Description of the DLL for the Velleman PC Function Generator PCGU1000

All the data transfer routines of the function generator are contained in a Dynamic Link Library (DLL) FGULink.DLL.
This document describes the functions and procedures of the DLL that are available for your application program. Calling the functions and procedures exported by the DLL, you may write custom Windows (98SE, 2000, Me, XP, Vista) applications in Delphi, Visual Basic or any other 32-bit Windows application development tool that supports calls to a DLL.
A complete overview of the functions and procedures that are exported by the FGULink.DLL follows. At the end of this document there are listings of example programs in order to gain an insight as to how to construct your own application programs. The examples are written in Delphi and Visual Basic. In the listings there are full declarations for the DLL functions and procedures.
Note that all the examples in the function and procedure description section are written for Delphi.

Microsoft Visual Studio .NET users please note: The FGULink.DLL is a standard Windows DLL, you cannot reference it.

Demo programs

In the subfolders there are axample programs written in Delphi, Visual Basic 6 and Visual Basic 2008 Express.

Note: In Delphi **longint** is a 32-bit integer.

Description of the procedures and functions of the FGULink.DLL

OpenGen

Syntax

```
PROCEDURE OpenGen;
```

Description

Turns the generator on.
This also enables access to other functions of the DLL.

Example

```
begin
 OpenGen;
end;
```

CloseGen

Syntax

```
PROCEDURE CloseGen;
```

Description

Turns the generator off.
Use this as the last function in your application before closing the application.

Example

```
begin
  CloseGen;
end;
```

StartGen

Syntax

```
PROCEDURE StartGen;
```

Description

Turns the generator signal output on.

Example

```
BEGIN
  StartGen;
END;
```

StopGen

Syntax

```
PROCEDURE StopGen;
```

Description

Turns the generator signal output off.

Example

```
BEGIN
  StopGen;
END;
```

SetGen

Syntax

```
procedure SetGen(func: Longint; freq, ampl, offset: Single);
```
Parameters

`func:` Selects the function:
1 - sine
2 - square
3 - triangle

`freq:` Frequency in Hz.
`ampl:` Peak-to-peak amplitude in volts.
`offset:` DC offset in volts.

Description
Set the function, frequency, amplitude and offset.

Example
```
procedure TForm1.Button4Click(Sender: TObject);
begin
 SetGen(1, 150.5, 8.5, 0.1); // Output a 8.5 Vpp sine wave at 150.5 Hz with
a 0.1 volt offset
end;
```

SetLibWave

Syntax
```
procedure SetLibWave(freq, ampl, offset: Single; filter: Longint; FileName:
PChar);
```

Parameters
`freq:` Frequency in Hz.
`ampl:` Peak-to-peak amplitude in volts.
`offset:` DC offset in volts.
`filter:` 1 = low pass filter on, 0 = low pass filter off
`FileName:` Library waveform file name.
Note: The library waveform files must be in "lib" subfolder of the FGU.EXE program's folder.

Description
Set the frequency, amplitude and offset of the library waveform.

Example
```
procedure TForm1.Button11Click(Sender: TObject);
var
  s:string;
begin
  s:='ramp_dn.lib';
  SetLibWave(1000, 5, 1, 1, PChar(s)); // Output a 5 Vpp 'ramp_dn.lib' wave
at 1 kHz with a 1 volt offset, filter on
end;
```

SetSweep

Syntax
```
procedure SetSweep(freq1, freq2, ampl, offset, time: Single; form:
Longint);
```

Parameters
`freq1:` Sweep start frequency in Hz.
`freq2:` Sweep stop frequency in Hz.
`ampl:` Peak-to-peak amplitude in volts.

`offset:` DC offset in volts.
`time:` Sweep time in seconds.
`form:` Sweep waveform type: 1=sine, 2=square.

Description
Set the the generator to output sweeping waveform.

Example
```
procedure TForm1.Button9Click(Sender: TObject);
begin
  SetSweep(20, 450, 5, 1.5, 5, 2); // Sweep start 20 Hz, stop 450 Hz,
amplitude 5 Vpp, offset 1.5 V, sweep time 5s, square waveform
end;
```

LogSweep

Syntax
procedure LogSweep(log: Boolean);

Parameter
Log: Boolean true activates logarithmic sweep. False activates linear sweep.

Description
Set the sweep mode either linear or logarithmic.

Example
```
begin
  LogSweep(true);
end;
```

ShowGen

Syntax
procedure ShowGen(visible: Boolean);

Parameter
Visible: Boolean true displays the user interface. False hides the user interface.

Description
Displays or hides the PCGU1000 user interface on the screen.

Example
```
begin
  ShowGen(not CheckBox2.checked);
end;
```

Note: In the FGen1000.ini there is a parameter ShowGen=1. Change it to ShowGen=0 and the generator GUI will be always hidden.

AttOn

Syntax
procedure AttOn(att: Boolean);

Parameter
Visible: Boolean true sets the 40dB attenuator on. False sets the attenuator off.

Description
Set the 40dB attenuator on or off.

Example
```
begin
  AttOn(CheckBox1.checked);
end;
```

GenReady

Syntax
```
function GenReady: Boolean;
```

Result
`Boolean`: TRUE indicates that the generator waveform setup is done.
`Boolean`: FALSE indicates that the generator is loading new waveform setup and is not yet ready.

Description
Checks that generator is ready to output the selected waveform.

Example
```
procedure TForm1.Timer1Timer(Sender: TObject);
begin
  if GenReady then label2.caption:='Generator Ready!'
  else label2.caption:='Generator Not Ready'
end;
```

GenStatus

Syntax
```
function GenStatus: Longint;
```

Result
Longint:
0: "Loading" period of the generator is not yet complete.
1: Everything OK.
-1: USB connection failed (cable disconnected).
-2: The power adaptor is disconnected.

Description
When thet FGU.exe is run from within the demo program or from your application, the GenStatus() function returns 0 until the "Loading" period of the generator is complete. After this period of time the GenStatus() function returns value 1 if everything OK. It returns -1 if the USB connection fails (cable disconnected) and -2 if the power adaptor is disconnected.

Example
```
label3.caption:='Status: '+inttostr(GenStatus);
```

Note: GenReady() and GenStatus() functions return the generator status about 1.5 seconds after every instruction sent to the generator. The previous generator state is returned until that, if no instruction is sent. Please allow in your application for the generator that amount of time after the instruction to update the status before using GenReady() and GenStatus() functions.