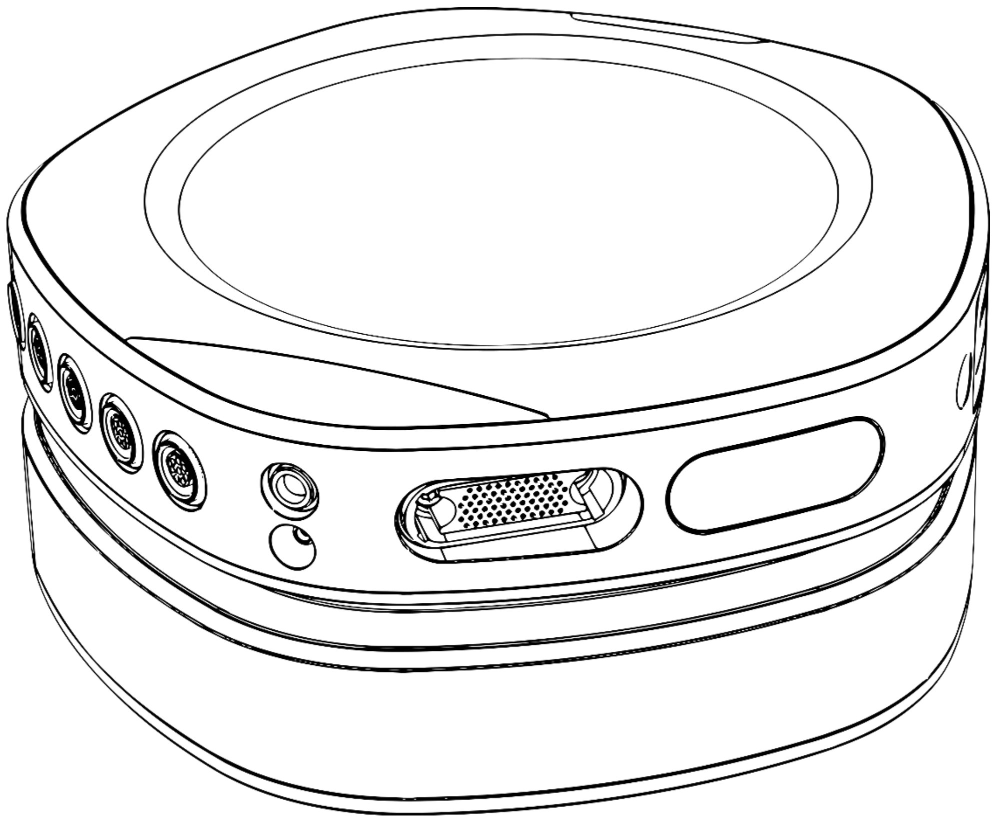
saga 32 64 +



API DOCUMENTATION

TMSI SAGA INTERFACE FOR MATLAB

TMSi

REF: 93-2303-0204-0-1-0

REV 2 **EN**

Last update: 2019-10-18

CE

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.

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.

The fact that this interface is free of charge has implications for the level of support you can expect from TMSi. 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 MATLAB interface can be downloaded from our TMSi website under ‘Support’ (www.tmsi.com). TMSi cannot and will not support you in writing MATLAB code. We can give some hints towards a possible solution or where to look in the code, but that is how far we can go with this. The interface itself is supported by TMSi, just like the usage of the SAGA device and the SAGA Device Driver.

1.5 TMSi Polybench or the TMSi SAGA interface for MATLAB?

TMSi Polybench is our toolbox-based software package which we continue to support and sell. The TMSi SAGA interface for MATLAB will not, in any way, 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 and not free of charge. Please contact sales@tmsi.com for more information.

The TMSi SAGA interface for MATLAB is to be used by engineers and researchers who are familiar or willing to become familiar with MATLAB and scripting. There are many tutorials, videos, webinars and examples on the web available to learn to use MATLAB. The TMSi MATLAB Interface code is easy to understand and well-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) Real-Time plotting of sampled data.
- Directly saving sampled data to Poly5 file format.
- Offline export of .Poly5 data to EEGLAB.

The library has been tested on the following MATLAB versions:

- MATLAB 2016a (Windows 10 / 64bit)
- MATLAB 2017b (Windows 10 / 64 bit)
- MATLAB 2018a (Windows 10 / 64-bit)
- MATLAB 2018b (Windows 10 / 64-bit)
- MATLAB 2019a (Windows 10 / 64-bit)
- MATLAB 2019a (Windows 7 / 64-bit)

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 7 or 10, 64-bit.
- SAGA Device Driver 1.0.1 or higher.
- MATLAB release 9.

The TMSi SAGA interface for MATLAB does not require special toolboxes, although the Signal Processing Toolbox is 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 `getFirstAvailableDevice`. From here on you will first have to `connect()` to the device, before you can start sampling with the `start()`. You will always need to call `stop()` to stop sampling and `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 `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 `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.

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, temprature, 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/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

The Poly5 class provide means to stream data to and store the data in Poly5 files. It also allows offline reading of Poly5 files. These files can be made using TMSi Polybench or the TMSi MATLAB Interface.

+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

A class containing some utility functions, for internal use only.

3 EXAMPLES

The easiest way to start using the interface is 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.

SampleData.m

A very basic measurement setup to give an overview of the steps to be taken: open the library, connect to the device, setup the data acquisition connection and settings and sample 10 times. After that, the connection is closed properly as should be done every time when you use the interface.

SampleToRealTimePlot.m

This example plots the data in real-time as long as the figure is visible. The plot can be scaled automatically by pressing ‘a’ and closed using ‘q’. When the figure is closed, the connection will be closed.

SampleImpedanceMode.m

SAGA devices support impedance measurement to verify that the patient connection is such that a reliable and high-quality measurement can be made. In this example the device is set into Impedance Mode and the impedance values are sampled.

SampleToMemory.m

This example plots the data in real-time and saves data to memory as long as the figure is visible.

SampleToPoly5.m

The acquisition is started and the data will be written into a Poly5-file. This file can later be used for offline processing or be opened in Polybench.

SampleSubsetOfChannels.m

This example shows how to change the device configuration and disable and enable channels. The example shows how to sample with only a subset of channels and change the sample rate and names of the channels.

SampleSensors.m

In this example, sampling of only sensor channels is demonstrated, including retrieval of sensor information.

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.

RepairDataMemory.m

In case of wireless connection between data recorder and docking station, data loss can occur. This example shows how to measure and save data to memory and repair missing samples afterwards.

RepairDataPoly5.m

This example shows how to measure and save data to Poly5 and repair missing samples afterwards.

ReadPoly5.m

This example shows how to read a Poly5-file.

WritePoly5.m

This example shows how to write a Poly5-file

CommonReferenceExample.m

This example shows how to sample in common reference mode.

EMGExample.m

This example shows how to measure EMG and scale it with respect to maximum intensity. The measurements are performed in common reference mode.

SpectogramExample.m

This example can be used to plot a spectrogram of the data and view the frequency content of the data.

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

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 sample rate and the divider have to be the same for all active channels. The `device.sample_rate` property contains the current sample rate the device and is updated after every `updateDeviceConfig()` call.

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 `updateDeviceConfig` function.

TMSi SDK	TMSiSAGA SDK

`sampler = device.createSampler();`
 	
`sampler.setSampleRate(4000);`
 	
`sampler.connect();`
`sampler.start();`
 	
`% rest of sampling code`
	`device = library.getFirstAvailableDevice();`
	
	`device.setBaseSampleRate(4000);`
	`device.updateDeviceConfig();`
	
	`device.connect();`
	`device.start();`
	
	`% rest of sampling code`

Sampling from device

Sampler has been removed and now you sample directly with the Device object. The function `updateDeviceConfig` has to be called manually and is no longer done automatically by the start sampling function.

TMSi SDK	TMSiSAGA SDK

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

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, 250, 125. To get these you will have to set the divider to respectively 0, 1, 2, 3, 4, 5.

`Sample_rate=(base_sample_rate/(2^n))` with `n=[0,1,2,3,4,5]`

The sample rate of all active channels has to be the same.

Can I use the TMSi MATLAB Interface 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.

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 the SAGA Device API.

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

A: This is impossible to answer, as it will depend on your PC, memory, other processes and in general: because we don’t know. We did some testing with Filtering, online spectrograms, and they worked fine.

6 MIT LICENSE

7 CHANGELOG

October 15th, 2019 – Initial Release (Rev1)

October 18th, 2019 – Added SoftPro disclaimer

Author	ASN
Reviewer	FM
Approver	CB