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.MOBIlab+ MOBILE LABORATORY

g.MOBIlab+ C API PC
USER MANUAL V2.14.01

How to contact g.tec:

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

Content

To the Reader

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

Our team is prepared to find the better solution for your needs.
Take advantage of our experience!

Dr. Christoph Guger Dr. Guenter Edlinger

Researcher and Developer

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

Scientist

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

PREFACE

This section includes the following topics:

Required Products – Microsoft Windows compatible compiler and development environment

Related Products

Using This Guide – Suggestions for reading the handbook

Conventions – Text formats in the handbook

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

REQUIRED PRODUCTS

g.MOBIlab+ C API comes with a demo application to give easy access to the hardware interface. The demo application, the source code and project files are provided for the Microsoft Visual Studio 2010 C++ environment.

However, the g.MOBIlab+ C API and DLL can also be used with other compilers under Microsoft Windows 7.

It is assumed that user has the following products available to perform the programming examples.

- **Microsoft Visual Studio 2010 C++ SP1**: for compiling the sample code of the demo application
- **g.MOBIlab+**: mobile biosignal laboratory system
- **PC**: Windows 7 Professional is necessary to run gMOBIlabPCdemo.exe

RELATED PRODUCTS

g.tec provides several biosignal analysis elements that are especially relevant to the kinds of tasks you perform with g.MOBIlab+ C 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

USING THIS GUIDE

It is assumed that you are familiar with:

- programming in C++
- Microsoft Visual Studio 2010 SP1 environment or equivalent programming environment
- g.MOBIlab+ hardware and manual

Chapter "Installation and Configuration" lists hardware and software requirements and explains the installation of the software.

Chapter "Interfacing g.MOBIlab+" discusses the software interface to g.MOBIlab+.

CONVENTIONS

Item	Format	Example
C++ source code	`Courier`	to open a g.MOBIlab+ type `HANDLE hdevice = OpenDevice();`
Files and Paths	`Courier`	Browse to the folder `C:\Program Files\gtec\gMOBIlabCAPI`
Menu items	**Boldface**	Select **Save as …** from the **File** menu.

INSTALLATION AND CONFIGURATION

Installing Microsoft Visual Studio 2010 C++

Microsoft Visual Studio 2010 C++ delivers a complete desktop development environment for creating applications and system components for Microsoft Windows.

Instructions:

- Please refer to Microsoft Visual Studio 2010 C++ installation instructions.

Installation of Bluetooth dongle and g.MOBIlab+ hardware

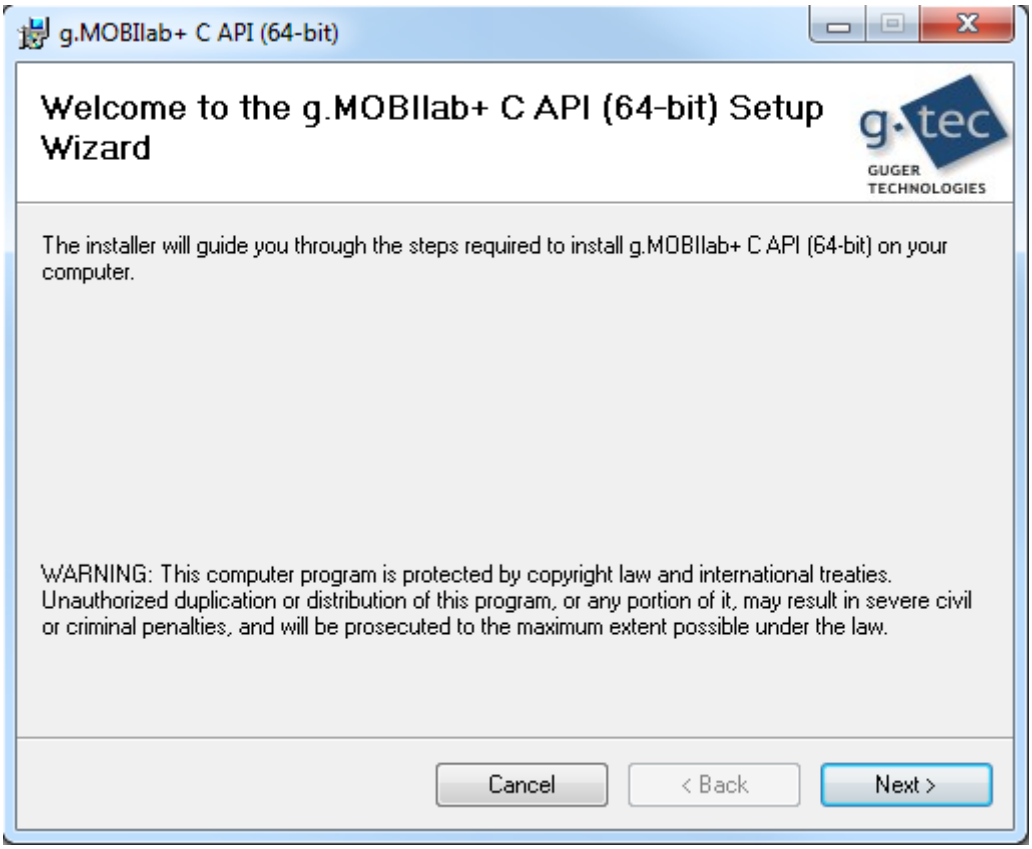
- For the g.MOBIlab+ hardware installation please refer to "gMOBIlabInstructionsForUse.pdf" for g.MOBIlab+ and if you also ordered the Bluetooth interface see the "gMOBIlabBluetoothInstallation.pdf" too. Both manuals are available on the g.MOBIlab+ CD.

Installing the g.MOBIlab+ C API software demo for PC

1. If there is any old version of the g.MOBIlab+ C API package on your computer please uninstall it.

2. It is highly recommended to turn off the User Account Control (UAC) of the Windows 7 operating system. Please see last page of this document how to do this.

3. Close all running applications.
   Insert the g.tec installation CD, open the `g.MOBIlab\g.MOBIlab+ C API` directory and select the correct directory for the architecture of the PC (`Win32` or `Win64`). To install the g.MOBIlab+ C API double-click the `setup.exe`. This opens the following welcome dialog (see picture below):



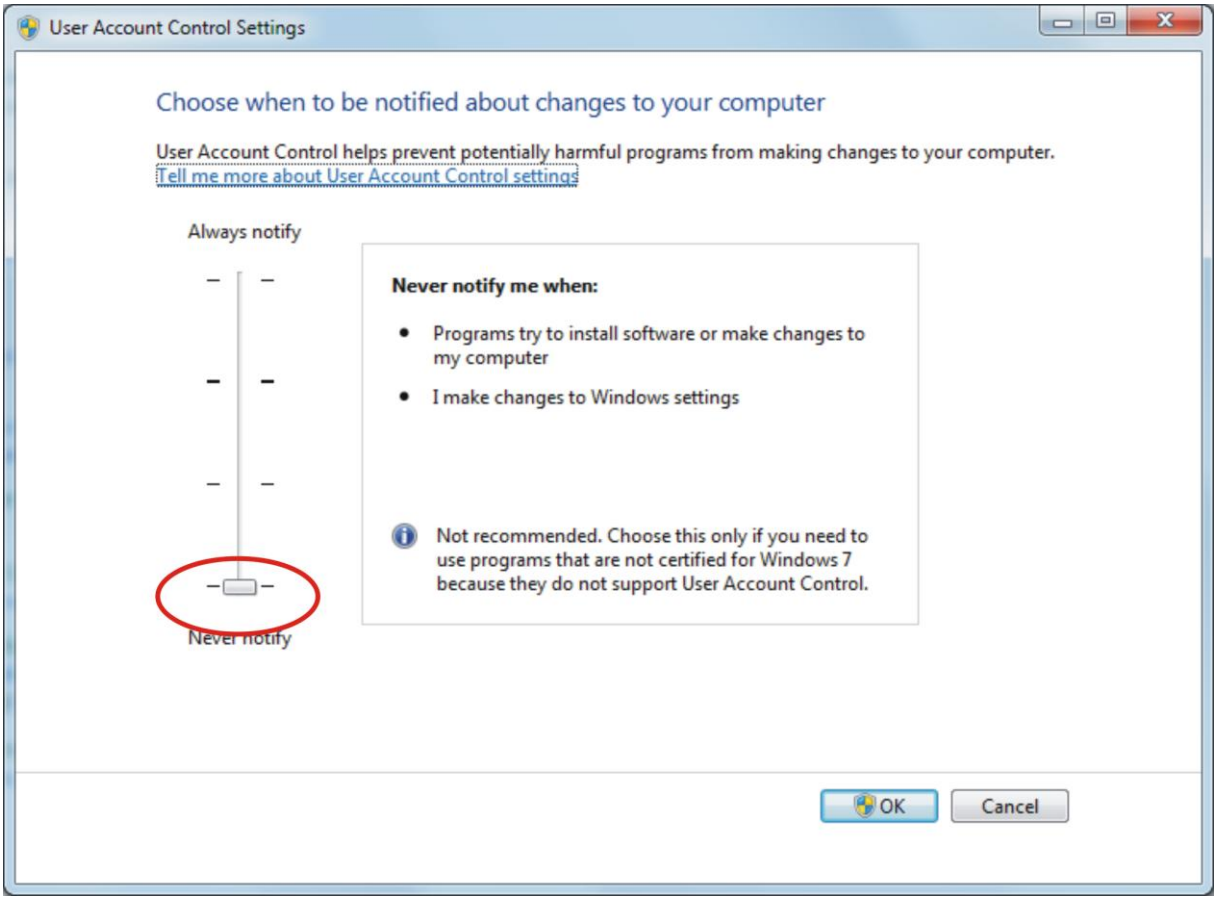
   Click **Next**. Please read the **License Agreement** for g.MOBIlab+ C API and if you agree with the terms click **I Agree** and **Next**. Then just follow the steps on the screen.

Documentation

The g.MOBIlab+ PC C API documentation can be found in the `Win32` or `Win64` folder on the CD in pdf format. Use the Acrobat Reader to view the documentation.

Turn off User Account Control (UAC)

Please switch off the User Account Control Settings on your system: Click on the Windows **Start** button → **Control Panel** → **System and Security** → **Change User Account Control Settings**. Set the slider to the lowest position (Never notify).



THE g.MOBIlab+ DEMO SOURCE CODE

A Visual Studio 2010 C++ demo project comes with your g.MOBIlab+ C-API installation which is intended to demonstrate the usage of the g.MOBIlab+ C-API. The Visual Studio 2010 project file `gMOBIlabPlusDemo.vcxproj` should be located under the following path:

```
C:\Program Files\gtec\gMOBIlabCAPI\gMOBIlabDemo\
```

On execution the demo project will open the specified g.MOBIlab+ device, initialize it and record data for a specific amount of seconds to a binary file on the harddisk. Before you compile and execute the demo project please read the following section to configure it properly.

MODIFYING THE SOURCE CODE PARAMETERS

In the `gMOBIlabPlusDemo.cpp` file all the communication with g.MOBIlab+ is done. After the include directives at the beginning of the file you'll find a section commented with "`//configuration`" where several global configuration parameters can be adjusted:

```
//configuration
const long NUM_SECONDS_RUNNING = 10;
const unsigned int BUFFER_SIZE_SECONDS = 1;
const unsigned int NUMBER_OF_CHANNELS = 8;
const unsigned int NUMBER_OF_SCANS = 8;

LPSTR _serialPort = "COM1";
BOOL _recordAllDigitalChannels = TRUE;
BOOL _enableTestmode = FALSE;
BOOL _streamToSDcard = FALSE;
LPSTR _filenameOnSDcard = "GMOBILAB";
```

NUM_SECONDS_RUNNING

The number of seconds that data will be recorded from the device.

BUFFER_SIZE_SECONDS

The demo project executes two separate threads: one continuously receives data from the device and stores it into an internal cyclic buffer. The other one continuously reads data from that cyclic buffer and writes it to the harddisk. This ensures that data acquisition is independend from data processing avoiding possible loss of data when processing is too time consuming.
The **BUFFER_SIZE_SECONDS** constant represents the length of the internal cyclic buffer in seconds and is set to 1 second in this example. It shouldn't be necessary to change this.

NUMBER_OF_CHANNELS

Specify the number of channels that should be recorded. The value can range from 1 to 8 but mustn't be bigger than 8 or less than 1. A value of 5 for example makes the application record channels 1 to 5.

NUMBER_OF_SCANS

The application periodically receives data from the device consisting of several complete scans. One complete scan consists of exactly one sample for each channel. The **NUMBER_OF_SCANS** value determines the number of complete scans that should be received at once per `GT_GetData` call. The device buffers data until this specified amount of scans is reached and then transmits it at once to the application.
Please ensure that the maximum amount of bytes that will be received per `GT_GetData` call doesn't exceed 1024. This amount of bytes per `GT_GetData` call can be calculated by: `NUMBER_OF_SCANS * (NUMBER_OF_CHANNELS + _recordAllDigitalChannels) * sizeof(short)`.

_serialPort

It is necessary to specify the serial port where your g.MOBIlab+ is currently connected to. In this example the port used in the demo is set to COM1.

_recordAllDigitalChannels

If set to TRUE all 8 digital channels will be recorded as inputs as well. FALSE to not record any digital channel.

_enableTestmode

To check proper operation of the g.MOBIlab+ it has a built-in testmode. When the device is set to testmode it sends test data (different sawtooths on the analog channels, a rectangle on the digital channels) on all analog and digital channels instead of the real measured samples. TRUE sets the device to testmode, FALSE records real measured data.

_streamToSDcard

If an SD card is inserted into the device's SD card slot data can additionally be streamed to and recorded on this SD card.
TRUE to additionally stream data to SD card, otherwise FALSE. In every case data will be recorded onto the harddisk.

_filenameOnSDcard

If **_streamToSDcard** is set to TRUE, you must specify a filename for the file on the SD card where the data will be recorded to. This filename mustn't contain more than 8 characters. A file extension is not necessary (it's a binary file, the same that will be recorded on the harddisk). Lowercase letters in the filename are converted internally to uppercase.

COMPILING THE SOURCE CODE

You can compile the source code for two different platforms: 32-bit (Win32) and 64-bit (x64). The desired platform can usually be selected through the Visual Studio toolbar or in the Visual Studio Configuration Manager. When compiling with x64 configuration you can execute the created `gMOBIlabPlusDemo.exe` file only on a 64-bit operating system with the 64-bit g.MOBIlab+ driver installed. Before compiling the source code a solution file has to be saved for the project.

EXECUTING THE PROJECT (gMOBIlabPlusDemo.exe)

On executing the compiled `gMOBIlabPlusDemo.exe` file a console window will show up telling you that a g.MOBIlab+ on the specified port will be opened. The state of the g.MOBIlab+ will be printed when it was opened successfully and the application starts to receive data for the specified amount of seconds. This data will be written to the file

```
receivedData.bin
```

in the same directory the `gMOBIlabPlusDemo.exe` was executed from (can be different if the demo is started from the Visual Studio project folder).

OPENING THE receivedData.bin FILE

The file format of the `receivedData.bin` file is compliant with the g.MOBIlab+ data file format. Please see the documentation of the g.MOBIlab+ data file Format for details.
You can easily open the file in MATLAB using the `importgMOBIlabplus` command which comes with the `importgMOBIlabplus.m` file, part of the g.MOBIlab+ C-API installation (in the Tools folder).

INTERFACING g.MOBIlab+

To have access to g.MOBIlab+ g.tec provides an easy to use API described in this chapter. It is assumed that you are familiar in programming C++ applications. Please use the provided sample code as a reference.

INCLUDING THE C API INTO YOUR APPLICATION

The C API can be accessed through a standard dynamic link library (gMOBIlabplus.dll). The files to include the C API into your project can be found in the folder

`C:\Program Files\gtec\gMOBIlabCAPI\Lib`

Corresponding to the architecture of the PC either `gMOBIlabplus.lib` or `gMOBIlabplus_x64.lib` are available in the folder mentioned above.

To include this interface into your custom application perform the following steps:

- Include gMOBIlabplus.lib e.g. in your Visual C++ source file(s) with

```
#ifdef _WIN64
#pragma comment(lib,"gMOBIlabplus_x64.lib")
#else
#pragma comment(lib,"gMOBILabplus.lib")
#endif
```

- Include the header-file gMOBIlabplus.h in your source file(s)

```
#include "gMOBIlabplus.h"
```

FUNCTION REFERENCE

There are several functions available to interface g.MOBIlab+. For examples please refer to the demo project included with the g.MOBIlab+ PC C API.

Structures

- _AIN Structure used to pass analog channel settings to the driver

  items: BOOL ain1: set TRUE if channel should be scanned, otherwise set to FALSE
  BOOL ain2: set TRUE if channel should be scanned, otherwise set to FALSE
  BOOL ain3: set TRUE if channel should be scanned, otherwise set to FALSE
  BOOL ain4: set TRUE if channel should be scanned, otherwise set to FALSE
  BOOL ain5: set TRUE if channel should be scanned, otherwise set to FALSE
  BOOL ain6: set TRUE if channel should be scanned, otherwise set to FALSE
  BOOL ain7: set TRUE if channel should be scanned, otherwise set to FALSE
  BOOL ain8: set TRUE if channel should be scanned, otherwise set to FALSE

- _DIO Structure used to pass digital channel settings to the driver

  items: BOOL dio1_enable; // TRUE enables, FALSE disables channel for scanning (DIN 1)
  BOOL dio2_enable; // TRUE enables, FALSE disables channel for scanning (DIN 1)
  BOOL dio3_enable; // TRUE enables, FALSE disables channel for scanning (DIN 3)
  BOOL dio4_enable; // TRUE enables, FALSE disables channel for scanning (DIO 4)
  BOOL dio5_enable; // TRUE enables, FALSE disables channel for scanning (DIO 5)
  BOOL dio6_enable; // TRUE enables, FALSE disables channel for scanning (DIO 6)
  BOOL dio7_enable; // TRUE enables, FALSE disables channel for scanning (DIO 7)
  BOOL dio8_enable; // TRUE enables, FALSE disables channel for scanning (DIO 8)

  BOOL dio4_direction; // TRUE sets direction to "IN", FALSE to "OUT" (DIO 4)
  BOOL dio5_direction; // TRUE sets direction to "IN", FALSE to "OUT" (DIO 5)
  BOOL dio6_direction; // TRUE sets direction to "IN", FALSE to "OUT" (DIO 6)
  BOOL dio7_direction; // TRUE sets direction to "IN", FALSE to "OUT" (DIO 7)

- _BUFFER_ST used to pass and retrieve data

  items: SHORT *pBuffer: pointer to a buffer of SHORT
  UINT size: buffer size in bytes (max 1024 bytes)
  UINT validPoints: number of data points (SHORT) returned from driver

- _ERRORSTR used to hold error string retrieved from driver

  items char Error[256]: character array to hold error string

- __CHANNEL used to hold configuration of one analog channel

  items: float highpass: cut off frequency of highpass
  float lowpass cut off frequency of lowpass
  float sensitivity analog input range [µV]
  float samplerate rate at which analog channel is sampled
  char polarity polarity of analog channel (B…bipolar, U…unipolar)

- __CFG used to hold configuration for g.MOBIlab+

  items:

	__int16 version	version of g.MOBIlab+
	char serial[14]	serial number of g.MOBIlab+
	__channel channels[8]	settings for the eight analog channels

- __DEVICESTATE used to hold the state of g.MOBIlab+ if streaming to SD card

  items:

	char serial[13]	serial number of g.MOBIlab+
	__int8 AChSel	selected analog channels (bit 0...channel 8, bit 7...channel 1)
	__int8 DchSel	selected digital channels (bit 0...channel 8, bit 7...channel 1)
	__int8 DchDir	direction of digital channels (0…input, 1…output, bit 7…channel 8, bit 0…channel 1)
	char SDstate	state of SD card ‘0’…no SD card inserted / not found, ‘1’…initialized but not streaming, ‘s’…streaming
	__int32 SDsize	size on SD card left in Bytes

See gMOBIlabplus.h for further information.

GT_OpenDevice

```
HANDLE GT_OpenDevice(LPSTR lpPort)
```

Opens the serial port, initializes the g.MOBIlab+ and returns a handle to it.

You have to store the handle since all function calls of the API need this handle as an input parameter.
You have to close this handle at the end of your application by calling GT_CloseDevice

See also:

GT_GetDeviceStatus
GT_GetSDcardStatus
GT_GetConfig

GT_GetDeviceStatus

```
BOOL GT_GetDeviceStatus(HANDLE hDevice, __DEVICESTATE * DevState);
```

Retrieves the actual status of g.MOBIlab+ when the device is in streaming mode.

Input parameters:

- HANDLE hDevice: gives access to the device
- Struct __DEVICESTATE: contains status information if g.MOBIlab+ is streaming to SD card

The function returns true if the call succeeded, otherwise it returns false.
The function also returns false if the device is not in streaming mode.

Use GT_GetLastError to retrieve error code. If g.MOBIlab+ is not in streaming mode, the error code returned is 16.

GT_GetSDcardStatus

```
BOOL GT_GetSDcardStatus(HANDLE hDevice, UINT * SDStatus);
```

Retrieves the remaining storage capacity available on the SD card in bytes. If no card is inserted into g.MOBIlab+, SDStatus is 0.

Input parameters:

- HANDLE hDevice: gives access to the device
- UINT * SDStatus: pointer to the UINT holding the remaining storage capacity on the SD card

The function returns true if the call succeeded, otherwise it returns false.

Use GT_GetLastError to retrieve error code

GT_GetConfig

```
BOOL GT_GetConfig(HANDLE hDevice, __CFG * cfg);
```

Retrieves the configuration of your g.MOBIlab+ and stores it in the structure __CFG.

Input parameters:

- HANDLE hDevice: gives access to the device
- __CFG * cfg: Pointer to a structure __CFG which holds the configuration

The function returns true if the call succeeded, otherwise it returns false.

Use GT_GetLastError to retrieve error code.

GT_InitChannels

```
BOOL GT_InitChannels(HANDLE hDevice, _AIN analogCh, _DIO digitalCh);
```

Initializes analog and digital channels for scanning and the direction for the digital I/O s

Input parameters:

- HANDLE hDevice: gives access to the device
- _AIN analogCh: sets analog channels selected for scanning
- _DIO digitalCh: sets digital lines selected for scanning and sets direction of digital I/O s

The function returns true if the call succeeded, otherwise it returns false.

Use GT_GetLastError to retrieve error code

NOTE: Do not call this function when device is running!

GT_SetTestmode

```
BOOL GT_SetTestmode(HANDLE hDevice, BOOL Testmode);
```

This function sets g.MOBIlab+ to test mode or to measurement mode.

Input parameters:

- HANDLE hDevice: gives access to the device
- BOOL Testmode: true if testmode is to be enabled, false if testmode is not used

The function returns true if the call succeeded, otherwise it returns false.

Use GT_GetLastError to retrieve error code.

GT_EnableSDcard

```
BOOL GT_EnableSDcard(HANDLE hDevice, BOOL enSDcard);
```

Enables the SDcard in g.MOBIlab+ for data streaming.

Input parameters:

- HANDLE hDevice: gives access to the device
- BOOL enSDcard: true if SD card is to be used, false otherwise

The function returns true if the call succeeded, otherwise it returns false.

Use GT_GetLastError to retrieve error code.

GT_SetFilename

```
BOOL GT_SetFilename(HANDLE hDevice, LPSTR filename, int length);
```

This function sets the filename which is to be used for the file written to SD card. It consists of max. 8 characters. Note that lowercase characters are converted to uppercase internally.

Input parameters:

- HANDLE hDevice: gives access to the device
- LPSTR filename: pointer to the string holding the filename
- int length: integer value holding the number of characters of filename

The function returns true if the call succeeded, otherwise it returns false.

Use GT_GetLastError to retrieve error code.

GT_StartAcquisition

```
BOOL GT_StartAcquisition(HANDLE hDevice);
```

Starts acquiring data from g.MOBIlab+; data can be retrieved using GT_GetData

Input parameters:

- HANDLE hDevice: gives access to the device

The function returns true if the call succeeded, otherwise it returns false.

Use GT_GetLastError to retrieve error code

Do not call any other C API functions except GT_GetData and GT_StopAcquisition after data acquisition is started.

GT_GetData

```
BOOL GT_GetData(HANDLE hDevice,_BUFFER_ST *buffer,LPOVERLAPPED lpOvl);
```

Retrieves data from g.MOBIlab+; acquisition must be started before. See GT_StartAcquisition.

Input parameters:

- HANDLE hDevice: gives access to the device
- _BUFFER_ST *buffer: pointer to a structure to hold data
- LPOVERLAPPED lpOvl: OVERLAPPED structure

The driver fills the pointed buffer with interleaved data. The analog channels are scanned in ascending order and the following two bytes hold the values of the digital lines:

Sample #1

SHORT channel 1	if analogCh.ain1 = TRUE	(see: GT_InitChannels)
SHORT channel 2	if analogCh.ain2 = TRUE	
SHORT channel 3	if analogCh.ain3 = TRUE	
SHORT channel 4	if analogCh.ain4 = TRUE	
SHORT channel 5	if analogCh.ain5 = TRUE	
SHORT channel 6	if analogCh.ain6 = TRUE	
SHORT channel 7	if analogCh.ain7 = TRUE	
SHORT channel 8	if analogCh.ain8 = TRUE	
SHORT digital lines	if the digital channels are used	

Sample #2

SHORT channel 1	if analogCh.ain1 = TRUE
SHORT channel 2	if analogCh.ain2 = TRUE

.
..

The values of the digital lines are coded in the bits of SHORT digital lines:

bit 0:	digital IN 1
bit 1:	digital IN 3
bit 2:	digital I/O 4
bit 3:	digital IN 2
bit 4:	digital I/O 5
bit 5:	digital I/O 6
bit 6:	digital I/O 7
bit 7:	digital IN 8

The function returns true if the call succeeded, otherwise it returns false.

Use GT_GetLastError to retrieve error code

GT_SetDigitalOut

```
BOOL GT_SetDigitalOut(HANDLE hDevice, UCHAR diout);
```

Sets digital output lines to a selected digital value.

Input parameters:

- HANDLE hDevice: gives access to the device
- UCHAR diout: 8 bit value to control the digital outputs

  bit 7: high value indicates that DIO 4 should be changed
  bit 6: high value indicates that DIO 5 should be changed
  bit 5: high value indicates that DIO 6 should be changed
  bit 4: high value indicates that DIO 7 should be changed
  bit 3: new value for DIO 4 (high or low)
  bit 2: new value for DIO 5 (high or low)
  bit 1: new value for DIO 6 (high or low)
  bit 0: new value for DIO 7 (high or low)

The function returns true if the call succeeded, otherwise it returns false.

Use GT_GetLastError to retrieve error code

GT_PauseXfer

```
BOOL GT_PauseXfer(HANDLE hDevice);
```

Function to disconnect g.MOBIlab+ from the PC while streaming data to SDcard. Must not be used when device is not streaming.
Input parameters:

- HANDLE hDevice: gives access to the device

The function returns true if the call succeeded, otherwise it returns false.
Use GT_GetLastError to retrieve error code.

GT_ResumeXfer

```
BOOL GT_ResumeXfer(HANDLE hDevice);
```

Resumes data transfer from g.MOBIlab+ to the PC.

Input parameter:

- HANDLE hDevice: gives access to the device

The function returns true if the call succeeded, otherwise it returns false.

Use GT_GetLastError to retrieve error code.

GT_StopAcquisition

```
BOOL GT_StopAcquisition(HANDLE hDevice);
```

Stops acquiring data from g.MOBIlab+

Input parameters:

- HANDLE hDevice: gives access to the device

The function returns true if the call succeeded, otherwise it returns false.

Use GT_GetLastError to retrieve error code

GT_CloseDevice

```
BOOL GT_CloseDevice(HANDLE hDevice);
```

Closes g.MOBIlab+ identified by the handle hDevice.

The function returns true if the call succeeded, otherwise it returns false.

Use GT_OpenDevice to retrieve a handle to the g.MOBIlab+.

Use GT_GetLastError to retrieve error code

For example see "gMOBIlabplusDemo.cpp" in your demo project.

GT_GetDriverVersion

```
float GT_GetDriverVersion();
```

Returns the driver version of the DLL used.

Has no input parameter.

Output parameter is a floating point number holding the current driver version.

GT_GetLastError

```
BOOL GT_GetLastError(UINT * LastError);
```

Retrieves error code from last occurred error

Input parameters:

- UINT * LastError: pointer to an unsigned integer to hold code of last occurred error

The function returns true if the call succeeded, otherwise it returns false.

GT_TranslateErrorCode

```
BOOL GT_TranslateErrorCode(_ERRSTR * ErrorString, UINT ErrorCode);
```

Translates error code to a string.

Input parameters:

- _ERRSTR ErrorString: structure to hold retrieved error string
- UINT ErrorCode: specifies the error (see: GT_GetLastError)

The function returns true if the call succeeded, otherwise it returns false.

Product Page (Web)

Please visit our homepage www.gtec.at for

- Update announcements
- Downloads
- Troubleshooting
- Additional demonstrations

CONTACT


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