GM05/GM08
Gaussmeter
Programming Manual

Hirst Magnetic Instruments Ltd
Tesla House
Tregoniggie
Falmouth
Cornwall
TR11 4SN U.K.

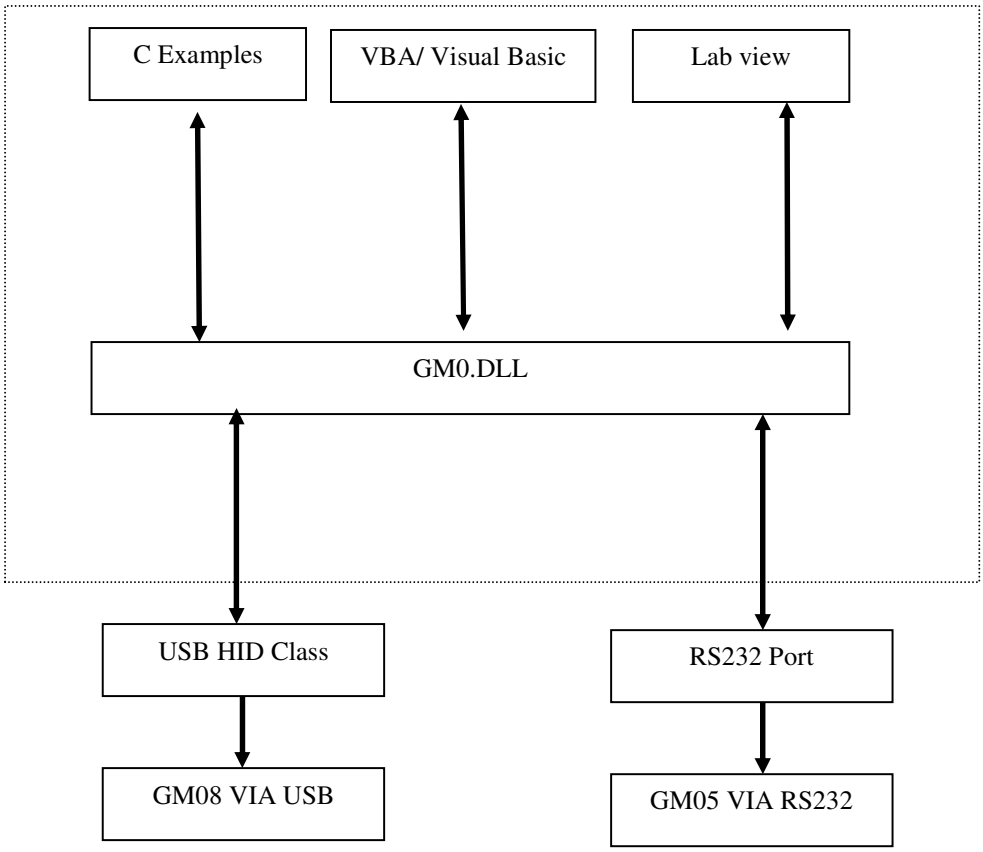
Tel: +44 (01326) 372734
Fax: +44 (01326) 378069

http://www.hirst-magnetics.com

Introduction

The driver software for the GM05/08 is presented as a dynamic link library (or shared library).

The GM0.dll is a standard MS Windows DLL (or a Unix .so library) that talks directly to the IO subsystem directly for RS232 ports and via the HID subsystem for USB ports. A Number of properties, methods and events are exposed to allow easy control of the GM05 & GM08.



It is possible to directly talk to the GM05 & GM08 via the RS232 interface without the use of the GM0.DLL and the communication protocol is documented. Due to the complexity of USB details of the USB command structure are not included in this document but can be derived from the source code to the shared library.

GM05 (& VGM01) RS232 Details

The RS232 used by the GM05 is as follows:-

Baud rate	4800
Stop Bits	0
Data Bits	8
Parity Bits	1
Hardware Handshake	NO
Software Handshake	NO

GM08 RS232 Details

The RS232 used by the GM08 is as follows:-

Baud rate	9600
Stop Bits	0
Data Bits	8
Parity Bits	1
Hardware Handshake	NO
Software Handshake	NO

RS232 Connection to PC

****GM05** It is VITAL that the interface supplied with the GM05 is used when connecting to a serial port. DAMAGE will occur to the GM05 if the interface is not used.**

The GM08 may be directly connected to a standard PC type serial port.

Standard PC 9 Way Port

Pin	Name	Function
2	RX	Data to PC
3	TX	Data from PC
5	GND	Signal Ground

Standard PC 25 Way Port

Pin	Name	Function
2	TX	Data from PC
3	RX	Data to PC
7	GND	Signal Ground

GM05 Serial Port (25 way, MUST use adapter)

Pin	Name	Function
2	TX	Data from gaussmeter
3	RX	Data to gaussmeter
7	GND	Signal Ground

GM08 Serial Port (9 way)

Pin	Name	Function
2	RX	Data to Gaussmeter
3	TX	Data from gaussmeter
7	GND	Signal Ground

To connect the GM08 to a PC 9 way serial port use a standard crossed (null modem) type cable. Pin 2 of one end should connect to pin 3 of the other and visa versa.

To connect the GM05 to a PC 9 way serial port, use a standard (straight) modem cable. Ensure that pin 2 of one end connects to pin 2 of the other

Serial command structure

The GM05 & GM08 utilises a system of packets to pass commands and data between the host and the device.

Each packet is 4 bytes long and has the following format

Host Sends CMD
GM0 Sends Status
Host Sends Data
GM0 Sends Data

Before entering command mode a '*' must be sent to the GM05 and a status byte received:-

Host Sends '*' (42)
GM0 Sends 0

To enter data mode it is necessary to send a mode 1 command. Data mode is enforce immediately after the mode 1 command set. The GM08 does not use mode 1 over USB and data must be polled, RS232 is identical format to GM05.

Host Sends 1
GM05 Sends 0
Host Sends 0
(GM05 Does not return packet data as MODE 1 is now active and data is streaming from the instrument)

GM05 Serial Commands

From the above overview it can be seen that data can flow from the GM05/08 to the host and vice versa. All data is made up of single bytes, and commands are expressed as a single byte. A description of each byte follows:

Byte	Description
Status/Ready	This byte is transmitted by the GM05/08 when it is ready to receive another command. No commands should be sent until this byte has been received. The value of the byte depends on the last command that the GM05/08 serviced. If it was successful, 0 is sent as the Status/Ready byte. If any errors occurred, bits will be set in the Statu/Ready byte indicating what error occurred. Error codes are specific to each command and are detailed in the command descriptions below.
Command	This gives the number of the command that the GM05/08 is required to perform. The commands and their numbers are given below in each command description.
Data Byte A	This is a byte of data from the GM05/08 to the host. Details of what it conveys depends on the command executed. Further information is given below.
Data Byte B	This is a byte of data from the host to the GM05/08. Details of what it conveys depends on the command executed. Further information is given below.

The serial interface commands can only exchange single bytes of information. To reduce the limits that this imposes on information exchange, the GM05/08 serial protocol has, in the GM05/08, several registers that can be used to hold values larger than one byte.

Serial Interface Commands

-GM05----------

Command Name:	Mode1
Command Number:	1
Data Byte A:	0
Data Byte B:	Ignored
Status Codes:	No status code is transmitted for this command as mode 1 communications is in force after it is executed.
Description:	This command puts sets the communications mode to 1. After data byte B has been transmitted, mode 2 communications are no longer in force. It is therefore not possible for this command to return a status code.

-GM0----------

Command Name:	Range
Command Number:	12
Data Byte A:	Current Range Settings
Data Byte B:	New Range Settings
Status Codes:	Always 0
Description:	This command is used to set the GM04 range or the range setting of the CALRIG Hall voltage simulator circuit.

Current Range Settings Byte

Bit 0,1	Current Range
Bit 2	Auto range (0-Fixed 1-Auto)

New Range Settings Byte

Bit 0,1	Current Range
Bit 2	Auto range (0-Fixed 1-Auto)
Bit 7	Read/Write (0-Read, 1-Write)

-GM0----------

Command Name:	PadDisable
Command Number:	13
Data Byte A:	Current keypad disable flag
Data Byte B:	New keypad disable flag
Status Codes:	Always 0
Description:	This command can be used to enable or disable the keypad. If the keypad disable flag is zero, the keypad is enabled, if it is non-zero, the keypad is disabled.

New KP disable flag byte:

Bit 0	Keypad disable(0-Enabled,1-Disabled)
Bit 7	Read/Write(0-Read,1-Write)

-GM0----------

Command Name: SimKey
Command Number: 14
Data Byte A: Current keycode.
Data Byte B: Key code of key to simulate
Status Codes: Always 0
Description: This command can be used to simulate keypresses and also read the keypad.
The keycodes are as follows:

Exit	'X'
Next	'N'
Off	'O'
Range	'R'
Reset	'0'
Spare	'S'
Enter	'E'

The ON key cannot be simulated.
When reading the keypad, if the MS bit is set, the key has been processed, if it is not set, the key has not yet been processed.

The kaypad cannot be read if the keypad is disabled. In order to read the keypad, set the key to simulate as 00 hex.

-GM0----------

Command Name: RetDisp
Command Number: 16
Data Byte A: 0
Data Byte B: Ignored
Status Codes: Always 0
Description: This command copies the current display into a buffer. The ADDRESS register is set up to point to the buffer.

-GM0----------

Command Name: LoBat
Command Number: 17
Data Byte A: 0
Data Byte B: The required battery measurement setting
Status Codes: Always 0
Description: This command allows the battery measurement functions to be stopped. If the data byte sent is 0 then the battery measurement will not be done, and the LOBAT symbol will not be updated.
If the data byte is 1 then the LOBAT symbol will be updated.

-GM0----------

Command Name: NA
Command Number: 18

DO NOT ATTEMPT TO USE THIS COMMAND AT RISK OF TOTAL CORRUPTION OF YOUR GAUSSMETER!

-GM0----------

Command Name:	Units
Command Number	19
Data Byte A:	Current Units Setting
Data Byte B:	New Units Setting
Status Codes:	Non-zero if Argument is out of range
Description:	This command is used to set the GM04 units

Current Units Settings Byte

Bit 0,1	Current units

New Units Setting Byte

Bit 0,1	New units
Bit 7	Read/Write (0-Read, 1-Write)

-GM0----------

Command Name:	Function
Command Number:	20
Data Byte A:	Current Function Setting
Data Byte B:	New Function Setting
Status Codes:	Non-zero if Argument is out of range
Description:	This command is used to set the GM04 function, i.e. DC, AC, DCPeak etc.

Current function Settings Byte

Bit 0,1	Current function

New function Setting Byte

Bit 0,1	New function
Bit 7	Read/Write (0-Read, 1-Write)

-GM0----------

Command Name:	Language
Command Number:	21
Data Byte A:	Current Language Setting
Data Byte B:	New Language Setting
Status Codes:	Non-zero if Argument is out of range
Description:	This command is used to set the GM04 Language.

Current Language Settings Byte

Bit 0,1,2	Current language

New Language Setting Byte

Bit 0,1,2	New language
Bit 7	Read/Write (0-Read, 1-Write)

-GM0----------

Command Name: ResPeak
Command Number: 22
Data Byte A: 0
Data Byte B: Ignored
Status Codes: 0
Description: This command initiates a peak detector reset. The peak detector takes 1 second to reset, so no actions to do with the measurement functions of the GM05 should be performed for this period after the issuing of this command.

-GM0----------

Command Name: DoNull
Command Number: 23
Data Byte A: 0
Data Byte B: Ignored
Status Codes: 0
Description: This command starts a Null function off. It returns immediately, so a delay equal to the menu 'Null' function should be inserted after issuing this command, before any measurement actions are taken.

-GM0----------

Command Name: DoAZ
Command Number: 24
Data Byte A: 0
Data Byte B: Ignored
Status Codes: 0
Description: This command starts an auto zero function off. It returns immediately, so a delay equal to the menu 'Auto Zero' function should be inserted after issuing this command, before any measurement actions are taken.

-GM0----------

Command Name: Time0
Command Number: 26
Data Byte A: The current contents of register 0 in the time buffer.
Data Byte B: The new contents of register 0 of the time buffer.
Status Codes: Non-zero if no clock detected

Description: This command sets register 0 of the time buffer. It does not set the time, use the Time command to do that.

-GM0----------

Command Name: Time1
Command Number: 27
Data Byte A: The current contents of register 1 in the time buffer.
Data Byte B: The new contents of register 1 of the time buffer.
Status Codes: Non-zero if no clock detected

Description: This command sets register 1 of the time buffer. It does not set the time, use the Time command to do that.

-GM0----------

Command Name: Time2
Command Number: 28
Data Byte A: The current contents of register 2 in the time buffer.
Data Byte B: The new contents of register 2 of the time buffer.
Status Codes: Non-zero if no clock detected

Description: This command sets register 2 of the time buffer. It does not set the time, use the Time command to do that.

-GM0----------

Command Name: Time3
Command Number: 29
Data Byte A: The current contents of register 3 in the time buffer.
Data Byte B: The new contents of register 3 of the time buffer.
Status Codes: Non-zero if no clock detected

Description: This command sets register 3 of the time buffer. It does not set the time, use the Time command to do that.

-GM0----------

Command Name: Time4
Command Number: 30
Data Byte A: The current contents of register 4 in the time buffer.
Data Byte B: The new contents of register 4 of the time buffer.
Status Codes: Non-zero if no clock detected

Description: This command sets register 4 of the time buffer. It does not set the time, use the Time command to do that.

-GM0----------

Command Name: Time5
Command Number: 31
Data Byte A: The current contents of register 5 in the time buffer.
Data Byte B: The new contents of register 5 of the time buffer.
Status Codes: Non-zero if no clock detected

Description: This command sets register 5 of the time buffer. It does not set the time, use the Time command to do that.

-GM0----------

Command Name: Time6
Command Number: 32
Data Byte A: The current contents of register 6 in the time buffer.
Data Byte B: The new contents of register 6 of the time buffer.
Status Codes: Non-zero if no clock detected

Description: This command sets register 6 of the time buffer. It does not set the time, use the Time command to do that.

-GM0----------

Command Name: Time7
Command Number: 33
Data Byte A: The current contents of register 7 in the time buffer.
Data Byte B: The new contents of register 7 of the time buffer.
Status Codes: Non-zero if no clock detected

Description: This command sets register 7 of the time buffer. It does not set the time, use the Time command to do that.

Command Name: Time
Command Number: 34
Data Byte A: Always 0
Data Byte B: A TIME command, see below
Status Codes: Non-zero if no clock detected in the GM05

Description: This command performs a TIME command as detailed below. If no clock is detected, due to a hardware error or lack of a clock device in the GM05, the returned status code is non-zero.

The time keeping device used in the GM05 is the Dallas DS1216 SmartWatch. All registers in the watch are programmable. Refer to the Dallas Semiconductor databook for details about programming this device. The GM08 uses an alternative device but the registers are presented in the same format for compatibility.

TIME Commands
==========

The TIMEn commands can read and write to an area of RAM in the GM05 called the TIME registers. These are a copy of the registers in the GM05 time keeping device. The register contents can be written into the clock device using the PUT_TIME TIME command. The clock device registers can be copied into the TIME registers using the GET_TIME command.

Value	Description
0	ONLY_RD This command prevents the TIMEn commands from writing new data into the TIME registers.
1	ALLOW_WR This command allows the TIMEn commands to write data into the TIME registers. The value returned by the TIMEn commands will be overwritten by the new value passed in the TIMEn command.
2	PUT_TIME This copies the TIME registers into the clock device.
3	GET_TIME This copies the clock device registers into the TIME registers.

-GM0----------

Command Name: SampleNow
Command Number: 35
Data Byte A: Always 0
Data Byte B: Ignored
Status Codes: Non-zero if sample couldn't be taken

Description: This command stores the current reading in the next sample register. The register pointer is incremented to point to the

next register.

-GM0----------

Command Name: CommFlag
Command Number: 36
Data Byte A: The current contents of the COMMFLAG register
Data Byte B: The new contents of the COMMFLAG register
Status Codes: 0

Description: This command sets and reads the COMMFLAG register. the bit assignments within the register is as follows:

Bit	Desc
0	0 - Mode 1 format is: SRRR.R abc 1 - Mode 1 format is: SRRR.R abc hh:mm:ss dd/mm/yy This bit controls the format of the Mode 1 comms, when set to 1 the current time and date in the GM05 is transmitted with the sample information.

-GM0----------

Command Name: GetReg
Command Number: 37
Data Byte A: 0
Data Byte B: Ignored
Status Codes: 0
Description: This command copies a block of 8 register bytes into the TIME buffer. The TIMEn commands can then be used to read the bytes out to the host.

-GM0----------

Command Name: RegPtr
Command Number: 38
Data Byte A: Current register pointer
Data Byte B: New register pointer
Status Codes: Non-zero if Argument is out of range
Description: This command is used to set the pointer to the registers that is used to determine which register the GetReg command will copy to the TIME buffer.

New register Pointer Byte

Bit 0,1,2,3,4,5,6 New Register Pointer(0-99)
Bit 7 Read/Write (Read=0,Write=1)

GM0.DLL REFERENCE

The software to drive the Hirst GM05 and VGM01 is contained with in a dynamic link library (dll) called gm0.dll or gm0D.dll for the version with debug symbols. On Unix systems the library will be called gm0.so

Dependencies

The dll depends on the following windows system files

MSVCRT.dll Microsoft's visual c runtime library
Kernel32.dll Core windows file
User32.dll Core windows file

On Unix it has no non standard dependencies.

Linkage specification

The dll is complied with all exported functions conforming to the __stdcall specification, this enables software such as Visual Basic™ (Microsoft Corp.) to access the dll.

Element	**Implementation**
Argument-passing order	Right to left.
Argument-passing convention	By value, unless a pointer or reference type is passed.
Stack-maintenance responsibility	Called function pops its own arguments from the stack.
Name-decoration convention	`No name decoration`
Case-translation convention	None
Linkage	__stdcall

This gives maximum compatibility with external programs and non c/c++ programs as it is the same as all internal Windows function calls. It is especially needed for easy linking to Visual Basic programs.

Type sizes

The dll is written in Microsoft C and the type sizes have the following incompatibility with Visual Basic. Always use long in Visual basic to receive a C int!

Data type	***MS 32bit C***	***VB 6.0***
UCHAR	1 byte	1 byte (byte in VB)
Int	4 bytes	2 bytes
long	4 bytes	4 bytes

float	4 bytes	4 bytes (single in VB)
double	8 bytes	8 bytes

Return structures used: -

```
struct gm_time{
UCHAR sec;
UCHAR min;
UCHAR hour;
UCHAR day;
UCHAR month;
UCHAR year;
};

struct gm_store{
struct gm_time time;
UCHAR range;
UCHAR mode;
UCHARunits;
float value;
};
```

Linking to user programs

Visual Studio / Microsoft Visual C Environment

To use the functionality of gm0.dll in a Visual C/C++ program make sure your program includes the gm0.h header file and make sure that your program links against the gm0.lib. All functions defined in gm0.h are they directly accessible from your Visual C/C++ program.

Note it is not necessary to use this method in visual C . You can use run time linking against the dll that is controlled by your code as per the next example.

Example, a visual studio Win32 console application

```
#include <stdio.h>
#include “gm0.h”

int main(int argv, char * argc[])
{
     HANDLEGM mygm;
     double value;
     gm_store store;

     mygm=gm0_newgm(1,0);

     gm0_range(mygm,1);

     store.value=gm0_getvalue(mygm0);
     store.range=gm0_getrange(mygm0);
     store.units=gm0_getunits(mygm0);

     value=gm0_getvalue(myifm);
     printf("Reading %lf range %d % units
%d\n",store.value,store.range,store.units);

     gm0_killgm(myifm);

     return 0;
}
```

Non Microsoft Specific C/C++ Environments (NON LINUX)

To access dll functions without the aid of the dll_import and dll_export macros present in Visual Studio it is necessary to use LoadLibrary() and GetProcAddress() functions to retrieve a pointer to the function in question. The gm0.h header file has been prepared to work with or without visual studio. It is necessary to define NOVC before including gm0.h to ensure the visual studio specific parts of the header file are turned off.

To import a function it is necessary to know the functions name and the length of its parameters, the example below shows gm0_newm, gm0_killgm etc. the function

name is taken from the gm0.h file the parameter length can be calculated by working out the type sizes of the parameters. Eg the definition for gm0_newgm is:-

GM05_API HANDLEGM LIBTYPE gm0_newgm BKET (int port);

The name is gm0_newifm, and it accepts one parameter if size int. An int is 4 bytes in "Windows 32bit" C so the exported name in the dll becomes _gm0_newifm@4 . Notice the leading underscore and the @ symbol before the parameter length.

Example

```
#include <windows.h>
#include <stdio.h>

#define NOVC /* very very very important */
#include "gm0.h"

int main(int argv, char * argvc[])
{
     HANDLEIFM mygm;
     double value;
     HMODULE lib;

     lib=LoadLibrary("gm0.dll");

     if(lib==NULL)
     {
          printf("\nCould not load gm0.dll, Exiting\n");
          exit(-1);
     }

     loadfunction(&gm0_newifm,"_gm0_newifm@4");
     loadfunction(&gm0_killifm,"_ gm0_killifm@4");
     loadfunction(&gm0_getvalue,"_ gm0_getvalue@4");


     mygm=gm0_newgm(1,0);

     store.value=gm0_getvalue(mygm);
     store.range=gm0_getrange(mygm);
     store.units=gm0_getunits(mygm);

     value=gm0_getvalue(mygm);
     printf("Reading %lf range %d % units
%d\n",store.value,store.range,store.units);

     gm0_killgm(mygm);


}
```

```
void loadfunction (void ** functionpointer,char *
functionname)
{
     (*
functionpointer)=GetProcAddress(lib,functionname);
     if(functionpointer==NULL)
     {
          printf("Error loading function %s
\n",functionname);
          exit(-1);
     }
     printf("Function %s loaded @ %d
OK\n",functionname,functionpointer);
     return;
}
```

Visual Basic or VBA Environments

To access the dll functions from visual basic make sure you include a definition for each function you use in a global section of a module :-

```
Declare Function fm0_newgm Lib "gm0.dll" Alias
"gm0_newgm" (ByVal port As Integer, ByVal mode as
Integer) As Integer
```

As per the previous section other functions can be deduced by examining the gm0.h file and adding up the parameter length. (See the previous section for details).

The functions are then accessible in visual basic as normal functions eg

```
Handle= gm0_newgm(1,0)
```

Unix Environments

The gm0.so library can be built using GNU automake/autoconf. To start the build process use “./configure” and then “make”, “make install” will also put all the libraries in the correct system location.

Compile the gm0.so library :-

```
gcc -D_LINUX -fPIC -g -c gm0.c -o gm0.o
gcc -D_LINUX -fPIC -g -c gm0_comms.c -o gm0_comms.o
gcc -D_LINUX -fPIC -g -c linuxcomms.c -o linuxcomms.o

gcc -D_LINUX -shared -Wl,-soname,libgm.so.1 -lc -lm -lpthread -o
libgm.so.1.0.0 gm0.o gm0_comms.o linuxcomms.o

cp libgm.so.1.0.0 /usr/local/lib/
ldconfig /usr/local/lib
ln -s /usr/local/lib/libgm.so.1.0.0  /usr/local/lib/libgm.so
ln -s /usr/local/lib/libgm.so.1.0.0 /usr/local/lib/libgm.so.1
```

To access any functions follow the visual studio example above then compile your program with

```
gcc yourfile.c -lgm0
```

to link against the gm0 library, no extra steps are required to load function addresses**.**

GM0.DLL API REFERENCE

Start and stop functions

HANDLEGM gm0_newgm(int port, int mode)

Parameters

`port` the serial COM port that the GM0/VGM is connected to, or -1 to connect via USB.
`mode` the connection mode to use (0 – GM05 mode, 1 – GM08 mode)

Return Values

If successful the function returns a handle that is 0 or greater.
If unsuccessful the function returns an ERROR code that is negitive

Description

This function allocates all the necessary memory and accesses the port to enable communications to a GM05/VGM01 Gaussmeter. If successful a handle is returned to the instrument. This handle must be used in all function calls. Each dll can support up to 255 separate handles subject to hardware and operating system limitations.

The GM05 and GM08 use different baud rates and this is reflected in the setting of the mode parameter.

A value of -1 for the port will request a connection via USB.

HANDLEGM gm0_startconnect(HANDLEGM hand)

Parameters

`hand` handle to instrument

Return values

Description

This function attempts to communicate to the GM05/VGM01 on the com port already initialised by gm0_newgm(). If successful a GM0_OK is returned else an error code is returned.

BOOL gm0_getconnect(HANDLEGM hand);

Parameters

hand handle to instrument

Return Values

`TRUE` for connected, `FALSE` for not connected.

Description

This function polls the driver to see if the GM0/VGM01 has successfully connected to the port. Once connected all other functions to operate the instrument are available.

`int gm0_killgm(HANDLEGM hand);`

Parameters

`hand` handle to instrument

Return Values

`status`

Description

This function frees all memory and resources and closes the connection to the GM0/VGM . This function should always be called if a gm0_newgm() was called even if it was unsuccessful.

Standard user Functions

`int gm0_setrange(HANDLEGM hand,unsigned char range);`

Parameters

`hand` handle to instrument
`range` range to be selected 0-4

Return Values

Description

Sets the current range on the GM0/VGM

`int gm0_getrange(HANDLEGM hand);`

Parameters

`hand` handle to instrument

Return Values

Range 0-3
Note GM08 Only – The return range will have 4 added to it if the meter is in auto range mode.

Description

Returns the current range of the GM0/VGM

`int gm0_setunits(HANDLEGM hand,unsigned char units);`

Parameters

`hand` handle to instrument
`units` units to be set

Return Values

Description

Sets the Gaussmeters units to the value given in `units`

TESLA	0
GAUSS	1
KAM	2
OE	3

`int gm0_getunits(HANDLEGM hand);`

Parameters

`hand` handle to instrument

Return Values

`units` the currently selected units

Description

Returns the current units of the Gaussmeter

`int gm0_setmode(HANDLEGM hand,unsigned char mode);`

Parameters

`hand` handle to instrument
`mode` require mode

Return Values

Description

Sets the Gaussmeter to the mode given by `mode`

DC	0
DCPK	1
AC	2
ACMAX	3
ACPK	4
GM_HOLD	5

`int gm0_getmode(HANDLEGM hand);`

Parameters

`hand` handle to instrument

Return Values

`mode` The current mode of the instrument

Description

Retrieves the current mode setting from the Gaussmeter

DC	0
DCPK	1
AC	2
ACMAX	3
ACPK	4
GM_HOLD	5

`int gm0_setlanguage(HANDLEGM hand,unsigned char lan);`

Parameters

`hand` handle to instrument

Return Values

Description

Sets the operating language of the GM05. Does not apply to the VGM01

ENGLISH	0
FRENCH	1
GERMAN	2
ITALIAN	3
SPANISH	4

PORTUGUESE 5

int gm0_getlanguage(HANDLEGM hand);

Parameters

`hand` handle to instrument

Return Values

Description

Returns the operating language of the GM05 gaussmeter. Does not apply to the VGM01

ENGLISH	0
FRENCH	1
GERMAN	2
ITALIAN	3
SPANISH	4
PORTUGUESE	5

double gm0_getvalue(HANDLEGM hand);

Parameters

`hand` handle to instrument

Return Values

`value` The last measured value of gaussmeter

Description

This function returns the last value acquired by the Gaussmeter.

int gm0_resetpeak(HANDLEGM hand);

Parameters

`hand` handle to instrument

Return Values

Description

This function resets the peak detector in peak and MAX modes.

Zero and null functions

int gm0_donull(HANDLEGM hand);

Parameters

`hand` handle to instrument

Return Values

Description

This function places the Gaussmeter in null mode. Please shield the probe before calling this function. The function will block until the gaussmeter has completed its null (this is done on a timer)

int gm0_doaz(HANDLEGM hand);

Parameters

`hand` handle to instrument

Return Values

Description

This function places the Gaussmeter in AutoZero mode. Please shield the probe before calling this function. The function will block until the gaussmeter has completed its null (this is done on a timer)

int gm0_resetnull(HANDLEGM hand);

Parameters

`hand` handle to instrument

Return Values

Description

This command signals the Gaussmeter that the probe is in the zero gauss can and to continue with the null or auto zero operation.

// Date time and register functions

int gm0_sendtime(HANDLEGM hand,BOOL extended);

Parameters

`hand` handle to instrument

Return Values

Description

This function enables the sending of date time information with measurement values. Only available on the GM05.

```
int  gm0_settime(HANDLEGM hand,struct gm_time);
```

Parameters

`hand`	handle to instrument
`gm_time`	a gm_time structure that contains all date and time information

Return Values

Description

```
struct gm_time  gm0_gettime(HANDLEGM hand);
```

Parameters

`hand` handle to instrument

Return Values

`gm_time`	a gm_time structure that contains all date and time information

Description

Retrevies the current on board time of the GM05

```
struct gm_store  gm0_getstore(HANDLEGM hand,int
pos);
```

Parameters

`hand`	handle to instrument
`pos`	the register to be retrevied

Return Values

`gm_store`	a gm_store structure that contains all date and time and measurement information

Description

Callback functions

`int gm0_setcallback(HANDLEGM hand, void (* pCallback)(HANDLEGM hand,struct gm_store store));`

Parameters

`hand` handle to instrument
`pCallback` address of a callback function to receive notification.

Return Values

Description

Enables callback notification for the new data event. The callback function must be declared void function(void) and be of a __stdcall linkage type. This event is fired when ever new data is available from the Gaussmeter.

`int gm0_setconnectcallback(HANDLEGM hand,void (* pCallback)(void));`

Parameters

`hand` handle to instrument
`pCallback` address of a callback function to receive notification.

Return Values

Description

Enables callback notification for the connected event. The callback function must be declared void function(void) and be of a __stdcall linkage type. This event is fired when a successful connection is established to a Gaussmeter.