

# Make your own software for the Oscilloscope / Function Generator PCSGU250 with the Dynamic Link Library PCSGU250D.DLL

The PCSGU250.DLL is a 32 bit Windows DLL. This document describes all the functions and procedures of the DLL that are available for your application programme. Calling the functions and procedures exported by the DLL, you may write custom Windows 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 procedures and functions that are exported by the PCSGU250.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 2008 Express. In the listings there are full declarations for the DLL function and procedures.

Note:

- All the examples in this manual are written for Delphi.
- In this manual "Long Integer" is 32-bits.

## Overview of the Procedures and Functions of the PCSGU250.DLL

### General

Start_PCSGU250;	<i>Starts the PCSGU250.exe program</i>
Stop_PCSGU250;	<i>Closes the PCSGU250.exe program</i>
Show_PCSGU250(Visible: Boolean);	<i>Show or hide the PCSGU250 user interface</i>

### Oscilloscope

RunOn(Run: Boolean)	<i>Set PCSGU250 Run mode on or off</i>
SingleOn(Single: Boolean)	<i>Set PCSGU250 Single mode on or off</i>
Voltage1(Volts: Longint)	<i>Set V/div scale of Ch1</i>
Voltage2(Volts: Longint)	<i>Set V/div scale of Ch2</i>
Time(TpDiv: Longint)	<i>Set Time/div scale</i>
TrgLevel(TrgLevel: Longint)	<i>Set trigger level</i>
TrgEdge(Positive_Negative: Longint)	<i>Set trigger edge</i>
TrgOn(trg_on: Boolean)	<i>Set trigger on or off</i>
TrgSource(CH1_CH2_Ext: Longint)	<i>Set trigger source</i>
YPosition1(y_pos: Longint);	<i>Set trace Y-position of Ch1</i>
YPosition2(y_pos: Longint);	<i>Set trace Y-position of Ch2</i>
Coupling1(AC_DC_GND: Longint)	<i>Set the input coupling of Ch1</i>
Coupling2(AC_DC_GND: Longint)	<i>Set the input coupling of Ch2</i>
GetSettings(Settings: Pointer)	<i>Get the oscilloscope settings</i>
ReadCh1(Buffer: Pointer)	<i>Reads the data of channel 1</i>
ReadCh2(Buffer: Pointer)	<i>Reads the data of channel 1</i>
DataReady	<i>Indicates if there is fresh data available</i>

### Function Generator

StartGen	<i>Turns the generator signal output on</i>
StartGen	<i>Turns the generator signal output off</i>
SetGen func: Longint; freq, ampl, offset: Single)	<i>Set the function, frequency, amplitude and offset.</i>
SetLibWave(freq, ampl, offset: Single; FileName: PChar)	

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

```
SetSweep(freq1, freq2, ampl, offset, time: Single)
```

*Set the generator sweep waveform  
parameters.*

## Procedures and Functions of the PCSGU250.DLL

---

### General Functions

#### Start\_PCSGU250

*Syntax*

```
function Start_PCSGU250: Boolean;
```

*Description*

Starts the PCSGU250.EXE program.

*Result*

Boolean: TRUE means that the PcLab2000LT program is run.

*Example*

```
procedure TForm1.Button1Click(Sender: TObject);  
begin  
  if Start_PCSGU250 then  
  begin  
    Run.enabled:=true;  
    Single.enabled:=true;  
  end;  
end;
```

---

#### Stop\_PCSGU250

*Syntax*

```
PROCEDURE Stop_PCSGU250;
```

*Description*

Closes the PCSGU250.EXE program.

*Example*

```
procedure TForm1.FormClose(Sender: TObject; var Action:  
TCloseAction);  
begin  
  Stop_PCSGU250;  
end;
```

---

#### Show\_PCSGU250

*Syntax*

```
procedure Show_PCSGU250(Visible: Boolean);
```

*Parameter*

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

### *Description*

Displays or hides the PCSGU250 user interface on the screen.

### *Example*

```
procedure TForm1.ShowPCSGU250Click(Sender: TObject);
begin
    Show_PCSGU250(true);
end;

procedure TForm1.HidePCSGU250Click(Sender: TObject);
begin
    Show_PCSGU250(false);
end;
```

---

## Oscilloscope Functions

### RunOn

#### *Syntax*

```
PROCEDURE RunOn(Run: Boolean);
```

#### *Parameter*

Run: Boolean true sets the scope to Run mode. False stops Run mode.

#### *Description*

Set Run mode on or off.

#### *Example*

```
procedure TForm1.RunClick(Sender: TObject);
begin
    RunOn(Run.down);
end;
```

---

### SingleOn

#### *Syntax*

```
PROCEDURE SingleOn(Single: Boolean);
```

#### *Parameter*

Single: Boolean true sets the scope to Single mode. False stops Single mode.

#### *Description*

Set Single mode on or off.

#### *Example*

```
procedure TForm1.SingleClick(Sender: TObject);
begin
    SingleOn(true);
end;
```

---

### Voltage1, Voltage2

#### *Syntax*

```
PROCEDURE Voltage1(Volts: Longint);
PROCEDURE Voltage2(Volts: Longint);
```

### *Parameter*

**Volts:** The index of the V/div range.

0 = 3 V/div  
1 = 1 V/div  
2 = 0.3 V/div  
3 = 0.1 V/div  
4 = 30 mV/div  
5 = 10 mV/div

### *Description*

Set the V/div setting of the PCSGU250 oscilloscope.

### *Example*

```
procedure TForm1.VoltageRangeClick(Sender: TObject);  
begin  
    Voltage1((sender as TSpeedButton).tag);  
end;
```

---

## Time

### *Syntax*

```
PROCEDURE Time(TpDiv:Longint);
```

### *Parameter*

**TpDiv:** The index of the time/div setting rate.

0 = 500 ms/div  
1 = 200 ms/div  
2 = 100 ms/div  
3 = 50 ms/div  
4 = 20 ms/div  
5 = 10 ms/div  
6 = 5 ms/div  
7 = 2 ms/div  
8 = 1 ms/div  
9 = 0.5 ms/div  
10 = 0.2 ms/div  
11 = 0.1 ms/div  
12 = 50 us/div  
13 = 20 us/div  
14 = 10 us/div  
15 = 5 us/div  
16 = 2 us/div

### *Description*

Set the Time/div setting of the PCSGU250 oscilloscope.

### *Example*

```
procedure TForm1.TimeRangeClick(Sender: TObject);  
begin  
    Time((sender as TSpeedButton).tag);  
end;
```

---

## Coupling1, Coupling2

### *Syntax*

```
PROCEDURE Coupling1(AC_DC_GND:Longint);
PROCEDURE Coupling2(AC_DC_GND:Longint);
```

#### *Parameter*

AC\_DC\_GND: The index of the coupling type.

0 = AC

1 = DC

2 = GND

#### *Description*

Set the input coupling setting of the PCSGU250 oscilloscope.

#### *Example*

```
procedure TForm1.SelectCouplingClick(Sender: TObject);
begin
    Coupling1((sender as TSpeedButton).tag);
end;
```

---

## TriggerLevel

#### *Syntax*

```
PROCEDURE TriggerLevel(TrgLevel:Longint);
```

#### *Parameter*

TrgLevel: The triggering level value between 0 and 255.

#### *Description*

Set the triggering level of the PCSGU250 oscilloscope.

#### *Example*

```
procedure TForm1.TriggerLevelChange(Sender: TObject);
begin
    TriggerLevel(TrgLevel.position);
end;
```

---

## YPosition1, YPosition2

#### *Syntax*

```
PROCEDURE YPosition1(Position.position);
PROCEDURE YPosition2(Position.position);
```

#### *Parameter*

Position: The Y position of the trace (ground reference), value between -128 and 127.

#### *Description*

Sets the Y position of the trace.

#### *Example*

```
procedure TForm1.SetPositionChange(Sender: TObject);
begin
    YPosition(SetPosition.position);
end;
```

---

## TrgOn

### *Syntax*

```
PROCEDURE TrgOn(trg_on: Boolean);
```

### *Parameter*

trg\_on: Boolean TRUE sets the triggering on and FALSE sets the triggering off.

### *Description*

Set the PCSGU250 trigger on or off.

### *Example*

```
procedure TForm1.TriggerOnClick(Sender: TObject);
begin
    TrgOn(true);
end;
```

```
procedure TForm1.TriggerOffClick(Sender: TObject);
begin
    TrgOn(false);
end;
```

---

## TrgEdge

### *Syntax*

```
PROCEDURE TrgEdge(Positive_Negative:Longint)
```

### *Parameter*

Positive\_Negative: Index of the trigger edge.

0 = Negative

1 = Positive

### *Description*

Set the PCSGU250 trigger edge.

### *Example*

```
procedure TForm1.TriggerEdgeClick(Sender: TObject);
begin
    TrgEdge((sender as TSpeedButton).tag);
end;
```

---

## TrgSource

### *Syntax*

```
PROCEDURE TrgSource(CH1_CH2:Longint)
```

### *Parameter*

CH1\_CH2: Index of the trigger source.

0 = Ch1

1 = Ch2

### *Description*

Set the PCSGU250 trigger source.

### *Example*

```
procedure TForm1.SpeedButton27Click(Sender: TObject);
begin
  TrgSource((sender as TSpeedButton).tag);
end;
```

---

## GetSettings

### *Syntax*

```
FUNCTION GetSettings(SettingsArray: Pointer): Boolean
```

### *Parameter*

**SettingsArray**: Pointer to an array of 11 long integers (32 bit). The elements of the array indicate the current control settings on the PCSGU250 user interface. The values corresponding the settings are same as previously described in this document.

```
SettingsArray[0]: Volts/Div setting of CH1
SettingsArray[1]: Volts/Div setting of CH2
SettingsArray[2]: Time/Div setting
SettingsArray[3]: Y-position of CH1
SettingsArray[4]: Y-position of CH2
SettingsArray[5]: Coupling of CH1
SettingsArray[6]: Coupling of CH2
SettingsArray[7]: Trigger On/Off
SettingsArray[8]: Trigger source
SettingsArray[9]: Trigger edge
SettingsArray[10]: Trigger level
```

### *Description*

Get the current settings of each of the user-modifiable controls on the PCSGU250 user interface.

### *Result*

Boolean: TRUE means that the PCSGU250 oscilloscope program is running, FALSE means that it is not running.

### *Example*

```
procedure TForm1.Button4Click(Sender: TObject);
var SettingsArray:array [0..10] of longint;
begin
  if GetSettings(@SettingsArray) then
  begin
    case SettingsArray[0] of // get the V/div setting of CH1
      0: SpeedButton1.Down:=true;
      1: SpeedButton2.Down:=true;
      2: SpeedButton3.Down:=true;
      3: SpeedButton4.Down:=true;
      4: SpeedButton5.Down:=true;
      5: SpeedButton6.Down:=true;
    end;
  end;
end;
```

---

## ReadCh1, ReadCh2

### *Syntax*

```
PROCEDURE ReadCh1(Buffer: Pointer);
```

```
PROCEDURE ReadCh2(Buffer: Pointer);
```

#### *Parameter*

Buffer: A pointer to the data array of 5000 long integers where the data will be read.

#### *Description*

Read all the data and the settings of channel 1 or channel 2 of the PCSGU250.

As a return the following data is put to the buffer:

[0] : Sample rate in Hz

[1] : Full scale voltage in mV

[2] : Ground level in A/D converter counts. The value may be beyond the 0...255 range if GND level is adjusted beyond the waveform display area.

[3...4098] : The acquired data in A/D converter counts (0...255).

The triggering point is at the data location 1018.

#### *Example*

```
procedure TForm1.DataReadClick(Sender: TObject);
var i: longint;
p:pointer;
begin
  p:= @data1[0];
  ReadCh1(p);
  p:= @data2[0];
  ReadCh2(p);
  mem1.clear;
  mem1.lines.add('Sample rate
[Hz]'+chr(9)+inttostr(data1[0])+chr(9)+inttostr(data2[0]));
  mem1.lines.add('Full scale
[mV]'+chr(9)+inttostr(data1[1])+chr(9)+inttostr(data2[1]));
  mem1.lines.add('GND level
[counts]'+chr(9)+inttostr(data1[2])+chr(9)+inttostr(data2[2]));
  mem1.lines.add('');
  begin
    for i:=0 to 8 do
      mem1.lines.add('Data
('+inttostr(i)+')'+chr(9)+chr(9)+inttostr(data1[i+3])+chr(9)+inttostr
(data2[i+3]));
  end;
end;
```

---

## DataReady

#### *Syntax*

```
FUNCTION DataReady : Boolean;
```

#### *Description*

Indicates if there is fresh data available. This function can be used to check that triggering has occurred.

#### *Result*

Boolean: TRUE means that there is new waveform data available from the oscilloscope.

#### *Example*

```
if DataReady then lab1.caption:='Yes' else lab1.caption:='No'
```

---

## Function Generator Functions



## StartGen

### *Syntax*

```
PROCEDURE StartGen;
```

### *Description*

Turns the generator signal output on.

### *Example*

```
StartGen;
```

---

## StopGen

### *Syntax*

```
PROCEDURE StopGen;
```

### *Description*

Turns the generator signal output off.

### *Example*

```
StopGen;
```

---

## 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*

```
SetGen(1, 150.5, 8.5, 0.1);  
// Output a 8.5 Vpp sine wave at 150.5Hz with a 0.1 volt offset
```

---

## SetLibWave

### *Syntax*

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

### *Parameters*

freq: Frequency in Hz.

ampl: Peak-to-peak amplitude in volts.

offset: DC offset in volts.

FileName: Library waveform file name.

Note: The library waveform files must be in "lib" subfolder of the PcLab2000LT.EXE programme's folder.

*Description*

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

*Example*

```
SetLibWave(freq, ampl, offs, PChar(Edit7.text));
```

---

## SetSweep

*Syntax*

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

*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.

*Description*

Set the generator sweep waveform parameters.

*Example*

```
SetSweep(start, stop, ampl, offs, tim);
```

---

## Examples

In the subfolders there are example projects written in Delphi and in Visual Basic 2008 Express.