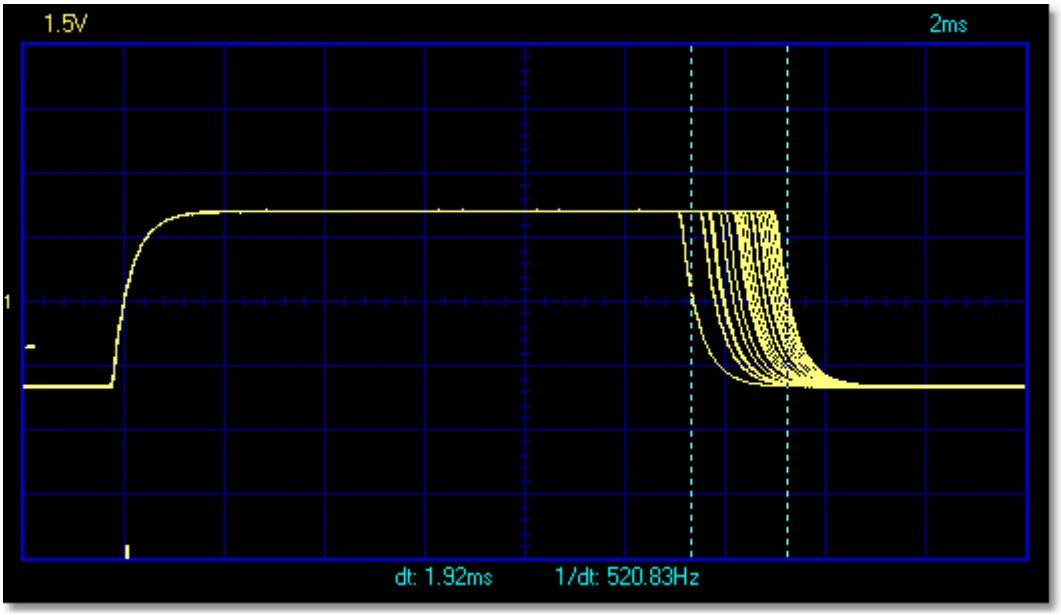
New PcLab2000 Features

PcLab2000 Version 1.38

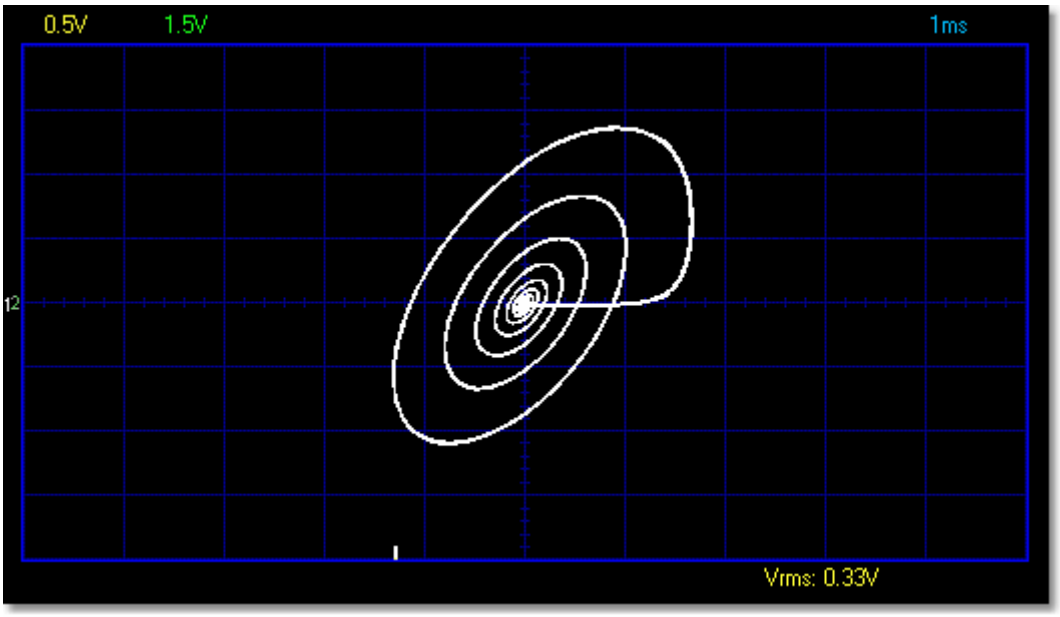
- **Persistence option added to PCS500**

Persistence display

When **Persist** button is depressed the scope captures many acquisitions of a signal to the screen. Record points accumulate until you release the button.
Using Persistence option you can easily analyze worst-case signal variations, such as jitter or noise. The Persist option can also be used to locate errors in digital signals. Using this option you can capture erroneous events even if they only occur once. Persist option makes it easy to compare known and unknown circuits. Click **Run** or **Single** button to capture multiple waveforms on the screen.



Persist option lets you see the range over which a signal varies



Use the Persist option to get solid XY patterns in the XY Plot mode on high sample rate

---

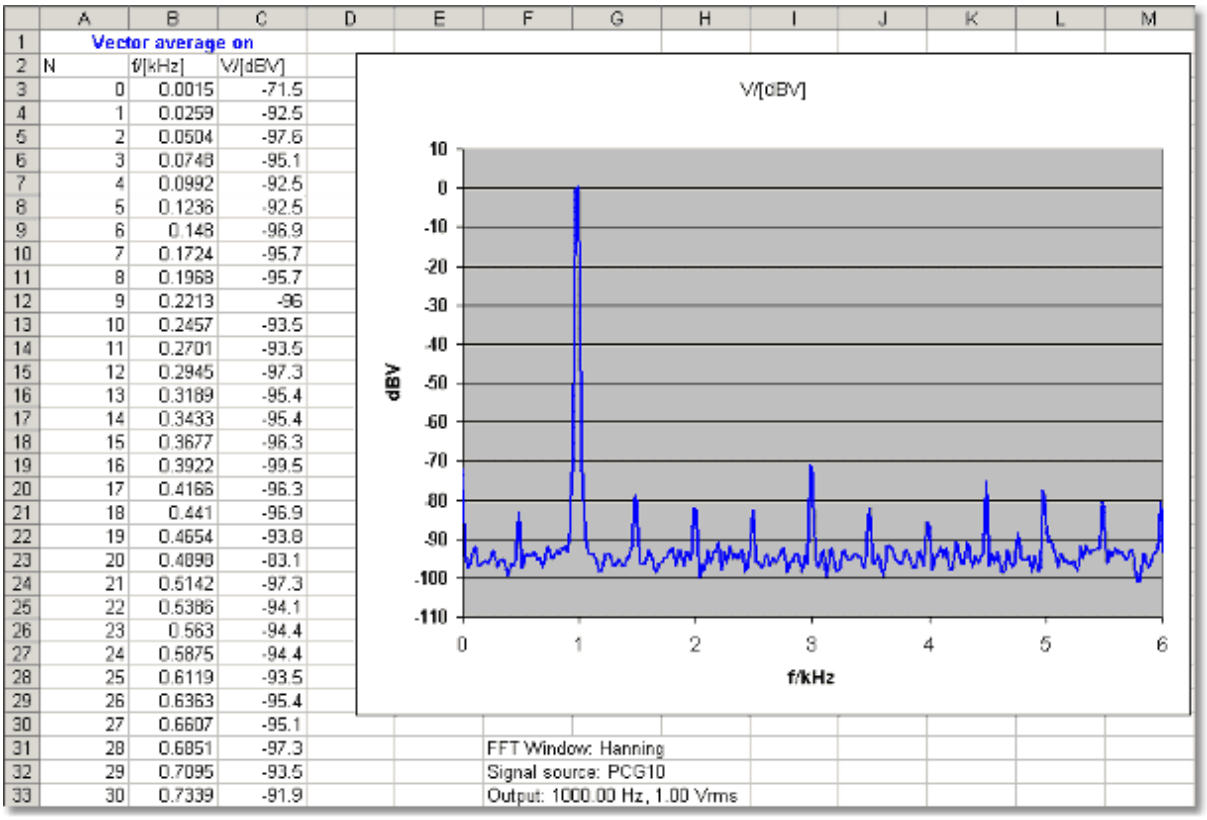
PcLab2000 Version 1.37

- **Added Vector Average mode to FFT spectrum analyzer to remove noise.**
- **The previous Average mode is replaced with RMS Average mode.**

Vector Average

Use this averaging mode to reduce random or uncorrelated noise in the synchronous signal you want to display.
Vector averaging requires a trigger - set **Trigger ON**.
The signal of interest must be both periodic and phase synchronous with the trigger.
Vector averaging reduces the noise floor for random signals since they are not phase coherent from time record to time record.
If not trigged, the signal will not add in phase and instead will cancel randomly.

If the noise level of the spectrum is beyond the 80dB display area you may use **File** menu option **Save FFT Data** to export the data to e.g. MS Excel.



	A	B	C
1	Vector average on		
2	N	f/[kHz]	V/[dBV]
3	0	0.0015	-71.5
4	1	0.0259	-92.5
5	2	0.0504	-97.6
6	3	0.0748	-95.1
7	4	0.0992	-92.5
8	5	0.1236	-92.5
9	6	0.148	-96.9
10	7	0.1724	-95.7
11	8	0.1968	-95.7
12	9	0.2213	-96
13	10	0.2457	-93.5
14	11	0.2701	-93.5
15	12	0.2945	-97.3
16	13	0.3189	-95.4
17	14	0.3433	-95.4
18	15	0.3677	-96.3
19	16	0.3922	-99.5
20	17	0.4166	-96.3
21	18	0.441	-96.9
22	19	0.4654	-93.8
23	20	0.4899	-93.1
24	21	0.5142	-97.3
25	22	0.5386	-94.1
26	23	0.563	-94.4
27	24	0.5875	-94.4
28	25	0.6119	-93.5
29	26	0.6363	-95.4
30	27	0.6607	-95.1
31	28	0.6851	-97.3
32	29	0.7095	-93.5
33	30	0.7339	-91.9

RMS Average
Use this averaging mode to reduce signal fluctuations.
RMS averaging provides an excellent estimate of the true signal and noise levels of the input signal.

Maximum
Maximum value of each frequency is displayed in Run mode.
This option can be used for recording signal levels as a function of frequency (Bode plot). You can use spreadsheet to display the frequency response curve including the frequency labels. On **File** menu, click **Save FFT Data** to export the data to the spreadsheet.

PcLab2000 Version 1.35

PCS500:

- **Added Spectral Density marker to the FFT spectrum analyzer to measure the noise density.**
- **Added Flat top window function to the FFT for accurate amplitude measurements.**

Spectral Density marker

The Spectral Density marker may be used when measuring the density of random or noise signals since it properly takes into account the frequency bin width and the FFT window function used by the spectrum analyzer when measuring noise-like signals.
The Spectral Density marker readout is automatically normalized to 1 Hz.
The displayed unit is: $dBV/\sqrt{Hz}$

Note: The Spectral Density marker should not be used to measure discrete frequency components as it will provide misleading level readings.

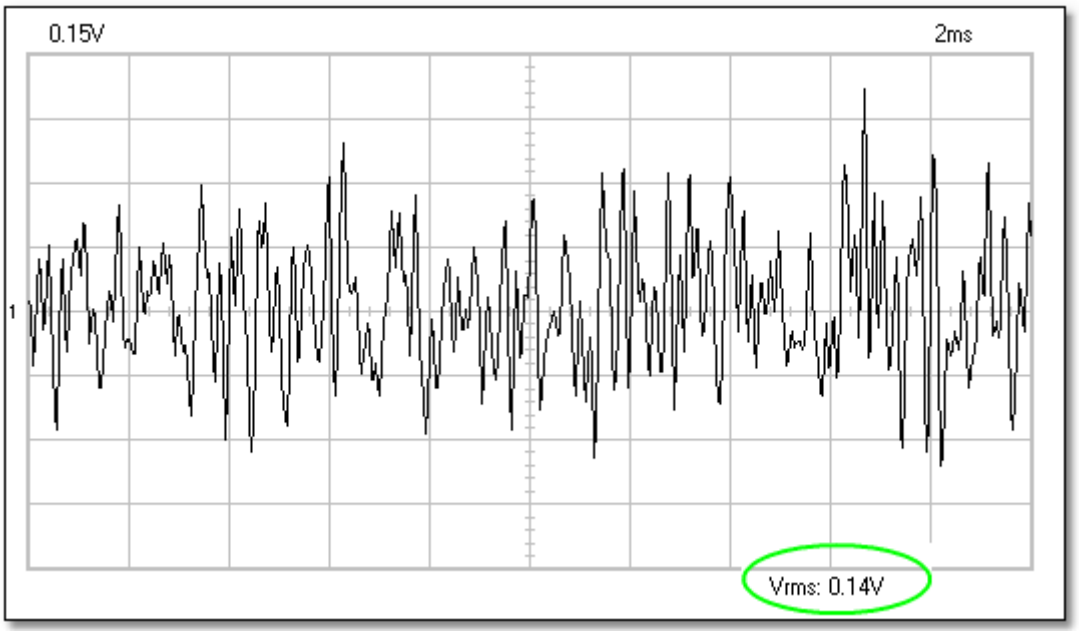
The Spectral Density is simply the magnitude of the spectrum normalized to a 1 Hz bandwidth. This

measurement approximates what the spectrum would look like if each frequency component were really a 1 Hz wide piece of the spectrum at each frequency bin.

When measuring broadband signals such as noise with a spectrum analyzer, the amplitude of the spectrum changes with the frequency span. This is because the FFT bin width changes and the frequency bins have a different noise bandwidth.

The Spectral Density marker normalizes all measurements to a 1 Hz bandwidth and the noise spectrum becomes independent of the frequency span. This allows measurements with different spans to be compared.
If the noise is Gaussian in nature, then the amount of noise amplitude in other bandwidths may be approximated by scaling the Spectral Density measurement by the square root of the noise bandwidth.

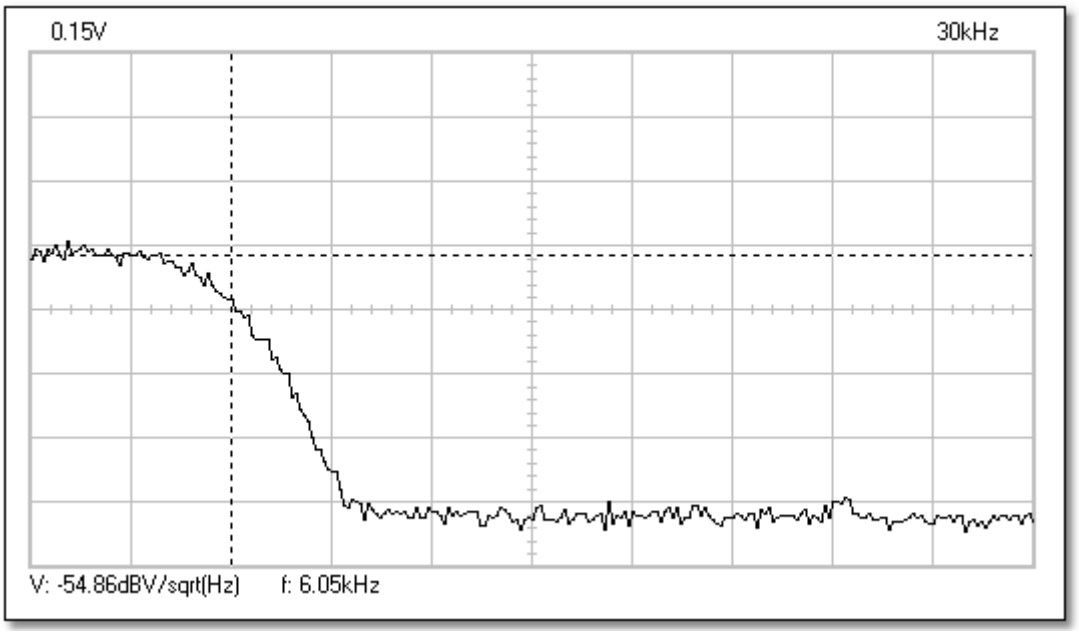
Example:



This image shows a band limited noise signal on an oscilloscope screen.

Spectrum analyzer can be used to measure the spectral density of this noise signal.
In the spectrum analyzer select the following menu options:

- **Options / FFT Options / RMS Average**
- **View / Markers (FFT) f & Spectral Density dBV/sqrt(Hz)**



This image shows the spectrum of the band limited noise signal.

The analysis of noise in the frequency domain shows the distribution of the noise amplitude as a function of frequency.

Using the **Spectral Density** marker and the Frequency marker, the voltage spectral density (V_{SdBV}) and the noise bandwidth (B_N) can be read from the spectrum analyzer display.

$$V_{SdBV} = -54.86\ dBV/\sqrt{Hz}$$
$$B_N = 6kHz$$

First convert the voltage spectral density to $V/\sqrt{Hz}$.
This can be accomplished using the following calculation:

$$V_S = 10^{-54.86/20} = 0.0018\ V/\sqrt{Hz}$$

This is the magnitude of the spectrum normalized to a 1 Hz bandwidth.
You may calculate the noise voltage over any bandwidth by multiplying this value by the square root of the bandwidth.

Assuming a 6 kHz bandwidth, the total output noise voltage is:

$$V_{rms} = V_S\sqrt{B_N}$$

$$V_{rms} = 0.0018\ V/\sqrt{Hz} * \sqrt{6000Hz} = \underline{0.139\ V}$$

(See the V_{rms} value of the oscilloscope waveform image of this noise signal.)

Spectrum analyzer supports now six different FFT windows

1. Rectangular
2. Bartlett
3. Hamming
4. Hanning
5. Blackman
6. Flat top

The Hamming window is set as default at the startup

Background information

It is common practice to taper the original signal before calculating the FFT (Fast Fourier Transformation). This reduces any discontinuities at the edges of the signal. This is done by multiplying the signal with a suitable window function. By choosing a tapered window it is possible to achieve a good compromise between main lobe width and sidelobe levels of a spectral line. The undesirable spectral leakage can be reduced using a tapered window, at the expense of some broadening around individual spectral lines. Many different windows have been designed for this purpose.

The choice of a suitable one depends on the nature of the signal or data, and on the type of information to be extracted from its spectrum. In general, a good FFT window has a narrow main spectral lobe to prevent local spreading of the spectrum, and low sidelobe levels to reduce 'distant' spectral leakage. In some cases it may be best to leave the data alone -- in effect, to use a rectangular window. For example, if a signal has close spaced components of roughly the same amplitude, the rectangular window will probably offer the best chance of resolving them. Conversely, if the amplitudes are very different, a window with low sidelobes will reduce leakage around the large component, and should make the small one easier to detect.

Comparison of the window functions

Window type	Window characteristics	Applications
Rectangular	Narrow mainlobe Slow rolloff rate	Broadband random (white noise) Closely spaced sine-wave signals
Bartlett	Narrow mainlobe Fast rolloff rate	
Hamming	Good spectral resolution Narrow mainlobe	Closely spaced sine-wave signals
Hanning	High maximum sidelobe level Good frequency resolution Reduced leakage Fast rolloff rate	Narrowband random signals Sine wave or combination of sine-wave signals
Blackman	Wide mainlobe Fast rolloff rate	
Flat top	Good amplitude accuracy Wide mainlobe Poor frequency resolution More spectral leakage	Sine wave with need for amplitude accuracy

PcLab2000 Version 1.34

PCS500:

- **Added Waveform Parameters display option to the View menu.**
- **Added voltage level display to horizontal markers.**

Waveform Parameters Display

When the menu option "Waveform Parameters" of the View menu is selected the software automatically calculates various voltage and time parameters of a signal, such as DC mean, amplitude, rise time etc.
These parameters are displayed in a separate window. Use the check boxes in the window to select the parameters you which to be displayed.

You may select the parameters to display for frozen waveform as well.
You may even re-open saved waveform data file to perform these measurements.
Please Note: Do not change the scope settings when the re-opened waveform parameters are to be read.

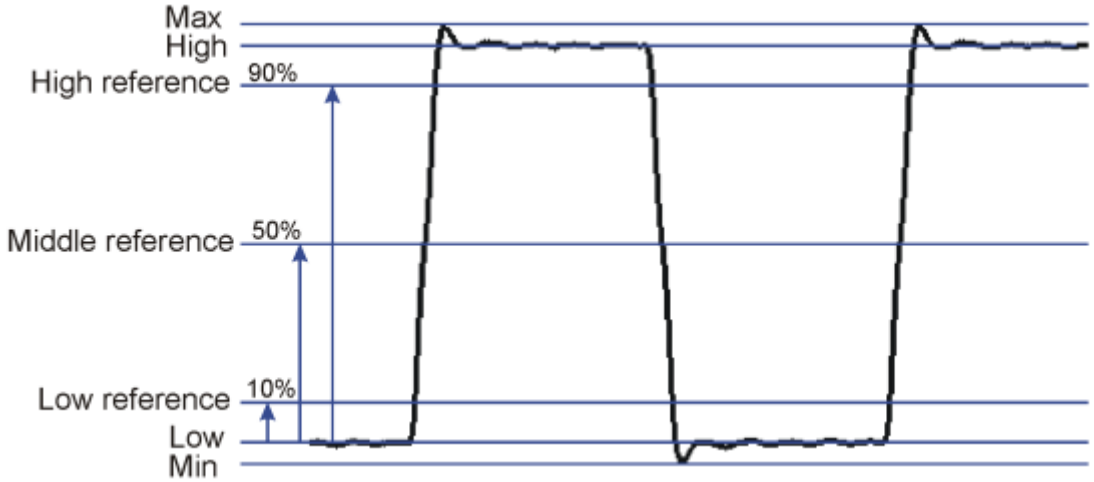
The green labeled parameters (High, Low, Amplitude, Rise time and Fall time) are mainly intended for pulse shaped waveform measurements only.

For proper waveform measurement the signal levels must be appropriate. Signal levels too small will result in noisy and inaccurate measurements. Signal levels too large will result in clipping and yield incorrect results.

Error indication:

?? Indicates that clipping has occurred.
??? Indicates that there are too few or too many wave cycles in the waveform data or the signal amplitude is too low. Also too noisy signal and variable frequency signal causes this indication.

Voltage parameters



DC Mean
The arithmetic mean of the entire waveform data.

Max
The signal's positive peak voltage.

(Difference between zero and highest value.)

Min
The signal's negative peak voltage.
(Difference between zero and lowest value.)

Peak-to-Peak
The signal's peak-to-peak voltage.
(Difference between highest and lowest value.)

High
The statistical maximum level recorded for all the cycles in the signal.

Low
The statistical minimum level recorded for all the cycles in the signal.

Amplitude
The voltage difference between the High and Low of the signal.

AC RMS
The true RMS value of the AC component of the signal is calculated and converted to voltage.

AC dBV
The measured signal (AC only) is converted to dBV (0dB= 1V).

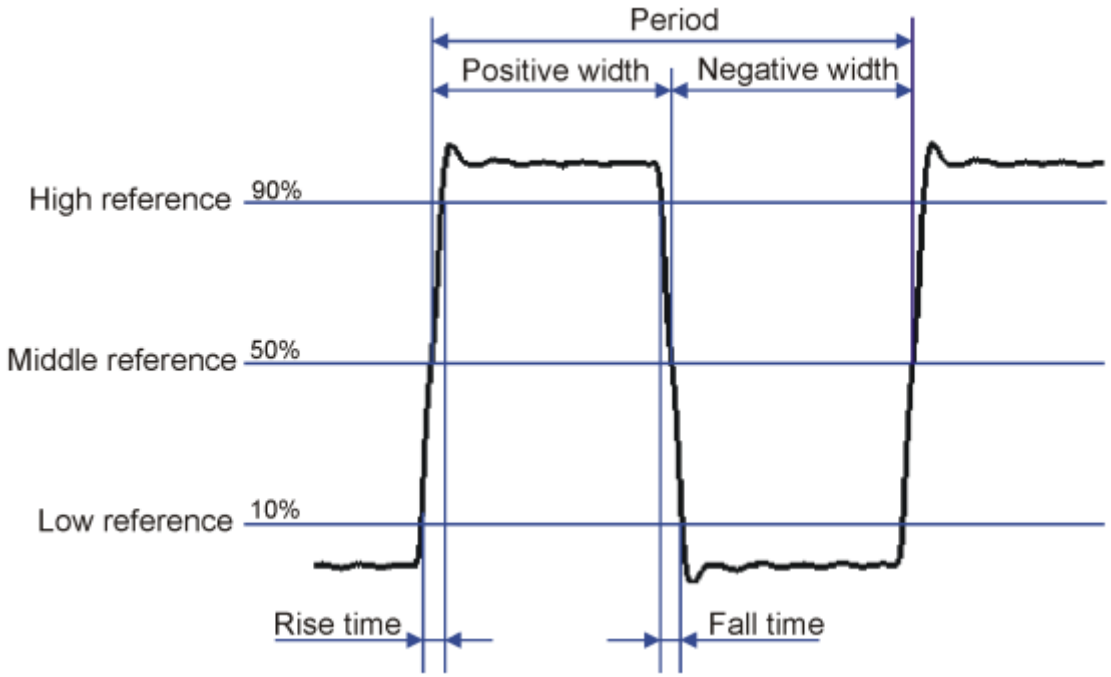
AC dBm
The measured signal (AC only) is converted to dBm (0dB= 0.775V).

AC+DC RMS
The true RMS value of the wave (AC+DC) is calculated and converted to voltage.

AC+DC dBV
The measured signal (AC+DC) is converted to dBV (0dB= 1V).

AC+DC dBm
The measured signal (AC+DC) is converted to dBm (0dB= 0.775V).

Time parameters



Duty Cycle
The ratio (expressed as a percentage) of the average positive pulse width to the average period of the signal. The time intervals are determined at middle reference level.
Duty Cycle = (Positive Pulse Width)/Period x 100%

Positive Width
Average positive pulse width in the waveform.
The time intervals are determined at middle reference level.
The middle reference is the middle point between the high and low levels.

Negative Width
Average negative pulse width in the waveform.
The time intervals are determined at middle reference level.

Rise Time
Time for a signal's rising edge to go from the low reference level to the high reference level.
The low reference level is 10% and high reference level is 90% of the pulse amplitude.

Fall Time
Time for a signal's falling edge to go from the high reference level to the low reference level.
The low reference level is 10% and high reference level is 90% of the pulse amplitude.

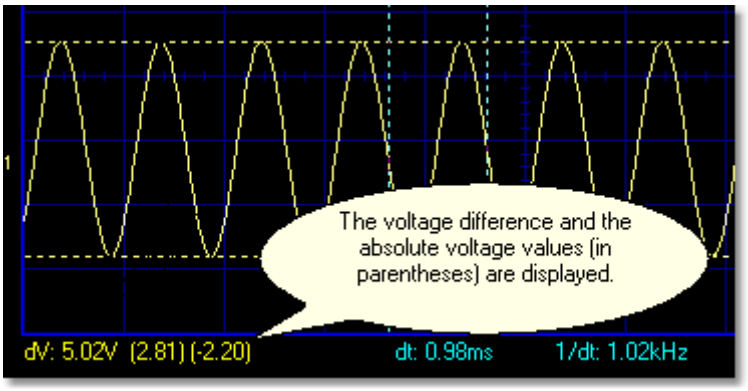
Period
The time interval between two consecutive crossings on the same slope of the signal at the middle reference level.

Frequency
The inverse of the Period of the signal.

Phase
Phase angle in degrees between CH1 and CH2.
For phase measurement the frequency of CH1 must be equal to the frequency of CH2.

Phase measurement is rather time consuming process. In slow PCs this reduces the display update rate.

Voltage level display of horizontal markers



PcLab2000 Version 1.33

- **Sample Rate display option added to the Transient Recorders.**

PcLab2000 Version 1.32

PCS500:

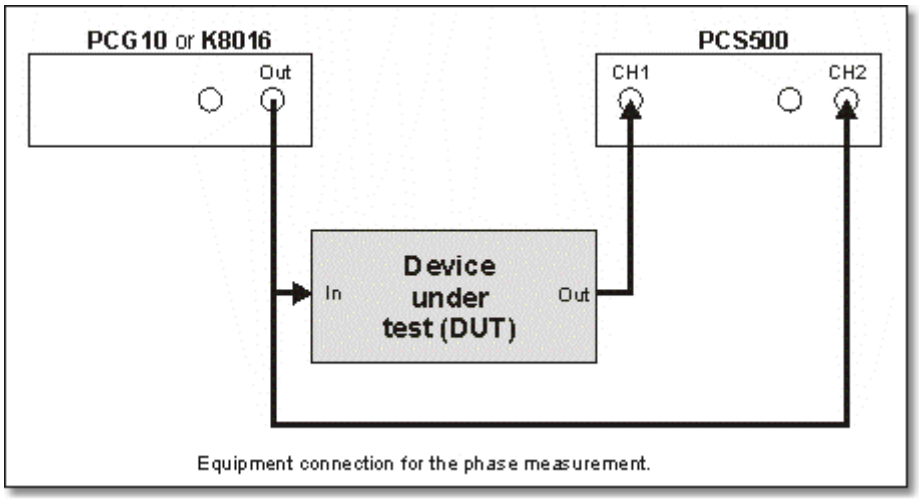
- **Phase Plot display option added to the Bode Plotter.**

Phase Plot

The phase relationship between the input and output of the DUT will be computed and displayed.

To get the Phase Plot:

- Connect the Function Generator output to both the input of the DUT and CH2 of the PCS500.
- Connect the output of the DUT to CH1 of the PCS500.
- From the **View** menu select **Phase Plot**.
- The signal level at CH2 must be appropriate. Signal levels too small will result in noisy measurements. Signal levels too large will result in clipping and yield incorrect results.

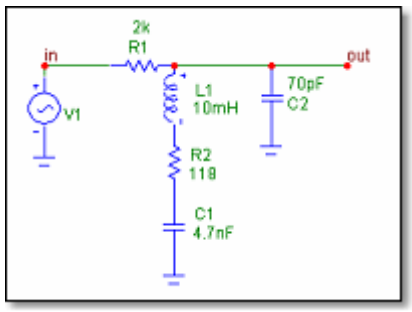


Equipment connection for the phase measurement.

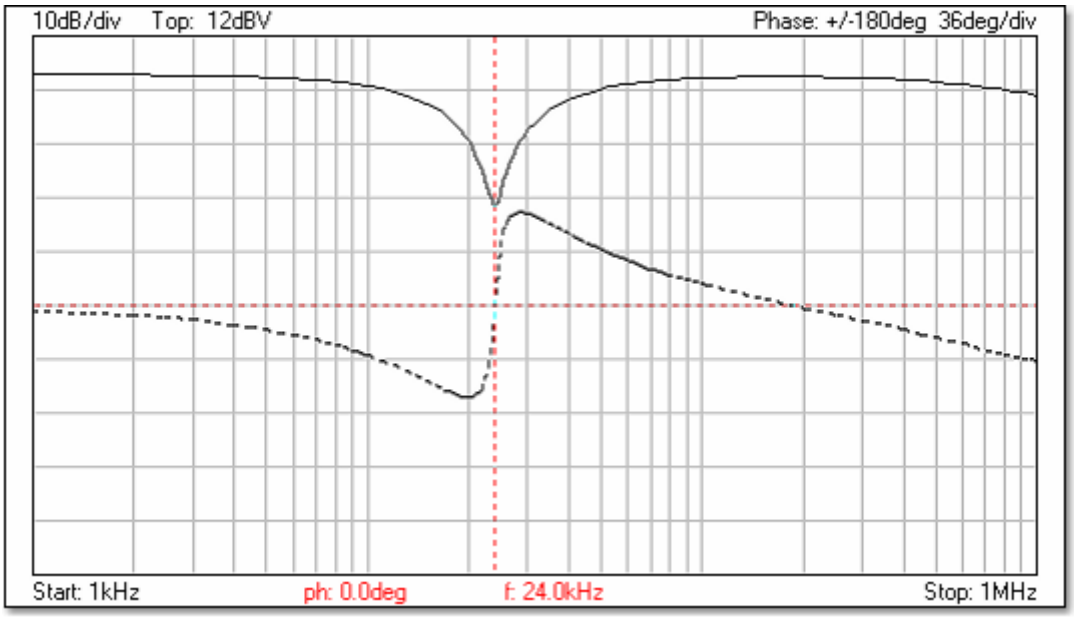
The frequency response graph will be plotted on the screen.
You can take a look to the oscilloscope display to see the amplifier output waveform.
If the graph is too low position or partly out of the display area change the **V Range** or oscilloscope's Volts/div setting, so that the graph is at the desired vertical position on the screen and click the **Start** button again.

You can afterwards change the display mode, frequency scale and the voltage scale (V Range) to survey the details of the graph.

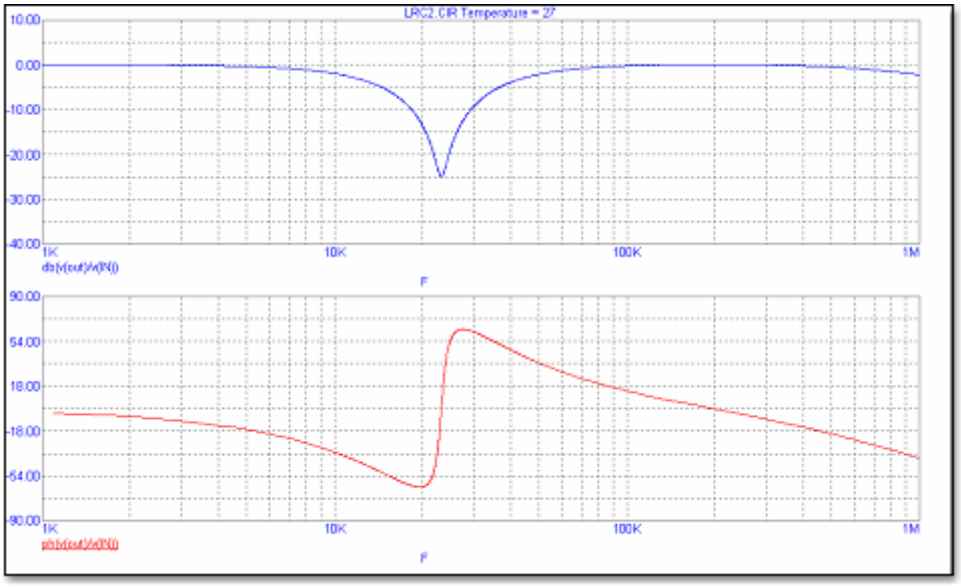
A Real-World Example of the PC-Lab2000 Bode Plotter output



Test circuit



PC-LAB2000 Bode Plot



Simulated Bode Plot (Spectrum Software, Micro-Cap 6)

PcLab2000 Version 1.31

- **Added Windows XP option to disable the Plug and Play detection of the LPT ports.**

Running Pc-Lab2000 software under Windows XP

The Plug and Play detection system of Windows XP periodically checks the presence of attached devices to the parallel port. This operation interferes with the operation of the PC oscilloscope and the function generator and may cause trace jumps on the oscilloscope screen and may turn off the

function generator.

There is following workaround for this problem:

Disable the Plug and Play detection of the LPT ports.

- Select the **Disable LPT Plug-and-Play** check box on the Pc-Lab2000 startup screen.
- You can any time restore the Plug and Play detection by clearing this check box before you click the **OK** button.
- If the setting has changed, you'll be prompted to restart the computer for the changes to take effect.
- This workaround can only be done using an account with administrator privileges.

PcLab2000 Version 1.30

- **Added direct data acquisition to other applications (Excel, Visual Basic, Delphi etc.)**
- **Added Save/Recall Settings option to the File menu.**
- **Added "Dot Join" display mode option to the View menu.**

Data acquisition to other applications

Pc-Lab2000 software package includes a DLL (Dynamic Link Library) *DSOLink.DLL,* installed to Windows' *SYSTEM32* folder.
This DLL allows you to write custom applications in Excel, Visual Basic, Delphi or any other 32-bit Windows application development tool that supports calls to a DLL.

The DLL gives you direct access to real-time data and settings information from the oscilloscope.

The complete example programs are located on the VELSOFT CD. Those may be used as a starting point how to construct your customized application programs.

Note: Before running the following example programs: The oscilloscope software must be running and "Run" or "Single" button pressed and trace displayed on the oscilloscope screen.

Description of the procedures of the DSOLink.DLL

`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 PCS500.
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), from PCS500.

The triggering point of the PCS500 is at the data location [1027].

Running the DSOLink in Delphi

Check the ***\PC-Lab2000 tools\PCS500 - PCS100 - K8031\Data transfer DSOLink_DLL\DSOLink_Demo_VB*** folder on the Velleman CD to locate the demo files.
This folder contains a ready to run *DSOLink_Demo.EXE* program and its source code.
You may copy the files to any folder and use Delphi to examine, edit and compile the files.

Example (in Delphi)

```
var
  data: array[0..5000] of longint;

procedure ReadCh1(Buffer: Pointer); stdcall; external 'DSOLink.dll';

procedure TForm1.Button1Click(Sender: TObject);
var i: longint;
p:pointer;
begin
  p:= @data[0];
  ReadCh1(p);
  memo1.clear;
  memo1.lines.add('Sample rate [Hz]'+chr(9)+inttostr(data[0]));
  memo1.lines.add('Full scale [mV]'+chr(9)+inttostr(data[1]));
  memo1.lines.add('GND level [counts]'+chr(9)+inttostr(data[2]));
  memo1.lines.add('');
  begin
    for i:=0 to 20 do
    memo1.lines.add( 'Data ('+inttostr(i)+')'+chr(9)+chr(9)+inttostr(data[i+3]));
  end;
end;
```

Running the DSOLink in Visual Basic

Make sure that the file *DSOLink.DLL* is copied to Windows' *SYSTEM32* folder.
Check the ***\PC-Lab2000 tools\PCS500 - PCS100 - K8031\Data transfer DSOLink_DLL\DSOLink_Demo_VB*** folder on the VELSOFT CD to locate the demo files.
This folder contains a ready to run *DSOLink_Demo.EXE* program and its source code.
You may copy the files to any folder and use Visual Basic to examine, edit and compile the files.

Example (in Visual Basic)

```
Option Explicit

Dim DataBuffer(0 To 5000) As Long
Private Declare Sub ReadCh1 Lib "DSOLink.dll " (Buffer As Long)

'This reads the settingsd and 4096 bytes of data from CH1 to the data buffer.
'The first 21 values are displayed.

Private Sub Read_CH1_Click(Index As Integer)
    Dim i As Long
    List1.Clear
    ReadCh1 DataBuffer(0)
    List1.AddItem "Sample rate [Hz]" + Chr(9) + Str(DataBuffer(0))
    List1.AddItem "Full scale [mV]" + Chr(9) + Str(DataBuffer(1))
    List1.AddItem "GND level [counts]" + Chr(9) + Str(DataBuffer(2))
    List1.AddItem ""
    For i = 0 To 20
      List1.AddItem "Data(" + Str(i) + ")" + Chr(9) + Chr(9) + Str(DataBuffer(i + 3))
    Next
End Sub
```

Running the DSOLink in Borland C++ Builder

The following files are available in the ***\PC-Lab2000 tools\PCS500 - PCS100 - K8031\Data transfer DSOLink_DLL\DSOLink_Demo_BCB*** folder on the VELSOFT CD for development with Borland C++Builder:

DSOLink.dll the dynamically linked library
DSOLink.h the C/C++ header file for function prototypes
DSOLink.lib the import library
DSOLink_demo.cpp demo source

1. Create a new project in Borland C++ Builder.
2. Add the import library to your project using **Project | Add to Project** menu option.
3. Add a `#include` statement in the main unit that includes *DSOLink.H*.
4. Finally, add code that calls the DLL functions.

DSOLink.h

```
//---------------------------------------------------------------------------
// DSOLink.h
#ifdef __cplusplus
extern "C" {          /* Assume C declarations for C++ */

#endif

#define FUNCTION __declspec(dllimport)

FUNCTION __stdcall ReadCh1(int* ptr);
FUNCTION __stdcall ReadCh2(int* ptr);

#ifdef __cplusplus
}
#endif
//---------------------------------------------------------------------------
```

Example (in Borland C++Builder)

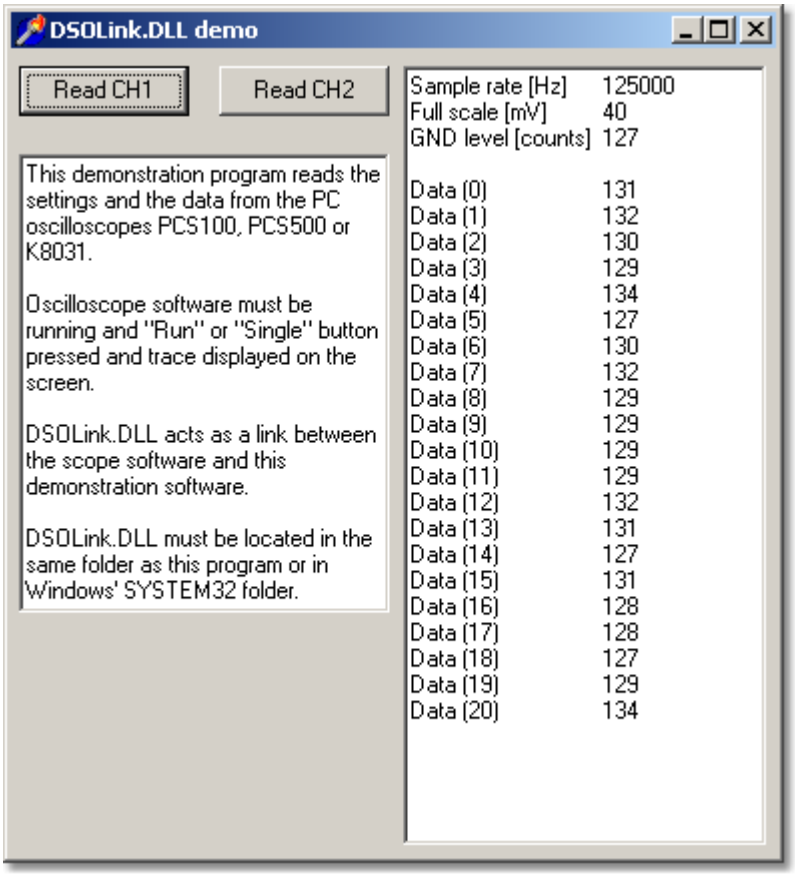
```
//---------------------------------------------------------------------------
// DSOLink_demo.cpp

#include <vcl.h>
#pragma hdrstop
#include "DSOLink.h"
#include "DSOLink_demo.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button1Click(TObject *Sender)
{
        int data[5000];
        ReadCh1(data);

        Memo1->Clear();
        Memo1->Lines->Add("Sample rate [Hz]:   "+IntToStr(data[0]));
        Memo1->Lines->Add("Full scale [mV]:    "+IntToStr(data[1]));
        Memo1->Lines->Add("GND level [counts]: "+IntToStr(data[2]));
        Memo1->Lines->Add("");

        for (int i = 0; i < 20; i++)
        {
                Memo1->Lines->Add("Data "+IntToStr(i)+char(9)+IntToStr(data[i+3]));
        }
}
//---------------------------------------------------------------------------
```

Note: If the import library is not compatible to your Borland C++ version, you can create an import library by running IMPLIB on the DLL.
IMPLIB works like this:

```
IMPLIB (destination lib name) (source dll)
```

For example,

```
IMPLIB DSOLink.lib DSOLink.dll
```



Data acquisition to Microsoft Excel

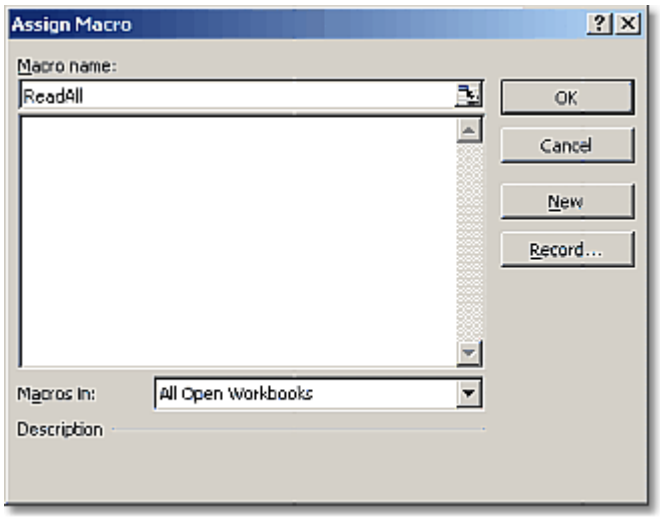
Pc-Lab2000 software package includes a DLL (Dynamic Link Library) *DSOLink.DLL*, installed to Windows' *SYSTEM32* folder.
This DLL allows you to write custom applications in Excel, Visual Basic, Delphi or any other 32-bit Windows application development tool that supports calls to a DLL.

Transferring Waveform Data to an Excel Spreadsheet

The example Excel macro shows you how to collect data directly into the spreadsheet from the Velleman PC oscilloscopes without the need for other software.

1. Open **Microsoft Excel** and start a new workbook document.
2. Select the **View / Toolbars** Menu and Select **Forms.**
   - The Forms toolbar appears.
3. Create a **Button**
   - In the Forms toolbar click on the "Button" button: the mouse-pointer will become a small cross.

- On Excel worksheet, use the mouse to draw a rectangle to mark where you want your button to appear.
- When you release the mouse after drawing the rectangle the "Assign Macro" dialog-box will appear.

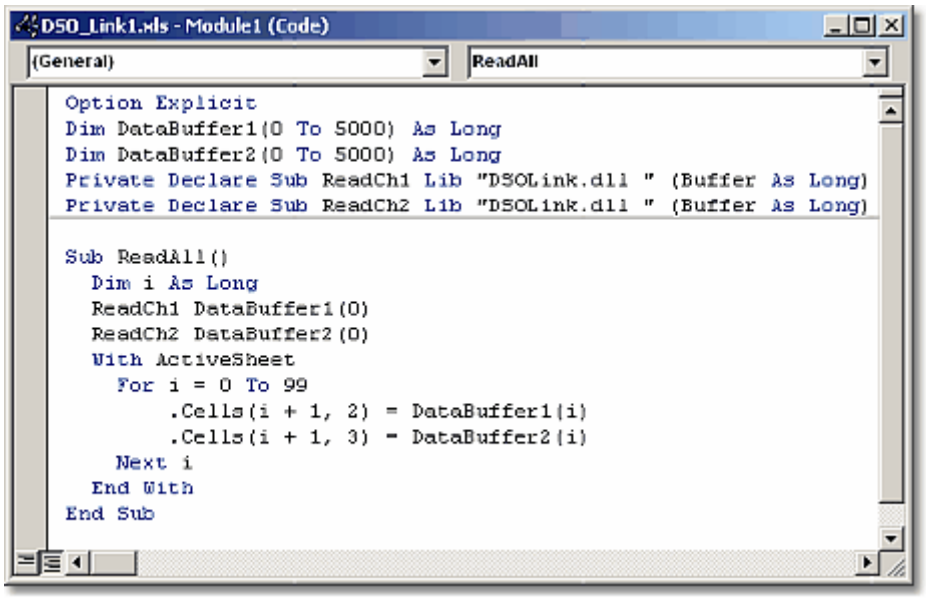


4. Type Macro name: ***ReadAll*** and click **New** button.
   - Microsoft Visual Basic edit window will open. A subroutine called *ReadAll* has been created.
5. Substitute the default text:

```
Sub ReadAll()
End Sub
```

with the following text in the edit window:
(Use Copy and Paste.)

```
Option Explicit
Dim DataBuffer1(0 To 5000) As Long
Dim DataBuffer2(0 To 5000) As Long
Private Declare Sub ReadCh1 Lib "DSOLink.dll " (Buffer As Long)
Private Declare Sub ReadCh2 Lib "DSOLink.dll " (Buffer As Long)

Sub ReadAll()
  Dim i As Long
  ReadCh1 DataBuffer1(0)
  ReadCh2 DataBuffer2(0)
  With ActiveSheet
    For i = 0 To 99
        .Cells(i + 1, 2) = DataBuffer1(i)
        .Cells(i + 1, 3) = DataBuffer2(i)
    Next i
  End With
End Sub
```



6. Press **Alt+F11** to return to Excel.
7. Type following texts to column **A**:

Sample rate [Hz]
Full scale [mV]
GND level [counts]
Data 0
Data 1
Data 2
...

8. Start oscilloscope program for PCS500, PCS100 or K8031 and click **Run** or **Single** button.
9. Click the button on the Excel worksheet. The created macro will execute and the data described in column **A** will appear to the worksheet columns **B** and **C**.
   - Rows 4...4099 contain the acquired data in A/D converter counts (0...255) for PCS500.
   - Rows 4...4083 contain the acquired data in A/D converter counts (0...255) for PCS100 and K8031.
   - The triggering point of PCS500 is on row 1030 and of PCS100 and K8031 on row 4.

   The three first rows contain the scope settings and the rest of the rows contain the raw oscilloscope data in A/D converter counts (0...255).
   Using the ***Sample rate***, ***Full scale*** and ***GND level*** values it is possible to reconstruct the waveform data into engineering units (e.g. Volts and seconds) for further analysis.

	A	B	C
1	**Sample rate [Hz]**	**625000**	**625000**
2	**Full scale [mV]**	**12000**	**4000**
3	**GND level [counts]**	**178**	**7**
4	Data 0	158	12
5	Data 1	140	11
6	Data 2	121	11
7	Data 3	116	11
8	Data 4	134	10
9	Data 5	152	10
10	Data 6	170	10
11	Data 7	188	102
12	Data 8	206	103
13	Data 9	224	104
14	Data 10	243	104
15	Data 11	232	104
16	Data 12	213	105
17	Data 13	195	105
18	Data 14	177	13
19	Data 15	159	13
20	Data 16	141	12
21	Data 17	123	11
22	Data 18	115	11