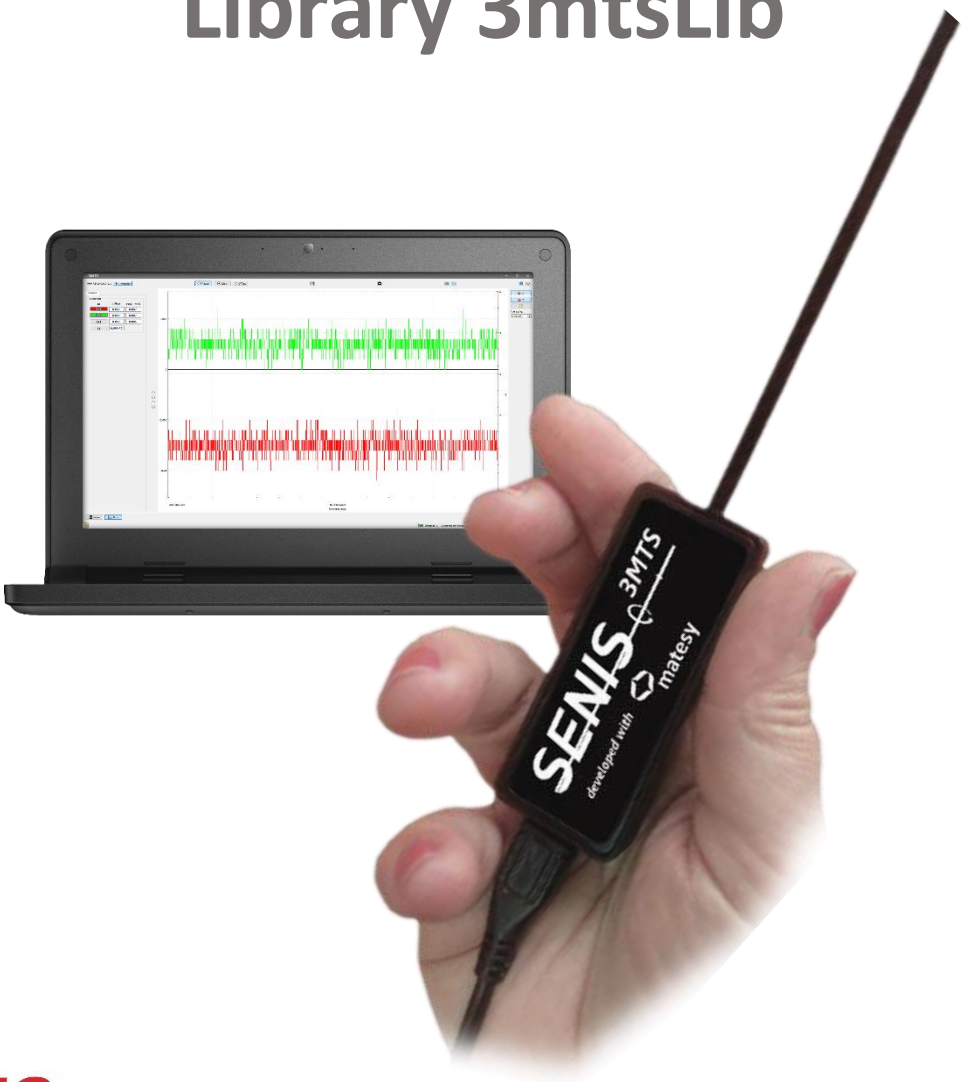
3MTS API Manual
Library 3mtsLib



Neuhofstrasse 5a
CH-6340 Baar, Switzerland
mappers@senis.ch
www.senis.ch
+41 44 508 7029

Distributed by: GMW Associates. 955 Industrial Road, San Carlos, CA, 94070, USA
PHONE: +1-650-802-8292 FAX: +1-650-802-8298 EMAIL: sales@gmw.com WEB: www.gmw.com

1. Table of Contents

1. Introduction

3mtsLib is a software library API for an easy data acquisition and communication with SENIS 3MTS teslameters. The software consists of a DLL to be imported into customer's data acquisition system.

2. Standard structures

```
typedef struct {
   int dimSize;
   int elt[x];
   } TD2;
typedef TD2 **TD2Hdl;

typedef struct {
   int dimSize;
   char elt[x];
   } TD3;
typedef TD3 **TD3Hdl;
```

3. Function Reference

3.1. Count devices

Count the available devices.

Declaration: int count_devices(unsigned short* number_of_devices)

Output:

number_of_devices : Number of available devices

Return value:

0x0	ok
0x8000	Device not initialized

3.2. Open device

Open the measurement device.

Declaration: int open_device(int* device_number)

Input:

device_number : Device number

Return value:

0x0	ok
0x8000	Device not initialized

3.3. Close device

Close the opened measurement device.

Declaration: int close_device(int* device_number)

Input:

device_number : Device number

Return value:

0x0	ok
0x8000	Device not initialized

3.4. Get sensor count

Return the number of sensor channels.

Declaration: int get_sensor_count(int* device_number, int *sensor_count)

Input:

device_number : Device number

Output:

sensor_count: Number of sensor channels

Return value:

0x0	ok
0x8000	Device not initialized

3.5. Get sensor values

Get measured values from the device. Before calingl this function, you have to allocate space for the TD2Hdl structure.

In trigger mode, the function returns either the last measured value or an error value of 1 if there is no new measured value in the buffer. See also chapter Trigger mode

Declaration:

int get_sensor_values (int* device_number, unsigned long* timestamp, TD2Hdl values)

Output:

timestamp: Time stamp

values: Measure values in µT

Return value:

0x0 ok

0x0001 no new value

0x8000 Device not initialized

Declaration:

int get_sensor_values_fl (int* device_number, unsigned long* timestamp, float* sensor_x, float* sensor_y, float* sensor_z)

Output:

timestamp: Time stamp

sensor_x,sensor_y,sensor_z: Measure values in µT

Return value:

0x0 ok

0x0001 no new value

0x8000 Device not initialized

3.6. Set range

Set the measurement range of the sensor.

Declaration: int set_range (int* device_number ,unsigned short range)

Input:

device_number : Device number

range: Measurement range

Value	Measurement range
0	0,1 T
1	0,5 T
2	3 T
3	20 T

Return value:

0x0	ok
0x8000	Device not initialized
0x8001	Range outside of value range

3.7. Get range

Get the measurement range of the sensor.

Declaration: int get_range (int* device_number, unsigned short* range)

Input:

device_number : Device number

Output:

range: Measurement range

Value	Measurement range
0	0,1 T
1	0,5 T
2	3 T
3	20 T

Return value:

0x0	ok
0x8000	Device not initialized

3.8. Set speed

Set measurement speed.

Declaration: int set_speed(int* device_number,unsigned short speed)

Input:

device_number : Device number

speed: Measure time period

Return value:

0x0	ok
0x8000	Device not initialized

3.9. Get speed

Get measurement speed.

Declaration: int get_speed(int* device_number, unsigned short* speed)

Input:

device_number : Device number

Output:

speed: Measure time period

Return value:

0x0	ok
0x8000	Device not initialized

3.10. Get firmware version

Get firmware version. Before you call this function, you have to allocate space for the TD3Hdl structure.

Declaration: int get_firmware_version(int* device_number,TD3Hdl values)

Input:

device_number : Device number

Output:

values: Firmware version

Return value:

0x0	ok
0x8000	Device not initialized

Declaration: int get_firmware_version_ch (int* device_number,char* name)

Input:

device_number : Device number

Output:

name:	Firmware version

Return value:

0x0	ok
0x8000	Device not initialized

3.11. Get device name

Get the name of the measurement device. Before you call this function, you have to allocate space for the TD3Hdl structure.

Declaration: int get_device_name(int* device_number,TD3Hdl values)

Input:

device_number : Device number

Output:

values: Device name

Return value:

0x0	ok
0x8000	Device not initialized

Declaration: int get_device_name_ch (int* device_number,char* name)

Input:

device_number : Device number

Output:

name: Device name

Return value:

0x0	ok
0x8000	Device not initialized

3.12. Clear buffer

Clear the internal value buffer.

Declaration: int clear_buffer(int* device_number)

Input:

device_number : Device number

Return value:

0x0	ok
0x8000	Device not initialized

3.13. set Trigger

Set Trigger mode.

Declaration: int set_trigger (int* device_number, unsigned short* triggermode)

Input:

device_number : Device number

triggermode: Trigger mode (0: off, 1: on)

Return value:

0x0	ok
0x8000	Device not initialized

3.14. get Trigger

Get state of trigger mode.

Declaration: get_trigger(int* device_number, unsigned short* triggermode)

Input:

device_number : Device number

Output:

triggermode: Trigger mode (0: off, 1: on)

Return value:

0x0	ok
0x8000	Device not initialized

4. Trigger mode

If the trigger mode is activated, all measured values must first be read out in the internal buffer of the driver. This is done by repeatedly calling the Get-values function. Thereafter, measured values can be stored in the buffer by means of a trigger pulse. The measured values can then be read out using the get-values function. If no new measured values are contained in the buffer, the function returns 0x1 as the return value.