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 V2.09a

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**

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.

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.

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 2005 C++ environment.

However, the g.MOBIlab+ C API and DLL can be utilized to program user applications based on different compilers supporting the Windows operation systems as well.

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

- **Microsoft Visual Studio 2005 C++**: for compiling the sample code of the demo application
- **g.MOBIlab+**: mobile biosignal laboratory system
- **PC**: running Windows Vista Service Pack 1 or XP Service Pack 3 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 using C++
- Microsoft Visual Studio 2005 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 "Quick Start" introduces basic features and capabilities of g.MOBIlab+.

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();`
Menu items	**Boldface**	Select **Save as …** from the **File** menu.

INSTALLATION AND CONFIGURATION

Installing Microsoft Visual Studio 2005 C++

The Microsoft Visual Studio 2005 C++ tool delivers a complete desktop development environment for creating applications and system components also for Windows PC powered devices.

Instructions:

- Please refer to Microsoft Visual Studio 2005 C++ installation instructions

Installation of Bluetooth dongle and g.MOBIlab+ hardware

- For the g.MOBIlab+ hardware installation please refer to “Instructions for Use” for g.MOBIlab+ and if you also ordered the Bluetooth interface see the “Ezurio Bluetooth driver installation” too. Both manuals are available on the g.MOBIlab+ CD.

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

1. Please insert the g.MOBIlab+ C API CD into the CD-drive of your PC
2. Browse to the `PC` folder
3. Start the installation by double clicking on `Setup.msi` in the `PC` folder on the g.MOBIlab+ C API CD. The default installation folder is

   ```
   C:\Program Files\gtec
   ```

   The folders

   ```
   \gMOBIlabCAPI\PC
   ```

   are created automatically during the installation.

Documentation

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

QUICK START

Once you have completed the installation of g.MOBIlab+ C API you can test your installation following the steps in this section.

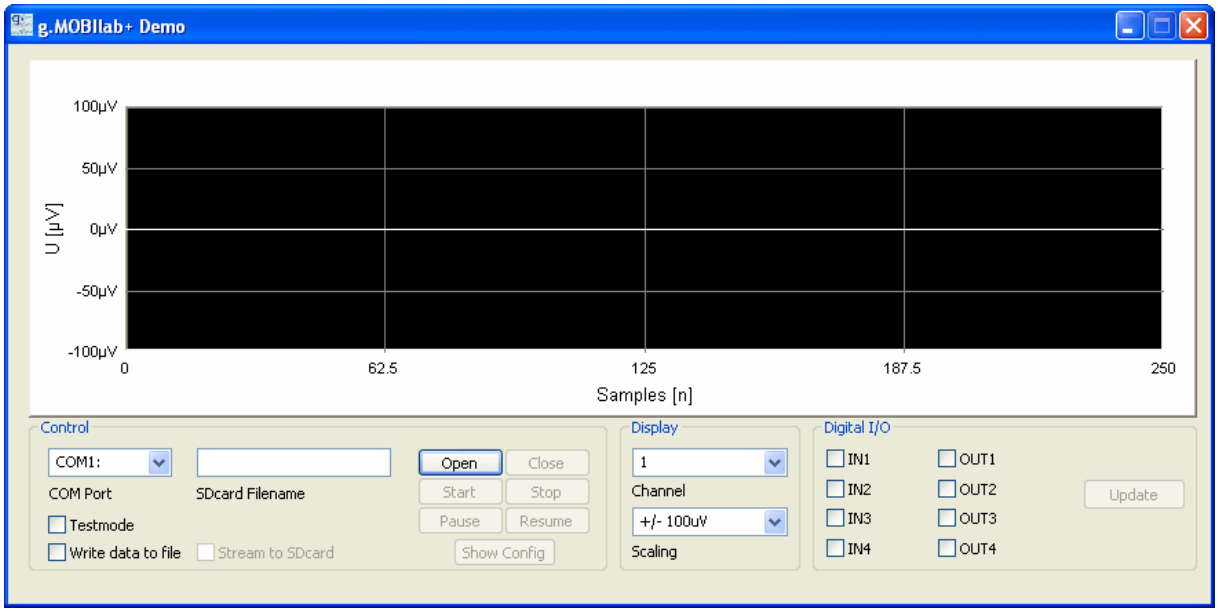
Quick Start will introduce you into the following topics:

RUNNING gMOBIlabPCdemo.exe

1. Run **gMOBIlabPCdemo.exe** to explore the functionality of the software package from the directory

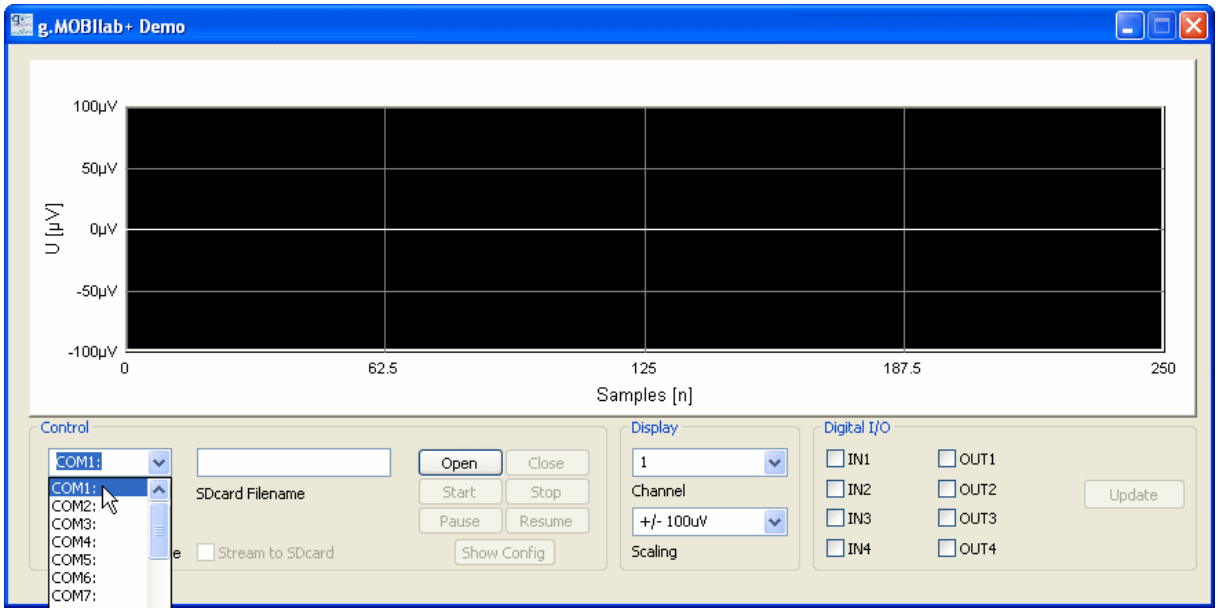
   ```
   C:\Program Files\gtec\gMOBIlabAPI\PC\Bin
   ```

   or double click on the icon **g.MOBIlab+ PC Demo** on the desktop. The following dialog appears:



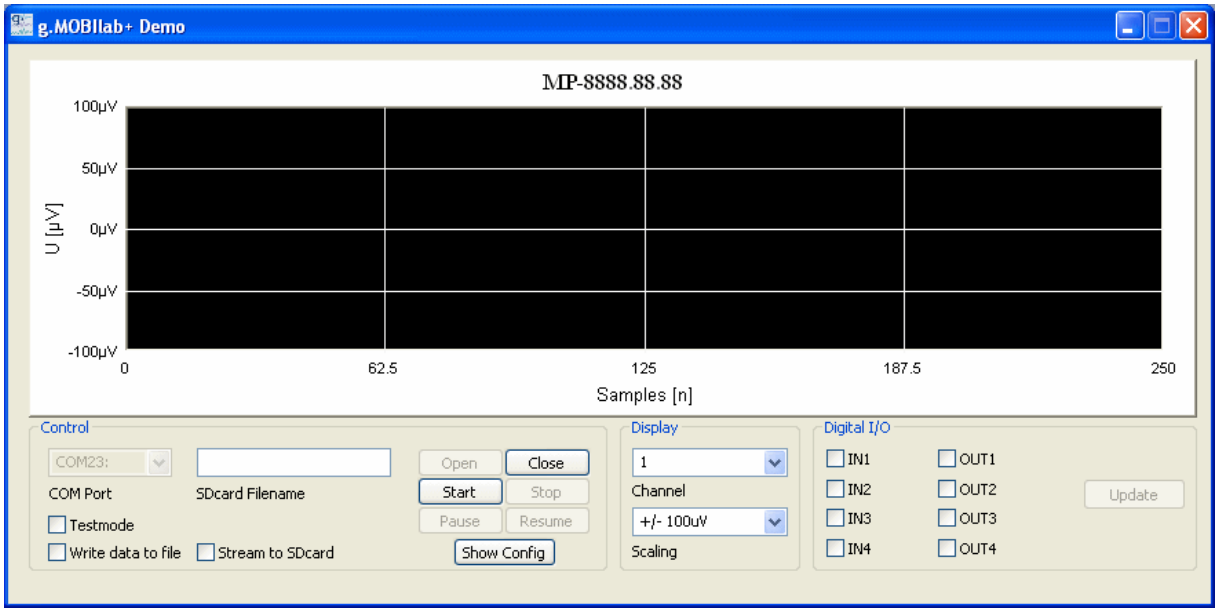
2. Establish the connection from the PC to the g.MOBIlab+ by Bluetooth connection or serial cable connection to the correct COM port. If you are not sure how to do this please refer to the “Ezurio Bluetooth Driver Installation” to set up the connection.

3. Select desired COM connection from the combo box **COM Port**



4. Press **Open** to open the connection to g.MOBIlab+ and read the device state (SDcard inserted, device is currently streaming to SDcard). The **COM Port** list box will be disabled by clicking the **Open** button until the **Close** button is clicked. The serial number of the g.MOBIlab+ device is displayed in the upper panel of the window. See figure below.

5. Select the desired channel you want to display in the section **Display** from the combo box **Channel**

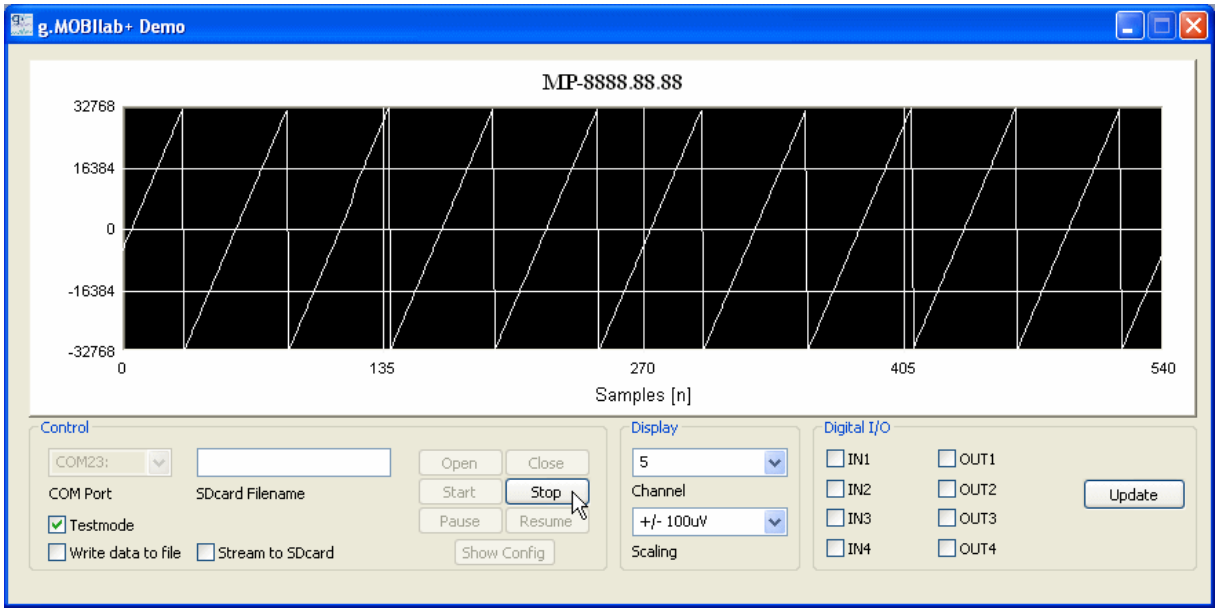


6. Use the button **Show Config** to inspect the configuration of your g.MOBIlab+. It is possible to store this configuration to a file (`gMOBIlabConfig.cfg` is the default filename) or copy it by drag & drop to a text file.

7. After initializing the Bluetooth or serial connection, you can choose between different operation modes of g.MOBIlab+. To test the connection and data transmission perform these steps:

    - Select the Checkbox **Testmode** in the Control section to run g.MOBIlab+ in testmode and start the data acquisition by clicking the **Start** button. You will see a saw tooth signal with different slopes depending on the channel selected to be displayed. The saw tooth signal is an integer consisting of 16 bit running from -32768 to +32767, see scaling in the figure below.

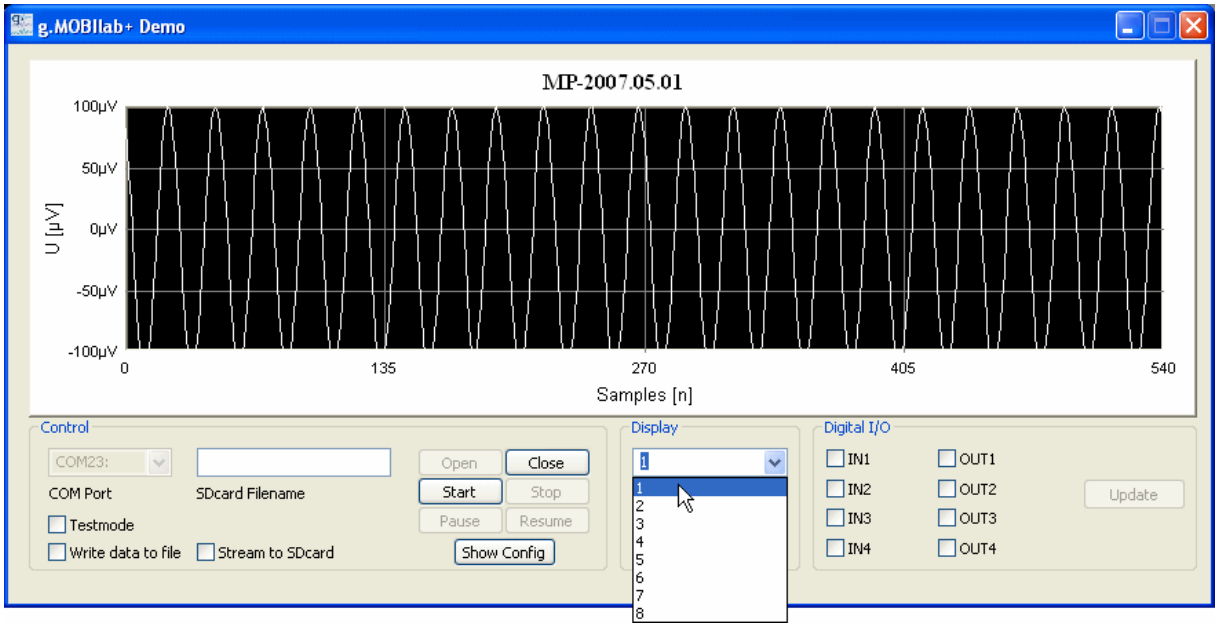
    - Use the **Stop** button to stop the data acquisition.



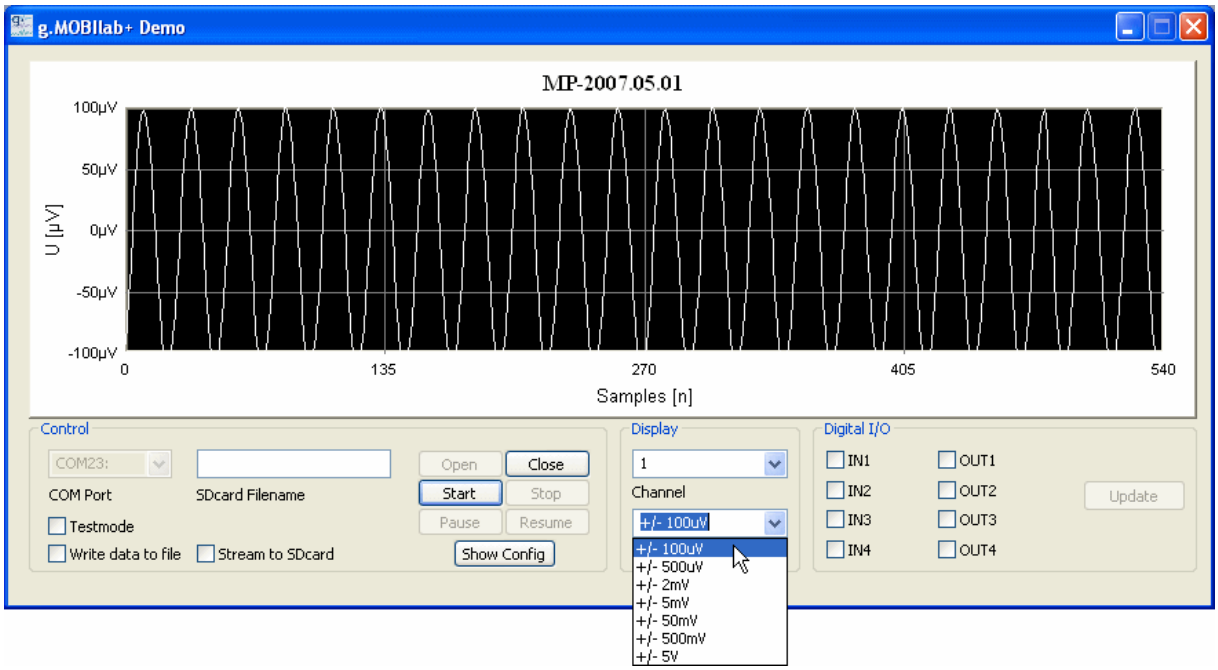
8. Finally use the **Close** button for closing the Bluetooth or serial connection.

SELECTING HARDWARE CHANNELS AND ANALOG CHANNEL SCALING

1. Select the desired analog channel to be displayed in the demo application from the combo box **Channel** in the **Display** section. You can choose one of the eight analog channels provided by g.MOBIlab+.



2. In the “Display” section the scaling for the analog channel to be displayed can be selected in the combo box “Scaling”.

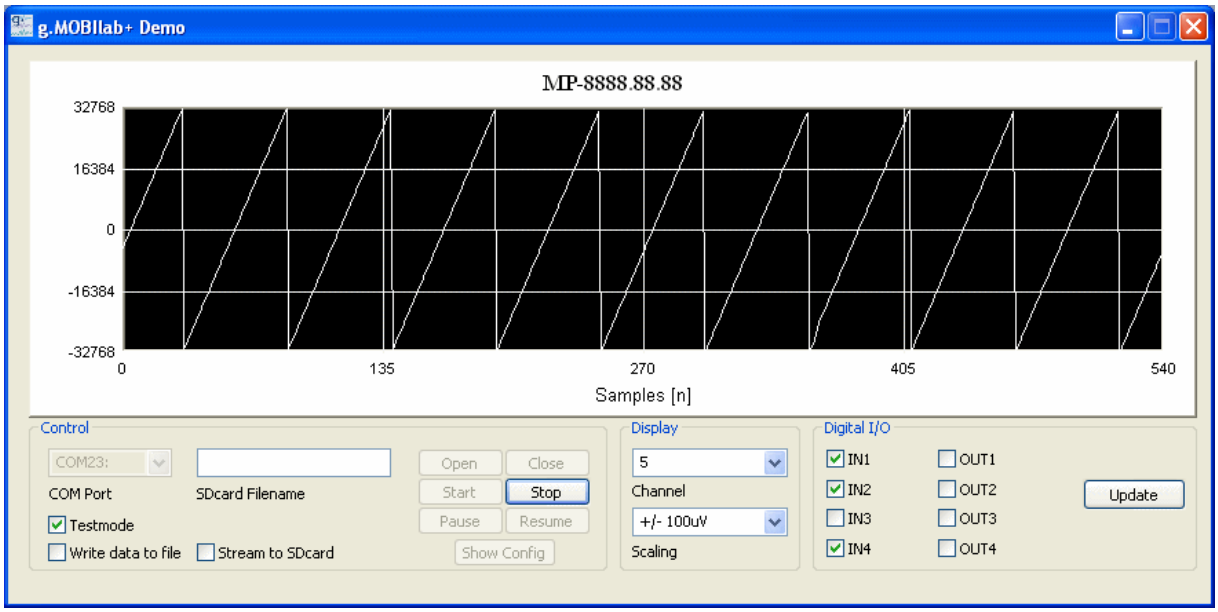


SETTING DIGITAL CHANNELS

1. g.MOBIlab+ provides eight digital channels. In the g.MOBIlab+ demo program the digital I/Os are set to be outputs (**OUT1** to **OUT4**) and the states of the digital inputs are displayed with the checkboxes **IN1** to **IN4**.

2. With the four check boxes **OUT1** to **OUT4** in the **Digital I/O** section the digital outputs can be set. To send the digital output settings to the g.MOBIlab+ hardware click the **Update** button in the **Digital I/O** section.

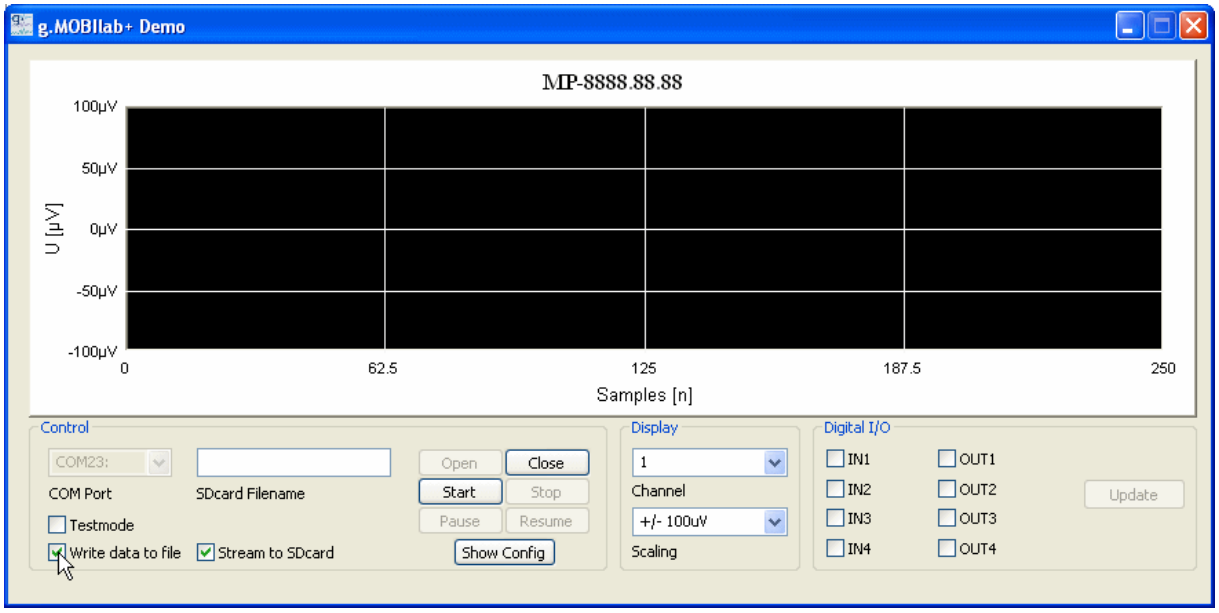
3. By clicking the **Update** button the four check boxes **IN1** to **IN4** show the current state of the four digital inputs.



Note that in testmode the four digital inputs are toggled internally by g.MOBIlab+. Digital inputs and outputs cannot be set by the user in this mode.

STORING DATA, PAUSE and RESUME DATA ACQUSITION

The g.MOBIlab+ demo application provides two options for storing data: (1) You can store the data transmitted from g.MOBIlab+ directly on your PC or (2) you can select to store data acquired by g.MOBIlab+ to the implemented SD card on the g.MOBIlab+ hardware.



1. Select the check box **Write data to file**. A file dialog will open when you click the **Start** button, which asks for an appropriate file name e.g. gMOBIlabData.bin, which is the default filename. The data of all channels will be stored to this file in the g.MOBIlab+ file format. For more information about the file format please refer to the "`gMOBIlabplusDataFormat.pdf`" found on the g.MOBIlab+ CD.

2. Select the check box **Stream to SDcard** to enable streaming to SD card.

    - Using this function a filename in the **SDcard Filename** text box must be provided. This filename consists of numbers or eight upper case characters.

    - When the acquisition is started, this checkbox selected will enable the **Pause** and the **Stop** button. Clicking on the **Pause** button during data acquisition stops the data transmission to the PC but g.MOBIlab+ continues to stream data to SD card in the same file format as in the first method to record data.

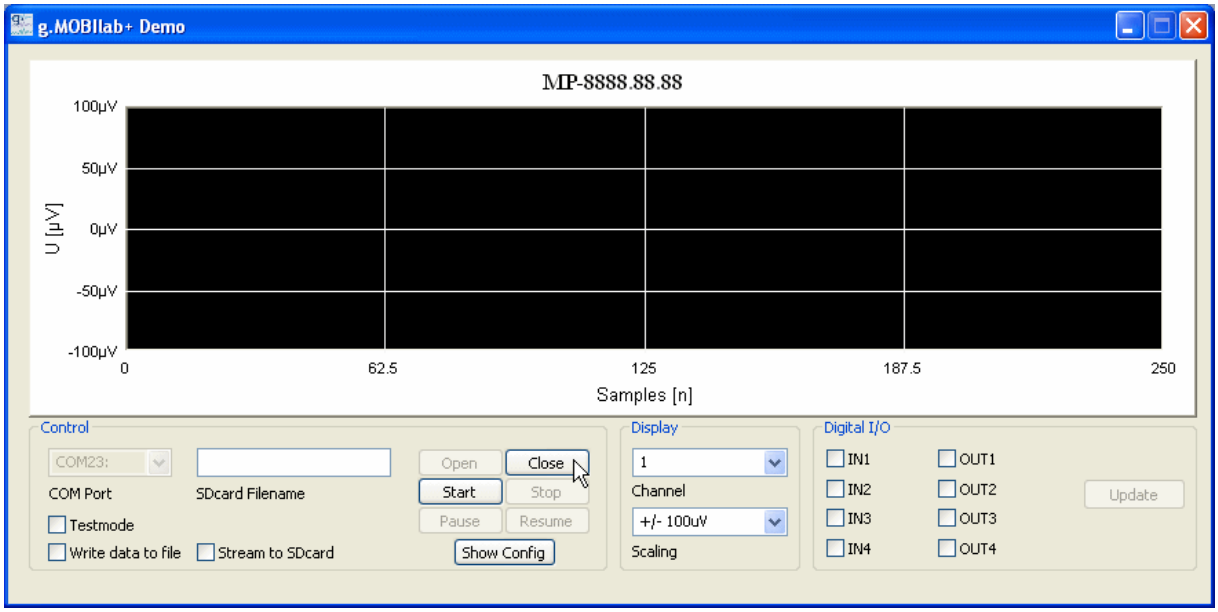
    - If **Pause** is clicked, the **Resume** button will be enabled, which allows the user to resume data transmission from g.MOBIlab+.

Both storing methods can be used at the same time.

Press **Stop** to stop the acquisition.

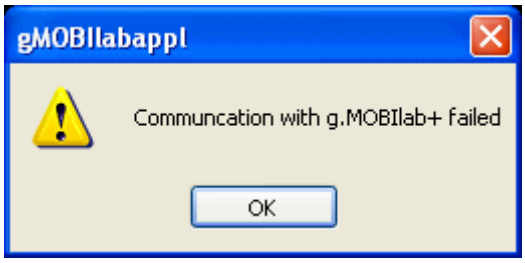
CLOSING gMOBIlabPCdemo.exe

- Press **Stop** if an acquisition is running
- Press **Close** to disconnect from g.MOBIlab+ and to free resources
- Press the cross icon on dialog window to close the demo application



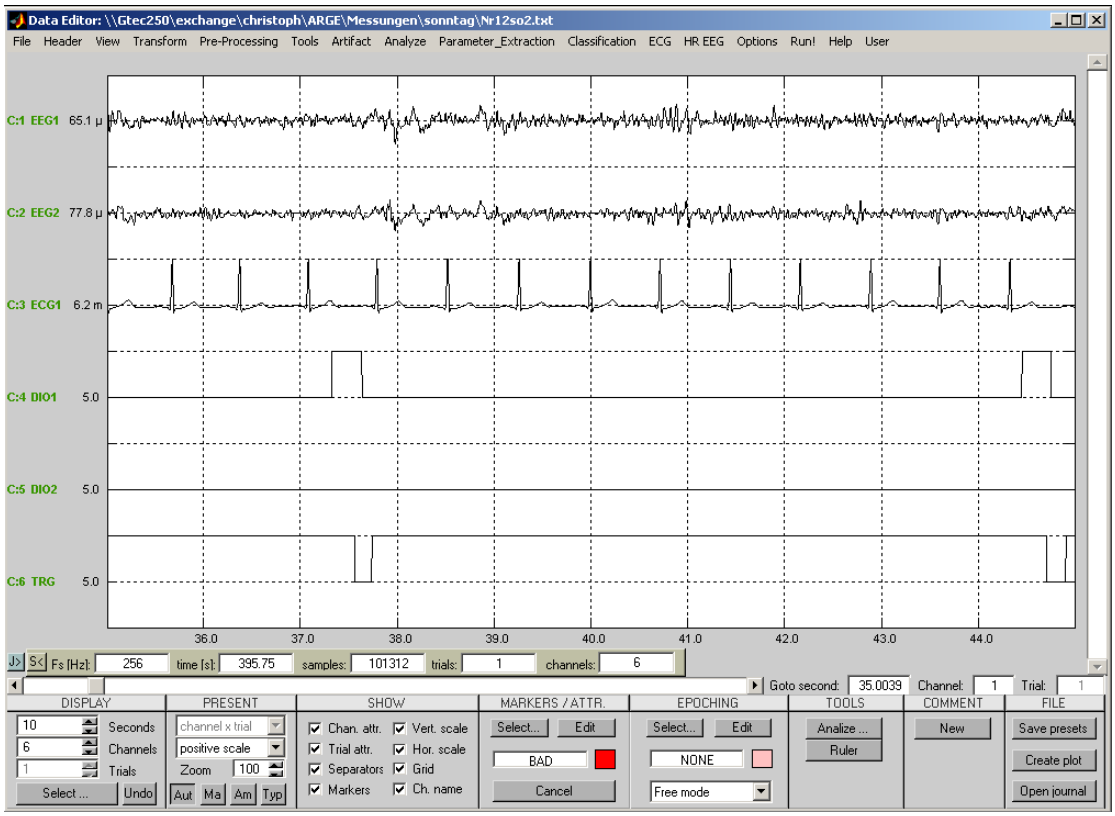
ERROR MESSAGES

Error messages are displayed in message boxes. Click **OK** to proceed.



VIEWING DATA

Use Microsoft Excel or *g.BSanalyze* to visualize the raw data or use any text editor to view signal values.



Viewing Data using g.BSanalyze

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 Visual C++ applications. Please use the provided sample code as a reference.

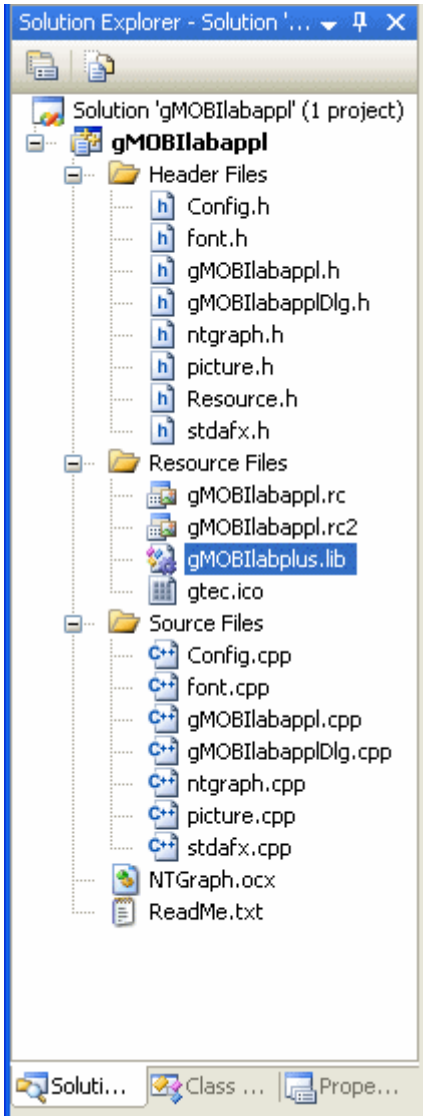
INCLUDING THE C API IN YOUR APPLICATION

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\PC\Lib
```

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

- Include gMOBIlabplus.lib e.g. in the section Resource Files in your Visual C++ development environment



- 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+. Please refer to the sample code.

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 1, 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 and initializes 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 delete this handle at the end of your application by calling GT_CloseDevice

In the example below the following functions are used:

GT_GetDeviceStatus
GT_GetSDcardStatus
GT_GetConfig

They are highlighted in the example code below.

Example:

This function opens the connection to g.MOBIlab+, detects the state of the device connected and sets the serial number of g.MOBIlab+ as display header.

```
void CgMOBIlabapplDlg::OnBOpen()
{
	//Function called when the Open button is clicked

	BOOL ret = FALSE;
	UINT * ErrCode = new UINT;
	int i = 0;
	CString portstr;
	size_t c;
	HANDLE hDevice = NULL;
	__DEVICESTATE * DevStatus = new __DEVICESTATE;
	char ser = 'M';
	char str[3];
	UINT * CardSize = new UINT[sizeof(UINT)];
	__CFG * config = new __CFG;
	CString cfgcapt;

	GetDlgItem(IDC_OPENDEV)->EnableWindow(FALSE);

	i = m_cbPort.GetCurSel();
	m_cbPort.GetLBText(i,portstr);
	portstr.Insert(0,L"\\\\.\\");
	i = portstr.GetLength();
	portstr.Insert(i-1,L'\0');
	char * prtadd = new char[i];
	wcstombs_s(&c,prtadd,i,portstr.GetString(),i);

	//open the connnection to g.MOBIlab+
	hDevice = GT_OpenDevice(prtadd);

	if (hDevice == NULL)
	{
		ErrorHandling();
		GetDlgItem(IDC_OPENDEV)->EnableWindow(TRUE);
		return;
	}

	m_hDevice = hDevice;
```

```
str[0] = '0';
str[1] = '1';
str[2] = 's';

//get the size of the SDcard. If none is inserted,
//Stream to SD is disabled
ret = GT_GetSDcardStatus(m_hDevice, CardSize);
if (!ret)
{
      CloseHandle(m_hDevice);
      m_hDevice = NULL;
      GetDlgItem(IDC_OPENDEV)->EnableWindow(TRUE);
      ErrorHandling();
      return;
}

if (*CardSize != 0)
{
      GetDlgItem(IDC_STREAM2SD)->EnableWindow(TRUE);
}
else
{
      CheckDlgButton(IDC_STREAM2SD,BST_UNCHECKED);
      GetDlgItem(IDC_STREAM2SD)->EnableWindow(FALSE);
}

//get the status of the device, to see if device is running
ret = GT_GetDeviceStatus(m_hDevice, DevStatus);
if (!ret)
{
      GT_GetLastError(ErrCode);
      if (*ErrCode == 15)
      {}
      else
      {
            CloseHandle(m_hDevice);
            m_hDevice = NULL;
            GetDlgItem(IDC_OPENDEV)->EnableWindow(TRUE);
            ErrorHandling();
            return;
      }
}

if (DevStatus->serial[0] == ser)
{
      if ((DevStatus->SDstate == str[0]) || (DevStatus->SDstate ==
      str[1]))
      {
            GetDlgItem(IDC_CLOSEDEV)->EnableWindow(TRUE);
            GetDlgItem(IDC_STARTACQ)->EnableWindow(TRUE);
            GetDlgItem(IDC_STOPACQ)->EnableWindow(FALSE);
            GetDlgItem(IDC_PAUSEXFER)->EnableWindow(FALSE);
            GetDlgItem(IDC_RESUMEXFER)->EnableWindow(FALSE);
      }

      if (DevStatus->SDstate == str[1])
      {
            GetDlgItem(IDC_STREAM2SD)->EnableWindow(TRUE);
      }

      if (DevStatus->SDstate == str[2])
```

```
		{
			GetDlgItem(IDC_CLOSEDEV)->EnableWindow(TRUE);
			GetDlgItem(IDC_STARTACQ)->EnableWindow(FALSE);
			GetDlgItem(IDC_STOPACQ)->EnableWindow(FALSE);
			GetDlgItem(IDC_PAUSEXFER)->EnableWindow(FALSE);
			GetDlgItem(IDC_RESUMEXFER)->EnableWindow(TRUE);
		}
	}
	else
	{
		GetDlgItem(IDC_CLOSEDEV)->EnableWindow(TRUE);
		GetDlgItem(IDC_STARTACQ)->EnableWindow(TRUE);
		GetDlgItem(IDC_STOPACQ)->EnableWindow(FALSE);
		GetDlgItem(IDC_PAUSEXFER)->EnableWindow(FALSE);
		GetDlgItem(IDC_RESUMEXFER)->EnableWindow(FALSE);
		GetDlgItem(IDC_STREAM2SD)->EnableWindow(FALSE);
	}

	m_hEvent = CreateEvent(NULL,FALSE,FALSE,NULL);
	m_DataDrawnow = CreateEvent(NULL,FALSE,FALSE,NULL);

	GetDlgItem(IDC_SHOWCONFIG)->EnableWindow(TRUE);
	GetDlgItem(IDC_COMPORT)->EnableWindow(FALSE);


	CWnd *pWnd = GetDlgItem(IDD_GMOBILABAPPL_DIALOG);

	//read config to print serial number on the diagram
	ret = GT_GetConfig(m_hDevice, config);
	if (!ret)
	{
		CloseHandle(m_hDevice);
		m_hDevice = NULL;
		GetDlgItem(IDC_OPENDEV)->EnableWindow(TRUE);
		ErrorHandling();
		return;
	}

	cfgcapt = config->serial;

	m_Display.SetCaption(cfgcapt);

	for (i=0;i<8;i++)
	{
		m_ChanScale[i] = config->channels[i].sensitivity;
	}
	delete config;
	delete DevStatus;
	delete CardSize;
	delete prtadd;
	delete ErrCode;

}
```

See also "gMOBIlabplusPCdemoDlg.cpp" in your demo project.

GT_GetDeviceStatus

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

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

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 will return false.

Use GT_GetLastError to retrieve error code. If g.MOBIlab+ is not in streaming mode, the error code returned is 15.

See GT_OpenDevice example code.

GT_GetSDcardStatus

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

Reads the remaining size on the SD card. 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 size on the SD card

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

Use GT_GetLastError to retrieve error code

See GT_OpenDevice example code.

GT_GetConfig

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

Reads 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 will return false.

Use GT_GetLastError to retrieve error code.

See GT_OpenDevice example 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 will return false.

Use GT_GetLastError to retrieve error code

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

In the example below the following functions are used too:

GT_SetTestmode
GT_EnableSDcard
GT_SetFilename

They are highlighted in the example code below.

Example:

This function sets the device to testmode if selected, enables the SDcard, sets the filename for the file recorded to the SDcard and initializes the channels selected for scanning.

```
void CgMOBIlabapplDlg::OnBStart()
{
	//function called when the Start button is clicked

	BOOL ret;
	_AIN aCh;
	_DIO dCh;
	CString str;
	int length,i;
	size_t c;
	char headerstart[600];
	char * filename = new char[9];
	__CFG * cfg = new __CFG;

	//set the channels used and the directions of the DIOS
	aCh.ain1 = TRUE;
	aCh.ain2 = TRUE;
	aCh.ain3 = TRUE;
	aCh.ain4 = TRUE;
	aCh.ain5 = TRUE;
	aCh.ain6 = TRUE;
	aCh.ain7 = TRUE;
	aCh.ain8 = TRUE;

	dCh.dio1_enable = TRUE;
	dCh.dio2_enable = TRUE;
	dCh.dio3_enable = TRUE;
	dCh.dio4_enable = TRUE;
	dCh.dio5_enable = TRUE;
```

```
dCh.dio6_enable = TRUE;
dCh.dio7_enable = TRUE;
dCh.dio8_enable = TRUE;

dCh.dio4_direction = FALSE;
dCh.dio5_direction = FALSE;
dCh.dio6_direction = FALSE;
dCh.dio7_direction = FALSE;

UpdateData(TRUE);

GetDlgItem(IDC_SHOWCONFIG)->EnableWindow(FALSE);

m_DispChannel = m_cbChanSel.GetCurSel();

//get the filename provided in the textbox
m_Filename.GetWindowTextW(str);
length = str.GetLength();
if (length>8)
{
      CString message;
      message = "Number of characters in Filename > 8.";
      MessageBox(message.GetBuffer(),caption.GetBuffer(),MB_OK |
      MB_ICONERROR);
      return;
}
else if ((length==0)&&(m_Stream2SD))
{
      CString message;
      message = "No filename provided.";
      MessageBox(message.GetBuffer(),caption.GetBuffer(),MB_OK |
      MB_ICONERROR);
      return;
}

wcstombs_s(&c,filename,9,str.GetString(),9);

//Set device to testmode if selected
ret = GT_SetTestmode(m_hDevice, m_Testmode);
if (!ret)
{
      ErrorHandling();
      return;
}

//Enable the SDcard if inserted into g.MOBIlab+
ret = GT_EnableSDcard(m_hDevice, m_Stream2SD);
if ((!ret)&&(m_Stream2SD))
{
      ErrorHandling();
      return;
}
if(m_Stream2SD)
{
      //Set the filename for the file recorded to SD card
      ret = GT_SetFilename(m_hDevice, filename, length);
      if (!ret)
      {
            ErrorHandling();
            return;
      }
}
delete filename;
```

```
//Set Filename to write data to file
if (m_Write2File)
{
	CString filestr;
	CString fileext;
	CString fileopt;
	CString defFilename;
	CString filedir;
	defFilename = "gMOBIlabData.bin";
	fileopt = "g.MOBIlab+ Files (*.bin)|*.bin||";
	fileext = "bin";
	filedir = "C:\\";
	CFileDialog Filename(FALSE,fileext,defFilename,OFN_HIDEREADONLY
	| OFN_OVERWRITEPROMPT,fileopt,NULL,0);
	CFileException fileException;

	Filename.GetOFN().lpstrInitialDir = filedir;

	if (Filename.DoModal() == IDOK)
	{
		filestr = Filename.GetPathName();
		if (!DataFile.Open(filestr, DataFile.modeCreate |
		DataFile.modeWrite, &fileException))
		{
			CString message;
			message = "Can not open selected file";
			MessageBox(message.GetBuffer(),caption.GetBuffer(),
			MB_OK | MB_ICONERROR);
		}
	}
	else
	{
		CString message;
		message = "No file selected";
		MessageBox(message.GetBuffer(),caption.GetBuffer(),MB_OK|
		MB_ICONERROR);
		return;
	}

	//read config to write it to the bin file header
	ret = GT_GetConfig(m_hDevice,cfg);
	if(!ret)
	{
		DataFile.Close();
		ErrorHandling();
		return;
	}

	length =
	sprintf_s(headerstart,sizeof("gtec\r\ngMOBIlab+\r\n3.0\r\n256\r
	\n"),"gtec\r\ngMOBIlab+\r\n3.0\r\n256\r\n");

	length += sprintf_s(headerstart +
	length,sizeof("%i%i%i%i%i%i%i%i%i%i%i%i%i%i%i%i%i%i%i%i%i%i
	\r\n%i\r\n%i\r\n%d\r\n%s\r\n"),"%i%i%i%i%i%i%i%i%i%i%i%i%i%i%i%
	i%i%i%i%i%i%i%i%i\r\n%i\r\n%i\r\n%d\r\n%s\r\n",aCh.ain8,aCh.ain
	7,aCh.ain6,aCh.ain5,aCh.ain4,aCh.ain3,aCh.ain2,aCh.ain1,dCh.dio
	8_enable,dCh.dio7_enable,dCh.dio6_enable,dCh.dio5_enable,dCh.di
	o4_enable,dCh.dio3_enable,dCh.dio2_enable,dCh.dio1_enable,1,dCh
	.dio7_direction,dCh.dio6_direction,dCh.dio5_direction,dCh.dio4_
	direction,1,1,1,3,2,cfg->version,cfg->serial);
```

```
	for (i=0;i<8;i++)
	{
		length += sprintf_s(headerstart + length,50,
			"%g/%g/%g/%g/%c\r\n",cfg->channels[i].highpass,
			cfg->channels[i].lowpass,cfg-
			>channels[i].sensitivity,
			cfg->channels[i].samplerate,cfg-
			>channels[i].polarity);
	}

	length += sprintf_s(headerstart +
	length,sizeof("EOH\r\n"),"EOH\r\n");
	DataFile.Write(headerstart,(UINT)strlen(headerstart));
}

//initialize the channels set to be recorded
ret = GT_InitChannels(m_hDevice, aCh, dCh);
if (!ret)
{
	ErrorHandling();
	return;
}

//set the display to be have width of 540 datapoints
int dispPoints = 540;
InitDisplay(dispPoints);

restart = FALSE;
_isrunning = TRUE;
_isdrawing = TRUE;

//create worker threads to record and draw data
m_hDataAcqTh = CreateThread(  NULL,// pointer to security attributes
                              0,// initial thread stack size
                              DataAcqTh,// pointer to thread function
                              this,// argument for new thread
                              CREATE_SUSPENDED,// creation flags
                              &_tid);// pointer to receive thread ID
SetThreadPriority(m_hDataAcqTh, THREAD_PRIORITY_TIME_CRITICAL );
//THREAD_PRIORITY_HIGHEST

m_hDataDrawTh = CreateThread( NULL,// pointer to security attributes
                              0,// initial thread stack size
                              DataDrawTh,// pointer to thread function
                              this,// argument for new thread
                              CREATE_SUSPENDED,// creation flags
                              &_tid);// pointer to receive thread ID
SetThreadPriority(m_hDataDrawTh, THREAD_PRIORITY_ABOVE_NORMAL );


pfDrawBuffer = NULL;
ResumeThread(m_hDataDrawTh); // start this thread first
ResumeThread(m_hDataAcqTh);

GetDlgItem(IDC_CLOSEDEV)->EnableWindow(FALSE);
GetDlgItem(IDC_OPENDEV)->EnableWindow(FALSE);
GetDlgItem(IDC_STARTACQ)->EnableWindow(FALSE);
GetDlgItem(IDC_STOPACQ)->EnableWindow(TRUE);
GetDlgItem(IDC_UPDATE)->EnableWindow(TRUE);
GetDlgItem(IDC_CHANNEL)->EnableWindow(FALSE);
GetDlgItem(IDC_SCALING)->EnableWindow(FALSE);
GetDlgItem(IDC_WRITEFILE)->EnableWindow(FALSE);
GetDlgItem(IDC_STREAM2SD)->EnableWindow(FALSE);
```

```
	GetDlgItem(IDC_FILENAME)->EnableWindow(FALSE);
	GetDlgItem(IDC_TESTMODE)->EnableWindow(FALSE);

	if(m_Stream2SD)
	{
		GetDlgItem(IDC_PAUSEXFER)->EnableWindow(TRUE);
	}
	else
	{
		GetDlgItem(IDC_PAUSEXFER)->EnableWindow(FALSE);
	}
	GetDlgItem(IDC_RESUMEXFER)->EnableWindow(FALSE);

}
```

See “gMOBIlabplusPCdemoDlg.cpp” in your demo project.

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 will return false.

Use GT_GetLastError to retrieve error code.

See GT_InitChannels example 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 will return false.

Use GT_GetLastError to retrieve error code.

See GT_InitChannels example 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 upper case characters.

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 will return false.

Use GT_GetLastError to retrieve error code.

See GT_InitChannels example 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 will return false.

Use GT_GetLastError to retrieve error code

Do not call any other C API functions in this data acquisition function.

Example:

The worker thread below extracts acquired data from the driver and writes it to file in int16 format and displays one channel in the demo application.

```
ULONG WINAPI DataAcqTh(void *pArg)
{
	//Worker thread to record data
	CgMOBIlabapplDlg *applptr = (CgMOBIlabapplDlg*) pArg;
	BOOL ret = FALSE;
	short buffer[540];
	OVERLAPPED ov;
	DWORD dwBytesReceived;
	short *pfl = NULL;
	int dispChannel = 0;

	ov.hEvent = applptr->m_hEvent;
	ov.Offset = 0;
	ov.OffsetHigh = 0;

	short *pDrawBuffer = new short[BUFFERLENGTH/2];
	_BUFFER_ST datbuffer;

	applptr->pfDrawBuffer = pDrawBuffer;
	datbuffer.validPoints = 0;
	datbuffer.size = sizeof(buffer);
	datbuffer.pBuffer = &buffer[0];

	if (!restart)
	{
		//Starts data acqusition
		ret = GT_StartAcquisition(applptr->m_hDevice);
		if (ret == FALSE)
		{
			applptr->ErrorHandling();
			return 0xdead;
		}
	}

	while (applptr->_isrunning)
	{
		//retrieves data from g.MOBIlab+
		ret = GT_GetData(applptr->m_hDevice, &datbuffer, &ov);
		WaitForSingleObject(applptr->m_hEvent,1000);
```

```
            GetOverlappedResult(applptr-
            >m_hDevice,&ov,&dwBytesReceived,FALSE);

            datbuffer.validPoints = dwBytesReceived/2;

            if (!ret || (dwBytesReceived != datbuffer.size))
            {
                  applptr->SendDlgItemMessage(IDC_STOPACQ,BN_CLICKED,0,0);
                  applptr->_isrunning = FALSE;
                  break;
            }
            //write data to the file on the PC
            if (applptr->m_Write2File)
            {
                  applptr->DataFile.Write(buffer,sizeof(buffer));
            }

            //set buffer to draw the selected channel
            //and set digital channel data
            dispChannel = applptr->m_DispChannel;
            for (int i=0;i<BUFFERLENGTH/2/NumOfChannels;i++)
            {
                  pfl = &buffer[dispChannel + i*NumOfChannels];
                  pDrawBuffer[i] = *pfl;
                  buffer[8+9*(i-1)] & 0x0001 ? applptr->in1 = TRUE :
                  applptr->in1 = FALSE;
                  buffer[8+9*(i-1)] & 0x0008 ? applptr->in2 = TRUE :
                  applptr->in2 = FALSE;
                  buffer[8+9*(i-1)] & 0x0002 ? applptr->in3 = TRUE :
                  applptr->in3 = FALSE;
                  buffer[8+9*(i-1)] & 0x0080 ? applptr->in4 = TRUE :
                  applptr->in4 = FALSE;
            }

            //Call draw thread
            SetEvent(applptr->m_DataDrawnow);


      }

      delete[] pDrawBuffer;
      applptr->pfDrawBuffer = NULL;
      applptr->DataFile.Abort();

      return 0xdead;

}
```

See also “gMOBIabplusPCdemoDlg.cpp” in your demo project.

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 buffer
- LPOVERLAPPED lpOvl: OVERLAPPED structure

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 will return false.

Use GT_GetLastError to retrieve error code

See GT_StartAcquisition example 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 bits to control the four digital I/Os. Bit 7 to bit 4 “1” set the DIO4 to DIO7 to be set, “0” is leave unchanged. Bit 3 to bit 0 (“1” for High, “0” for Low) represent the value of DIO4 to DIO7.

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

Use GT_GetLastError to retrieve error code

Example:

When the Update button is clicked, the digital outputs of g.MOBIlab+ are set to the set values.

```
void CgMOBIlabapplDlg::OnBUpdate()
{
	//called when Update button for DIOs is clicked

	BOOL ret;
	UCHAR Diout = 0;

	UpdateData(TRUE);

	m_in1 = in1;
	m_in2 = in2;
	m_in3 = in3;
	m_in4 = in4;

	if (m_out1)
	{
		Diout += 136;
	}
	else if (!m_out1)
	{
		Diout +=128;
	}
	if (m_out2)
	{
		Diout += 68;
	}
	else if (!m_out2)
	{
		Diout += 64;
	}
	if (m_out3)
	{
		Diout += 34;
	}
	else if (!m_out3)
	{
		Diout +=32;
	}
	if (m_out4)
```

```
	{
		Diout += 17;
	}
	else if (!m_out4)
	{
		Diout += 16;
	}

	ret = GT_SetDigitalOut(m_hDevice, Diout);
	if(!ret)
	{
		ErrorHandling();
		return;
	}

	UpdateData(FALSE);
}
```

See also "gMOBIlabplusPCdemoDlg.cpp" in your demo project.

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 will return false.
Use GT_GetLastError to retrieve error code.

Example:
When the Pause button is clicked the data transfer from g.MOBIlab+ to the PC is stopped and the worker threads are closed.

```
void CgMOBIlabapplDlg::OnBPause()
{
	//Called when the Pause button is clicked

	BOOL ret;

	//Pauses data transfer to the PC
	ret = GT_PauseXfer(m_hDevice);
	if (!ret)
	{
		ErrorHandling();
		return;
	}

	_isrunning = FALSE;
	_isdrawing = FALSE;
	m_Write2File = FALSE;
	UpdateData(FALSE);

	GetDlgItem(IDC_CLOSEDEV)->EnableWindow(FALSE);
	GetDlgItem(IDC_OPENDEV)->EnableWindow(FALSE);
	GetDlgItem(IDC_STARTACQ)->EnableWindow(FALSE);
	GetDlgItem(IDC_STOPACQ)->EnableWindow(FALSE);
	GetDlgItem(IDC_PAUSEXFER)->EnableWindow(FALSE);
	GetDlgItem(IDC_RESUMEXFER)->EnableWindow(TRUE);
	GetDlgItem(IDC_WRITEFILE)->EnableWindow(FALSE);
	GetDlgItem(IDC_UPDATE)->EnableWindow(FALSE);
	GetDlgItem(IDC_CHANNEL)->EnableWindow(TRUE);
	GetDlgItem(IDC_SCALING)->EnableWindow(TRUE);
	GetDlgItem(IDC_TESTMODE)->EnableWindow(TRUE);

}
```

See “gMOBIlabplusPCdemoDlg.cpp” in your demo project.

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 will return false.

Use GT_GetLastError to retrieve error code.

Example:

When the Resume button is clicked the Display is initialized to draw data, the data transfer from g.MOBIlab+ to the PC is resumed and the worker threads for data acquisition and data drawing are started.

```
void CgMOBIlabapplDlg::OnBResume()
{
	//Called when Resume button is clicked

	BOOL ret;
	UpdateData(TRUE);

	m_DispChannel = m_cbChanSel.GetCurSel();

	int dispPoints = 540;
	InitDisplay(dispPoints);

	//Resume data transfer to the PC
	ret = GT_ResumeXfer(m_hDevice);
	if (!ret)
	{
		ErrorHandling();
		return;
	}

	restart = TRUE;
	_isrunning = TRUE;
	_isdrawing = TRUE;

	m_hDataAcqTh = CreateThread( NULL,// pointer to security attributes
				     0,// initial thread stack size
				     DataAcqTh,// pointer to thread function
				     this,// argument for new thread
				     CREATE_SUSPENDED,// creation flags
				     &_tid);// pointer to receive thread ID
	SetThreadPriority(m_hDataAcqTh, THREAD_PRIORITY_TIME_CRITICAL );
	//THREAD_PRIORITY_HIGHEST

	m_hDataDrawTh = CreateThread(NULL,// pointer to security attributes
				     0,// initial thread stack size
				     DataDrawTh,// pointer to thread function
				     this,// argument for new thread
				     CREATE_SUSPENDED,// creation flags
				     &_tid);// pointer to receive thread ID
```

```
        SetThreadPriority(m_hDataDrawTh, THREAD_PRIORITY_ABOVE_NORMAL );


        pfDrawBuffer = NULL;
        ResumeThread(m_hDataDrawTh); // start this thread first
        ResumeThread(m_hDataAcqTh);

        GetDlgItem(IDC_CLOSEDEV)->EnableWindow(FALSE);
        GetDlgItem(IDC_OPENDEV)->EnableWindow(FALSE);
        GetDlgItem(IDC_STARTACQ)->EnableWindow(FALSE);
        GetDlgItem(IDC_STOPACQ)->EnableWindow(TRUE);
        GetDlgItem(IDC_PAUSEXFER)->EnableWindow(TRUE);
        GetDlgItem(IDC_RESUMEXFER)->EnableWindow(FALSE);
        GetDlgItem(IDC_UPDATE)->EnableWindow(TRUE);
        GetDlgItem(IDC_CHANNEL)->EnableWindow(FALSE);
        GetDlgItem(IDC_SCALING)->EnableWindow(FALSE);
        GetDlgItem(IDC_TESTMODE)->EnableWindow(FALSE);

}
```

See “gMOBIlabplusPCdemoDlg.cpp” in your demo project.

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 will return false.

Use GT_GetLastError to retrieve error code

Example:

Stops data acquisition, streaming data to SD card and data transmission to the PC when the Stop button is clicked.

```
void CgMOBIlabapplDlg::OnBStop()
{
	//Called when the Stop button is clicked

	_isrunning = FALSE;
	_isdrawing = FALSE;

	//stop data acquisition
	GT_StopAcquisition(m_hDevice);

	GetDlgItem(IDC_CLOSEDEV)->EnableWindow(TRUE);
	GetDlgItem(IDC_OPENDEV)->EnableWindow(FALSE);
	GetDlgItem(IDC_STARTACQ)->EnableWindow(TRUE);
	GetDlgItem(IDC_STOPACQ)->EnableWindow(FALSE);
	GetDlgItem(IDC_PAUSEXFER)->EnableWindow(FALSE);
	GetDlgItem(IDC_RESUMEXFER)->EnableWindow(FALSE);
	GetDlgItem(IDC_SHOWCONFIG)->EnableWindow(TRUE);
	GetDlgItem(IDC_WRITEFILE)->EnableWindow(TRUE);
	GetDlgItem(IDC_UPDATE)->EnableWindow(FALSE);
	GetDlgItem(IDC_CHANNEL)->EnableWindow(TRUE);
	GetDlgItem(IDC_SCALING)->EnableWindow(TRUE);
	GetDlgItem(IDC_STREAM2SD)->EnableWindow(TRUE);
	GetDlgItem(IDC_FILENAME)->EnableWindow(TRUE);
	GetDlgItem(IDC_TESTMODE)->EnableWindow(TRUE);

}
```

See also "gMOBIlabplusPCdemoDlg.cpp" in your demo project

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 will return false.

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

Use GT_GetLastError to retrieve error code

Example:

If the device was stopped before, this function closes the connection to g.MOBIlab+.

```
void CgMOBIlabapplDlg::OnBClose()
{
	//function called when the Close button is clicked

	BOOL ret = FALSE;

	if (_isrunning)
	{
		CString message;
		message = "Stop acquisition first";
		MessageBox(message.GetBuffer(),caption.GetBuffer(),MB_OK |
		MB_ICONERROR);
		return;
	}

	//Close the connection to g.MOBIlab+
	ret = GT_CloseDevice(m_hDevice);
	if (!ret)
	{
		ErrorHandling();
		return;
	}

	m_hDevice = NULL;
	CloseHandle(m_hEvent);
	CloseHandle(m_DataDrawnow);

	CheckDlgButton(IDC_STREAM2SD,BST_UNCHECKED);

	GetDlgItem(IDC_CLOSEDEV)->EnableWindow(FALSE);
	GetDlgItem(IDC_OPENDEV)->EnableWindow(TRUE);
	GetDlgItem(IDC_STARTACQ)->EnableWindow(FALSE);
	GetDlgItem(IDC_STOPACQ)->EnableWindow(FALSE);
	GetDlgItem(IDC_PAUSEXFER)->EnableWindow(FALSE);
	GetDlgItem(IDC_RESUMEXFER)->EnableWindow(FALSE);
	GetDlgItem(IDC_SHOWCONFIG)->EnableWindow(FALSE);
	GetDlgItem(IDC_COMPORT)->EnableWindow(TRUE);
	GetDlgItem(IDC_STREAM2SD)->EnableWindow(FALSE);

}
```

See also "gMOBIlabplusPCdemoDlg.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 floating point number holding the current driver version.

See “gMOBIlabplusPCdemoDlg.cpp” in your demo project.

GT_GetLastError

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

Retrieves error code from last occurred error

Input parameters:

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

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

Example:

If a function returns FALSE, this functions reads the error code returned by the API.

```
BOOL CgMOBIlabapplDlg::ErrorHandling()
{
	//Called if an error occured in the demo program
	UINT i = 0;
	BOOL ret;
	CString ErrorStr;
	_ERRSTR strError;

	ret = GT_GetLastError(&i); // get number of last error
	GT_TranslateErrorCode(&strError,i); // get corresponding error string

	ErrorStr = strError.Error;
	MessageBox(ErrorStr,caption.GetBuffer(),MB_OK | MB_ICONERROR);

	return TRUE;
}
```

See also "gMOBIlabplusPCdemoDlg.cpp" in your demo project.

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 will return false.

See GT_GetLastError example code.

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