pico® Technology Limited

PicoScope 3000 Series PC Oscilloscopes

User Guide

PS3000044-1.5

Contents

1 Welcome

The PicoScope 3000 Series of PC Oscilloscopes from Pico Technology is a range of high-specification, real-time measuring instruments that connect to the USB port of your computer. The oscilloscopes obtain their power supply through the USB cable, so they do not need an additional power supply and are therefore highly portable. The 3000 Series consists of two ranges:

High-performance range (PicoScope 3204, 3205 and 3206 variants)

High-precision range (PicoScope 3224 and 3424 variants).

With the PicoScope software you can use PicoScope 3000 Series PC Oscilloscopes as oscilloscopes and spectrum analysers; and with the PicoLog software you can use them as data loggers. Alternatively, using the API functions, 17 you can develop your own programs to collect and analyse data from the oscilloscope.

A typical PicoScope 3000 Series PC Oscilloscope is supplied with the following items:

- USB cable, for use with USB 1.1 and 2.0 ports
- Software and Reference CD
- Installation Guide

2 Introduction

2.1 Safety symbols

Symbol 1: Warning Triangle

This symbol indicates that a safety hazard exists on the indicated connections if correct precautions are not taken. Read all safety documentation associated with the product before using it.

Symbol 2: Equipotential

This symbol indicates that the outer shells of the indicated BNC connectors are all at the same potential (shorted together). You must therefore take necessary precautions to avoid applying a potential across the return connections of the indicated BNC terminals as this may cause a large current to flow, resulting in damage to the product and/or connected equipment.

2.2 Safety warning

We strongly recommend that you read the general safety information below before using your oscilloscope for the first time. Safety protection built in to equipment may cease to function if the equipment is used incorrectly. This could cause damage to your computer, or lead to injury to yourself and others.

Maximum input range

PicoScope 3000 Series PC Oscilloscopes are designed to measure voltages in the range -20 V to +20 V. Inputs are protected to ±100 V (±30 V for external trigger). Contact with voltages outside the protection range may cause permanent damage to the unit.

Mains voltages

Pico Technology products are not designed for use with mains voltages. To measure mains, use a differential isolating probe specifically designed for a high source voltage.

Safety grounding

PicoScope 3000 Series PC Oscilloscopes connect direct to the ground of a computer through the interconnecting cable provided to minimise interference.

As with most oscilloscopes, avoid connecting the ground input to any potential other than ground. If in doubt, use a meter to check that there is no significant AC or DC voltage between the ground input of the oscilloscope and the point to which you intend to connect it. Failure to check may cause damage to your computer, or lead to injury to yourself and others.

You should not rely on the product to provide a protective safety earth.

Repairs

The oscilloscope contains no user-serviceable parts. Repair or calibration of the oscilloscope requires specialised test equipment and must be performed by Pico Technology.

2.3 FCC notice

This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to **Part 15 of the FCC Rules**. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference in which case the user will be required to correct the interference at his or her own expense.

For safety and maintenance information see the safety warning 4.

2.4 CE notice

The PicoScope 3000 Series PC Oscilloscopes meet the intent of the **EMC directive 89/336/EEC** and have been designed to **EN61326-1 (1997) Class A Emissions and Immunity** standard.

PicoScope 3000 Series PC Oscilloscopes also meet the intent of the **Low Voltage Directive** and have been designed to meet the **BS EN 61010-1:2001 IEC 61010-1:2001** (safety requirements for electrical equipment, control, and laboratory use) standard.

2.5 Licence conditions

The material contained in this release is licensed, not sold. Pico Technology Limited grants a **licence** to the person who installs this software, subject to the **conditions** listed below.

Access
The licensee agrees to allow access to this software only to persons who have been informed of these conditions and agree to abide by them.

Usage
The software in this release is for use only with Pico products or with data collected using Pico products.

Copyright
Pico Technology Limited claims the copyright of, and retains the rights to, all material (software, documents etc.) contained in this release. You may copy and distribute the entire release in its original state, but must not copy individual items within the release other than for backup purposes.

Liability
Pico Technology and its agents shall not be liable for any loss, damage or injury, howsoever caused, related to the use of Pico Technology equipment or software, unless excluded by statute.

Fitness for purpose
Because no two applications are the same, Pico Technology cannot guarantee that its equipment or software is suitable for a given application. It is your responsibility, therefore, to ensure that the product is suitable for your application.

Mission-critical applications
This software is intended for use on a computer that may be running other software products. For this reason, one of the conditions of the licence is that it excludes usage in mission-critical applications; for example, life-support systems.

2.6 Trademarks

Windows, Excel and **Visual Basic** are registered trademarks or trademarks of Microsoft Corporation in the USA and other countries. **Delphi** is a registered trademark of Borland Software Corporation. **Agilent VEE** is a registered trademark of Agilent Technologies, Inc. **LabView** is a registered trademark of National Instruments Corporation.

Pico Technology Limited, **PicoLog** and **PicoScope** are trademarks of Pico Technology Limited, registered in the United Kingdom and other countries.

2.7 Warranty

Pico Technology **warrants** upon delivery, and for a period of 24 months unless otherwise stated from the date of delivery, that the Goods will be free from defects in material and workmanship.

Pico Technology shall not be liable for a breach of the warranty if the defect has been caused by fair wear and tear, wilful damage, negligence, abnormal working conditions or failure to follow Pico Technology's spoken or written advice on the storage, installation, commissioning, use or maintenance of the Goods or (if no advice has been given) good trade practice; or if the Customer alters or repairs such Goods without the written consent of Pico Technology.

2.8 Company details

Address:

Pico Technology Limited
The Mill House
Cambridge Street
St Neots
Cambridgeshire
PE19 1QB
United Kingdom

Phone: +44 (0) 1480 396 395
Fax: +44 (0) 1480 396 296

Email:

Technical Support: support@picotech.com
Sales: sales@picotech.com

Web site: www.picotech.com

3 Product information

3.1 Minimum system requirements

To ensure that your PicoScope 3000 Series PC Oscilloscope operates correctly, you must have a computer with the minimum system requirements to run Windows or the following (whichever is the higher specification):

Processor	Pentium-class processor or equivalent, or better.
Memory	32 MB minimum.
Disk space	10 MB minimum.
Operating system	Microsoft Windows 98SE, ME, 2000 or XP.
Ports	USB 1.1 compliant port minimum. USB 2.0 compliant port recommended. Must be connected direct to the port or a powered USB hub. Will not work on a passive hub.

3.2 Installation instructions

Important
Do not connect your PicoScope 3000 Series PC Oscilloscope to the PC until you have installed the software.

Procedure

- Follow the instructions in the Installation Guide included with your product package.
- Connect your PC Oscilloscope to the PC using the USB cable supplied.
- There is no need for an additional power supply, as the unit obtains its power from the USB port.

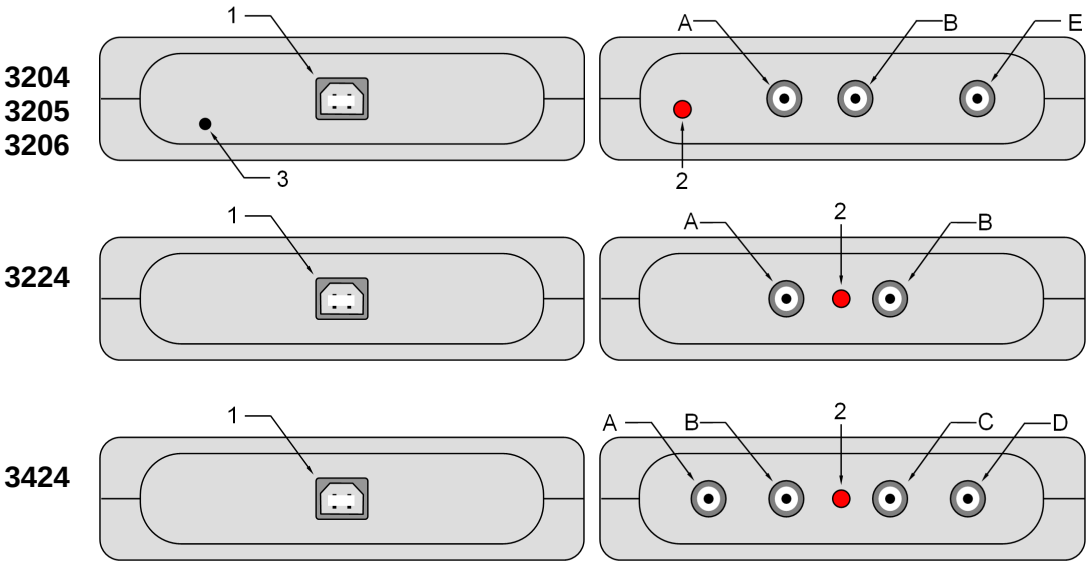
Checking the installation

Once you have installed the software and connected the PC Oscilloscope to the PC, start the PicoScope or PicoLog software. The software should now display any signal connected to the scope inputs. If you are using the PicoScope software and have a probe connected to your oscilloscope, you should see a small 50 or 60 hertz signal in the oscilloscope window when you touch the probe tip with your finger .

Standard oscilloscope connectors

PicoScope 3000 Series PC Oscilloscopes have standard oscilloscope connectors. The input impedance is also standard, so the x10 function on scope probes works correctly.

Connector diagrams



1 USB port
2 LED: lights when the oscilloscope is sampling data
3 Power socket: 12 V DC at 500 mA
A Input channel A
B Input channel B
C Input channel C
D Input channel D
E External trigger input / signal generator output *

* The BNC connector labelled '**E**' on the 3204/5/6 models has two functions. In normal use it is the external trigger input with an adjustable threshold. Alternatively, on some oscilloscopes, it can also be used to output sine, square and triangle waveforms which can be swept back and forth at a user-defined frequency. The integrated signal generator 15 can be controlled by the PicoScope software or by API calls. The signal generator can also be used to compensate x10 scope probes when set to output a square wave.

Ground loops

If you experience excessive noise or voltage offsets when using the PicoScope 3204/5/6 variants, you may have a ground loop problem. See Powering the 3204/5/6 43 for advice on overcoming this.

Moving your PicoScope PC Oscilloscope to another USB port

When you install the PicoScope 3000 Series PC Oscilloscope by plugging it into a USB port, Windows associates the Pico driver 10 with that port. If you later move the oscilloscope to a different USB port, Windows will display the "New Hardware Found Wizard" again. When this occurs, simply follow the steps listed in the Installation Guide after the instruction "Connect the PicoScope 3000 Series PC Oscilloscope to the PC..." As all the software you need is already installed on your computer, there is no need to insert the Pico Software and Reference CD again.

3.3 Specifications

Variant	3204	3205	3206	3224	3424
Vertical resolution	8 bits			12 bits	
Analog bandwidth	50 MHz	100 MHz	200 MHz	10 MHz	
Max. sampling rate One channel in use Two channels in use 3 or 4 channels in use Repetitive signals	 50 MS/s 50 MS/s - 2.5 GS/s	 100 MS/s 100 MS/s - 5 GS/s	 200 MS/s 100 MS/s - 10 GS/s	 20 MS/s 10 MS/s - -	 20 MS/s 10 MS/s 5 MS/s -
Trigger bandwidth	50 MHz	100 MHz	150 MHz	10 MHz	
Buffer size (samples per channel) One channel in use Two channels in use 3 or 4 channels in use	 256 K 128 K -	 512 K 256 K -	 1 M 512 K -	 512 K 256 K -	 512 K 256 K 128 K
Inputs	2 BNC channels				4 BNC channels
	1 MΩ impedance AC/DC coupling 20 pF capacitance				
Outputs Signal generator	Fixed (Note 1)	Variable (Note 2)		None	
External trigger	1 BNC input shared with signal generator Variable trigger threshold ± 20 V Rising/falling 12.2 mV resolution 1 MΩ impedance			None	
Voltage ranges	± 100 mV to ± 20 V in 8 ranges			± 20 mV to ± 20 V in 10 ranges	
Accuracy	3% voltage 100 ppm time			1% voltage 100 ppm time	
Operating environment Temperature range Humidity	 0°C to 70°C (25°C for quoted accuracy) 25% to 75% RH			 0°C to 70°C (20°C to 30°C for quoted accuracy) 25% to 75% RH	
Overload protection Channels External trigger	 ± 100 V ± 30 V			 ± 100 V -	
PC connection	USB 2.0 Compatible with USB 1.1				
Power supply	From USB port: 4.6 to 5.25 V 500 mA External power supply is not required			From USB port	
Dimensions	140 mm x 200 mm x 45 mm				
Compliance	CE standard [5]; FCC Part 15 [4]				

(1) 1 BNC shared with external trigger. Fixed frequency 1 kHz. 5 V square wave. 600 Ω output impedance.
(2) 1 BNC shared with external trigger. Variable frequency 100 Hz to 1 MHz. 5 V square wave, 1 V sine wave and triangle functions. Repeat sweep function. Dual slope function. 600 Ω output impedance.

4 Technical reference

4.1 Driver

The Windows 98SE/ME/2000/XP 32-bit **driver**, `picopp.sys`, is installed under the control of an information file, `picopp.inf`.

Once you have installed the PicoScope and PicoLog software, Windows will automatically install the driver when you plug in the PicoScope 3000 Series PC Oscilloscope for the first time.

4.2 Programming with the PicoScope 3000 Series

The `ps3000.dll` library in your PicoScope installation directory allows you to program a PicoScope 3000 Series oscilloscope using standard C function calls. 17

A typical program for capturing data consists of the following steps:

- Open 17 the scope unit.
- Set up the input channels with the required voltage ranges 10 and coupling mode 10.
- Set up triggering 11.
- Set up ETS 12, if required.
- Start capturing data. (See Sampling modes 11, where programming is discussed in more detail.)
- Wait until the scope unit is ready.
- Copy data to a buffer.
- Stop capturing data.
- Close the scope unit.

Numerous sample programs 39 are installed with your PicoScope software. These show how to use the functions of the driver software in each of the modes available.

4.2.1 Voltage ranges

It is possible to set the gain for each channel with the ps3000_set_channel() 20 function. This will give an input **voltage range** between ±20 mV (for the 3224 and 3424) or ±100 mV (for the 3204/5/6) and ±20V. The external trigger on the 3204/5/6 variants has a fixed input range of ±20 V.

4.2.2 AC/DC coupling

Using the ps3000_set_channel() 20 function, each channel can be set to either **AC** or **DC** coupling. When AC coupling is used, any DC component of the signal is filtered out.

4.2.3 Triggering

PicoScope 3000 Series PC Oscilloscopes can either start collecting data immediately, or be programmed to wait for a **trigger** event to occur. In both cases you need to use the ps3000_set_trigger() 25 function. A trigger event can occur when one of the signal or trigger input channels crosses a threshold voltage on either a rising or a falling edge.

Applicability	Available in block mode 11 and fast streaming mode 14 only. Calls to the ps3000_set_trigger() 25 function have no effect in compatible streaming mode 13. The external trigger input on the PicoScope 3204, 3205 and 3206 scope units uses the same BNC connector as the signal generator output, so these two functions cannot be used independently. It is possible, however, to use the output from the signal generator as a trigger.

4.2.4 Sampling modes

PicoScope 3000 Series PC Oscilloscopes can run in various **sampling modes**.

- **Block mode.** 11 At the highest sampling rates, the oscilloscope collects data much faster than a PC can read it. To compensate for this, the oscilloscope stores a block of data in an internal memory buffer, delaying transfer to the PC until the required number of data points have been sampled.

- **Streaming modes.** 13 At all but the highest sampling rates, these modes allow accurately timed data to be transferred back to the PC without gaps. The computer instructs the oscilloscope to start collecting data. The oscilloscope then transfers data back to the PC without storing it in its own memory, so the size of the data set is limited only by the size of the PC's memory. Sampling intervals from less than one microsecond to 60 seconds are possible. There are two streaming modes:
  - **Compatible streaming mode.** 13 Works with all scope variants.
  - **Fast streaming mode.** 14 Works with 3224 and 3424 variants.

4.2.4.1 Block mode

In **block mode**, the computer prompts a PicoScope 3000 series PC Oscilloscope to collect a block of data into its internal memory. When the oscilloscope has collected the whole block, it will signal that it is ready and then transfer the whole block to the computer's memory through the USB port.

The maximum number of values depends upon the size of the oscilloscope's memory. A PicoScope 3000 Series PC Oscilloscope can sample at a number of different rates. These rates correspond to the maximum sampling rate divided by 1, 2, 4, 8 and so on.

There is a separate memory buffer for each channel. When a channel is unused, its memory can be borrowed by the enabled channels. On the faster models, one input can be routed to two circuits in the oscilloscope, thereby doubling the effective sampling rate of a single channel. These features are handled transparently by the driver.

The driver normally performs a number of setup operations before collecting each block of data. This can take up to 50 milliseconds. If it is necessary to collect data with the minimum time interval between blocks, avoid calling setup functions between calls to ps3000_run_block() 27, ps3000_ready() 29, ps3000_stop() 29 and ps3000_get_values() 30.

See Using block mode 12 for programming details.

4.2.4.2 Using block mode

This is the general procedure for reading and displaying data in **block mode:** 11

1. Open the oscilloscope using ps3000_open_unit(). 17
2. Select channel ranges and AC/DC coupling using ps3000_set_channel(). 20
3. Using ps3000_set_trigger() 25, set the trigger if required.
4. Using ps3000_get_timebase() 21, select timebases until the required ns per sample is located.
5. If required, set the signal generator frequency using ps3000_set_siggen(). 22
6. Start the oscilloscope running using ps3000_run_block(). 27
7. Wait until the oscilloscope says it is ready using ps3000_ready(). 29
8. Transfer the block of data from the oscilloscope using ps3000_get_values() 30 or ps3000_get_times_and_values(). 31
9. Display the data.
10. Repeat steps 6 to 9.
11. Stop the oscilloscope using ps3000_stop() 29.

4.2.4.3 ETS (Equivalent Time Sampling)

ETS is a way of increasing the effective sampling rate when working with repetitive signals. It is controlled by the ps3000_set_trigger() 25 and ps3000_set_ets() 24 functions.

ETS works by capturing many instances of a repetitive waveform, then combining them to produce a composite waveform that has a higher effective sampling rate than the individual instances. The scope uses special circuitry to add a tiny variable delay, a small fraction of a single sampling interval, to each trigger event. This shifts each capture slightly in time so that the samples occur at slightly different times relative to those in the previous capture. The result is a much larger set of samples spaced by a small fraction of the original sampling interval. The maximum effective sampling rates that can be achieved with this method are listed in the Specifications 9 table.

Because of the high sensitivity of ETS mode to small time differences, you must set up the trigger to provide a stable waveform that varies as little as possible from one capture to the next.

Applicability	Available in block mode 11 only. Available on the PicoScope 3204, 3205 and 3206 variants. As ETS will return random time intervals, the ps3000_get_times_and_values() 31 function must be used. The ps3000_get_values() 30 function will return FALSE (0). Not suitable for one-shot (non-repetitive) signals.

4.2.4.4 Using ETS mode

This is the general procedure for reading and displaying data in **ETS** mode:

1. Open the oscilloscope using ps3000_open_unit(). 17
2. Select channel ranges and AC/DC switches using ps3000_set_channel(). 20
3. Using ps3000_set_trigger() 25, set the trigger if required.
4. Set ETS mode using ps3000_set_ets(). 24
5. Start the oscilloscope running using ps3000_run_block(). 27
6. Wait until the oscilloscope says it is ready using ps3000_ready(). 29
7. Transfer the block of data from the oscilloscope using ps3000_get_times_and_values(). 31
8. Display the data.
9. Repeat steps 5 to 8 as necessary.
10. Stop the oscilloscope using ps3000_stop(). 29

4.2.4.5 Streaming modes

The **streaming modes** are alternatives to block mode 11 that can capture data without gaps between blocks. There are two streaming modes.

- Compatible streaming mode 13
- Fast streaming mode 14

4.2.4.6 Compatible streaming mode

Compatible streaming mode is a basic streaming mode 13 that works with all scope units, at speeds from one sample per minute to a few thousand samples per second.

The oscilloscope's driver transfers data to a computer program using either normal or windowed mode. In normal mode, any data collected since the last data transfer operation is returned in its entirety. Normal mode is useful if the computer program requires fresh data on every transfer. In windowed mode, a fixed number of samples is returned, where the oldest samples may have already been returned before. Windowed mode is useful when the program requires a constant time period of data.

Once the oscilloscope is collecting data in streaming mode, any setup changes (for example, changing a channel range or AC/DC setting in the PicoScope software application) will cause a restart of the data stream. The driver can buffer up to 32K samples of data per channel, but the user must ensure that the ps3000_get_values() 30 function is called frequently enough to avoid buffer overrun.

See Using compatible streaming mode 14 for programming details.

Applicability	Works with all PicoScope 3000 Series scope units. Does not support triggering 11. The ps3000_get_times_and_values() 31 function will always return FALSE (0) in streaming mode.

4.2.4.7 Using compatible streaming mode

This is the general procedure for reading and displaying data in **compatible streaming mode** 13:

1. Open the oscilloscope using ps3000_open_unit(). 17
2. Select channel ranges and AC/DC switches using ps3000_set_channel(). 20
3. Start the oscilloscope running using ps3000_run_streaming(). 28
4. Transfer the block of data from the oscilloscope using ps3000_get_values(). 30
5. Display the data.
6. Repeat steps 4 and 5 as necessary.
7. Stop the oscilloscope using ps3000_stop(). 29

4.2.4.8 Fast streaming mode

Fast streaming mode is an advanced streaming mode 13 that can transfer data at speeds of a million samples per second or more, depending on the computer's performance. This makes it suitable for **high-speed data acquisition**, allowing you to capture very long data sets limited only by the computer's memory.

Fast streaming mode also provides data aggregation, which allows your application to zoom in and out of the data with the minimum of effort.

Applicability	Works with triggering. 11 Works with the high-resolution PicoScope 3000 Series (3224 and 3424) units.

See Using fast streaming mode 14 for programming details.

4.2.4.9 Using fast streaming mode

This is the general procedure for reading and displaying data in fast streaming mode: 14

1. Open the oscilloscope using ps3000_open_unit(). 17
2. Select channel ranges and AC/DC switches using ps3000_set_channel(). 20
3. Set the trigger using ps3000_set_trigger(). 25
4. Start the oscilloscope running using ps3000_run_streaming_ns(). 32
5. Get a block of data from the oscilloscope using ps3000_get_streaming_last_values(). 33
6. Display or process the data.
7. If required, check for overview buffer overruns by calling ps3000_overview_buffer_status(). 38
8. Repeat steps 5 to 7 as necessary or until `auto_stop` is `TRUE`.
9. Stop fast streaming using ps3000_stop(). 29
10. Retrieve any part of the data at any time scale by calling ps3000_get_streaming_values(). 35
11. If you require raw data, retrieve it by calling ps3000_get_streaming_values_no_aggregation(). 37
12. Repeat steps 10 to 11 as necessary.
13. Close the oscilloscope by calling ps3000_close_unit(). 38

4.2.5 Oversampling

When the oscilloscope is operating at sampling rates less than the maximum, it is possible to **oversample**. Oversampling is taking more than one measurement during a time interval and returning an average. If the signal contains a small amount of noise, this technique can increase the effective vertical resolution of the oscilloscope by the amount given by the equation below:

Increase in resolution (bits) = log (oversample) / log (4)

Applicability	Available in block mode 11 only.

4.2.6 Scaling

The PicoScope 3000 Series PC Oscilloscopes have resolutions of 8 bits or 12 bits, but the oscilloscope driver normalises all readings to 16 bits. This enables it to take advantage of noise reduction from oversampling 15, when this is enabled. The following table shows the relationship between the reading from the driver and the voltage of the signal.

Constant	Reading	Voltage
PS3000_LOST_DATA	-32 768	Indicates a buffer overrun in fast streaming 14 mode.
PS3000_MIN_VALUE	-32 767	Negative full scale
0	0	Zero volts
PS3000_MAX_VALUE	32 767	Positive full scale

4.2.7 Signal generator

The PicoScope 3204/5/6 PC Oscilloscopes have a built-in **signal generator** which is set using ps3000_set_siggen() 22. The output of the 3204 is a fixed-frequency square wave, while the 3205 and 3206 can produce a selection of accurate frequencies from 100 Hz to 1 MHz, and the waveform can be set to sine, square or triangle and swept back and forth in frequency. These options are selected under software control.

Applicability	Works with PicoScope 3204, 3205 and 3206 oscilloscopes. The signal generator output and external trigger input share the same connector, so these two functions cannot be used independently. It is possible, however, to use the output from the signal generator as a trigger.

4.2.8 Combining oscilloscopes

It is possible to collect data using up to four PicoScope 3000 Series PC Oscilloscopes at the same time. Each oscilloscope must be connected to a separate USB port. If a USB hub is used it must be a powered hub. The ps3000_open_unit() 17 function returns a handle to an oscilloscope. All the other functions require this handle for oscilloscope identification. For example, to collect data from two oscilloscopes at the same time:

```
handle1 = ps3000_open_unit()
handle2 = ps3000_open_unit()

ps3000_set_channel(handle1)
... set up unit 1
ps3000_run_block(handle1)

ps3000_set_channel(handle2)
... set up unit 2
ps3000_run_block(handle2)

ready = FALSE
while not ready
    ready = ps3000_ready(handle1)
    ready &= ps3000_ready(handle2)

ps3000_get_values(handle1)
ps3000_get_values(handle2)
```

Note 1: It is not possible to synchronise the collection of data between oscilloscopes that are being used in combination.

Note 2: PicoLog includes the ability to combine data from up to four oscilloscopes.

4.2.9 Functions

The PicoScope 3000 Series API exports the following functions for you to use in your own applications.

4.2.9.1 ps3000_open_unit

```
short ps3000_open_unit ( void )
```

This function opens a PicoScope 3000 Series PC Oscilloscope. The driver can support up to four oscilloscopes.

Applicability	All modes.
Arguments	None.
Returns	`-1` if the oscilloscope fails to open, `0` if no oscilloscope is found, `>0` (device handle) if the device opened.

4.2.9.2 ps3000_open_unit_async

```
short ps3000_open_unit_async ( void )
```

This function opens a PicoScope 3000 Series PC Oscilloscope without waiting for the operation to finish. You can find out when it has finished by periodically calling ps3000_open_unit_progress() 18 until that function returns a non-zero value.

The driver can support up to four oscilloscopes.

Applicability	All modes.
Arguments	None.
Returns	`0` if there is a previous open operation in progress. `1` if the call has successfully initiated an open operation.

4.2.9.3 ps3000_open_unit_progress

```
short ps3000_open_unit_progress (
      short   *handle,
      short   *progress_percent )
```

This function checks on the progress of ps3000_open_unit_async() 17.

Applicability	All modes. Use only with ps3000_open_unit_async() 17.
Arguments	`handle,` a pointer to a location in which the function will store the handle of the opened device. `0` if no unit is found or the unit fails to open, handle of device (valid only if function returns `TRUE`) `progress_percent,` a pointer to an estimate of the progress towards opening the unit, from 0 to 100. 100 implies that the operation is complete.
Returns	`1` if the driver successfully opens the unit, `0` if opening still in progress `-1` if the unit failed to open or was not found

4.2.9.4 ps3000_get_unit_info

```
short ps3000_get_unit_info (
      short    handle,
      char     *string,
      short    string_length,
      short    line )
```

This function writes oscilloscope information to a character string. If the oscilloscope fails to open, only `line` types 0 and 6 are available to explain why the last open unit call failed.

Applicability	All modes.
Arguments	`handle`, the handle to the device from which info is required. If an invalid handle is passed, the error code from the last unit that failed to open is returned. `string`, a pointer to the character string buffer in the calling function where the unit information string (selected with `line`) will be stored. If a null pointer is passed, no information will be written. `string_length`, the length of the character string buffer. If the string is not long enough to accept all of the information, only the first `string_length` characters are returned. `line`, an enumerated type specifying what information is required from the driver.
Returns	The length of the string written to the character string buffer, `string`, by the function. If one of the parameters is out of range, or a null pointer is passed for string, zero will be returned.

`line`		**String returned**	**Example**
0	`PS3000_DRIVER_VERSION`	The version number of the DLL used by the oscilloscope driver.	"1, 0, 0, 2"
1	`PS3000_USB_VERSION`	The type of USB connection that is being used to connect the oscilloscope to the computer.	"1.1" or "2.0"
2	`PS3000_HARDWARE_VERSION`	The hardware version of the attached oscilloscope.	"1"
3	`PS3000_VARIANT_INFO`	The variant of PicoScope 3000 Series PC Oscilloscope that is attached to the computer.	"3206"
4	`PS3000_BATCH_AND_SERIAL`	The batch and serial number of the oscilloscope.	"CMY66/052"
5	`PS3000_CAL_DATE`	The calibration date of the oscilloscope.	"21Oct03"
6	`PS3000_ERROR_CODE`	One of the Error codes 42.	"4"

4.2.9.5 ps3000_set_channel

```
short ps3000_set_channel (
      short    handle,
      short    channel,
      short    enabled,
      short    dc,
      short    range )
```

Specifies if a channel is to be enabled, the AC/DC coupling mode and the input range.

Applicability	All modes.
Arguments	`handle`, the handle to the required device. `channel`, an enumerated type. Use `PS3000_CHANNEL_A (0)`, `PS3000_CHANNEL_B (1)`, `PS3000_CHANNEL_C (2)` or `PS3000_CHANNEL_D (3)`. Channels C and D are not available on all models. `enabled`, specifies if the channel is active: `TRUE`=active, `FALSE`=inactive. `dc`, specifies the AC/DC coupling mode: `TRUE`=DC, `FALSE`=AC. `range`, a code between 1 and 10. See the table below.
Returns	`0` if unsuccessful, or if one or more of the arguments are out of range. `1` if successful.

Code	Enumeration	Range	
1	PS3000_20MV	±20 mV	*Not available on all variants.*
2	PS3000_50MV	±50 mV	*Not available on all variants.*
3	PS3000_100MV	±100 mV	
4	PS3000_200MV	±200 mV	
5	PS3000_500MV	±500 mV	
6	PS3000_1V	±1 V	
7	PS3000_2V	±2 V	
8	PS3000_5V	±5 V	
9	PS3000_10V	±10 V	
10	PS3000_20V	±20 V	

4.2.9.6 ps3000_get_timebase

```
short ps3000_get_timebase (
      short    handle,
      short    timebase,
      long     no_of_samples,
      long     *time_interval,
      short    *time_units,
      short    oversample,
      long     *max_samples )
```

This function discovers which timebases are available on the oscilloscope. You should set up the channels using ps3000_set_channel() 20 and, if required, ETS mode using ps3000_set_ets() 24 first.

Applicability	All modes.
Arguments	`handle,` the handle of the required device. `timebase,` a code between 0 and the maximum timebase (dependent on variant). Timebase 0 is the fastest timebase, timebase 1 is twice the time per sample of timebase 0, timebase 2 is four times, etc. `no_of_samples,` the number of samples required. This value is used to calculate the most suitable time unit to use. `time_interval,` a pointer to the time interval, in `time_units,` between readings at the selected timebase. If a null pointer is passed, nothing will be written here. `time_units,` a pointer to the time units that `time_interval` is measured in. This value should also be passed when calling ps3000_get_times_and_values() 31. If a null pointer is passed, nothing will be written here. `oversample,` the amount of oversample required. An oversample of 4 would quadruple the time interval and quarter the maximum samples. At the same time it would increase the effective resolution by one bit. See Oversampling 15 for more details. `max_samples,` a pointer to the maximum samples available. The maximum samples may vary depending on the number of channels enabled, the timebase chosen and the oversample selected. If this pointer is null, nothing will be written here.
Returns	`1` if all parameters are in range, otherwise `0`.

4.2.9.7 ps3000_flash_led

```
short ps3000_flash_led ( short handle )
```

Flashes the LED on the front of the oscilloscope three times and returns within one second.

Applicability	All modes.
Arguments	`handle`, the handle of the PicoScope 3000 Series PC Oscilloscope.
Returns	`1` if a valid handle is passed, `0` if not.

4.2.9.8 ps3000_set_siggen

```
long ps3000_set_siggen (
      short    handle,
      short    wave_type,
      long     start_frequency,
      long     stop_frequency,
      float    increment,
      short    dwell_time,
      short    repeat,
      short    dual_slope )
```

This function is used to enable or disable the signal generator 15 and sweep functions.

Applicability	Sweep functions are not available if the oscilloscope is in streaming mode 14. The signal generator is available only on the PicoScope 3204/5/6 PC Oscilloscope variants. See remarks below and specifications 9 for more information.

Arguments	`handle`, the handle of the required device. `wave_type`, the type of wave. Choose `PS3000_SQUARE (0)`, `PS3000_TRIANGLE (1)` or `PS3000_SINE (2)`. *This argument has no effect if used with the PicoScope 3204 variant.* `start_frequency`, the required frequency, in the range 0 < freq < 1 MHz, to start the sweep or the frequency generated in a non-sweep mode. `0` switches the signal generator off. `stop_frequency`, the required stop frequency of the sweep, in the range 0 < freq < 1 MHz but not necessarily greater than `start_frequency`. If the start and stop frequencies are the same, the signal generator will be run with a constant frequency. *This argument has no effect if used with the PicoScope 3204 variant., or if run in streaming mode.* 13 `increment`, the size of the steps to increment or decrement the frequency by in a sweep mode. This must always be positive; the start and stop frequencies will determine whether to increment or decrement. This must be a frequency in the range 0.1 Hz < `increment` < `|stop_frequency - start_frequency|`. This is not used in a non-sweep mode. *This argument has no effect if used with the PicoScope 3204 variant..* `dwell_time`, the time, in milliseconds, to wait before increasing the frequency by `increment` in a sweep mode. This is unused in a non-sweep mode. *This argument has no effect if used with the PicoScope 3204 variant..* `repeat`, if `TRUE` restarts the sweep when the `stop_frequency` is reached, if `FALSE` continues indefinitely at `stop_frequency` when it is reached. *This argument has no effect if used with the PicoScope 3204 variant..* `dual_slope`, if `repeat` is `TRUE` this specifies what to do at the `stop_frequency`. `TRUE` will sweep back towards the `start_frequency`, `FALSE` will restart the sweep from `start_frequency`. *This argument has no effect if used with the PicoScope 3204 variant..*
Returns	The actual frequency or start frequency, in hertz, that is generated. `0` if one of the parameters is not in range.

Remarks
The PicoScope 3204 variant has a simple 1 kHz square wave signal generator for scope probe calibration. With this variant, therefore, only two arguments of this function have any effect:

To switch the square wave on, use a valid `handle` and set `start_frequency` to a non-zero value. To switch the square wave off, use a valid `handle` and set `start_frequency` to `0`.

4.2.9.9 ps3000_set_ets

```
long ps3000_set_ets (
      short    handle,
      short    mode,
      short    ets_cycles,
      short    ets_interleave )
```

This function is used to enable or disable ETS 12 (equivalent time sampling) and to set the ETS parameters.

Applicability	ETS 12 applies only to the PicoScope 3204, 3205 and 3206 variants.
Arguments	`handle`, the handle to the required device. `mode`, `PS3000_ETS_OFF (0)` - disables ETS. `PS3000_ETS_FAST (1)` - enables ETS and provides `ets_cycles` cycles of data, which may contain data from previously returned cycles, `PS3000_ETS_SLOW (2)` - enables ETS and provides fresh data every `ets_cycles` cycles. `PS3000_ETS_SLOW` takes longer to provide each data set, but the data sets are more stable and unique. `ets_cycles`, specifies the number of cycles to store: the computer can then select `ets_interleave` cycles to give the most uniform spread of samples. `ets_cycles` should be between two and five times the value of `ets_interleave`. `ets_interleave`, specifies the number of ETS interleaves to use. If the sample time is 20 ns and the interleave 10, the approximate time per sample will be 2 ns.
Returns	If ETS is enabled, the effective sample time in nanoseconds. `0` if ETS is disabled or one of the parameters is out of range.

4.2.9.10 ps3000_set_trigger

```
short ps3000_set_trigger (
      short    handle,
      short    source,
      short    threshold,
      short    direction,
      short    delay,
      short    auto_trigger_ms )
```

This function is used to enable or disable triggering and its parameters.

Applicability	Triggering is available in block mode 11 and fast streaming mode 14.
Arguments	`handle`, the handle to the required device. `source`, specifies where to look for a trigger. Use `PS3000_CHANNEL_A (0)`, `PS3000_CHANNEL_B (1)`, `PS3000_CHANNEL_C (2)`, `PS3000_CHANNEL_D (3)`, `PS3000_EXTERNAL(4)` or `PS3000_NONE(5)`. The number of channels available will depend on the scope variant. `threshold`, the threshold for the trigger event. This is scaled in 16-bit ADC counts at the currently selected range. If an external trigger is enabled the range is fixed at +/-20V. `direction`, use `PS3000_RISING (0)` or `PS3000_FALLING (1)`. `delay`, specifies the delay, as a percentage of the requested number of data points, between the trigger event and the start of the block. It should be in the range -100% to +100%. Thus, 0% means that the trigger event is at the first data value in the block, and -50% means that it is in the middle of the block. If you wish to specify the delay as a floating-point value, use ps3000_set_trigger2() 26 instead. `auto_trigger_ms`, the delay in milliseconds after which the oscilloscope will collect samples if no trigger event occurs. If this is set to zero the oscilloscope will wait for a trigger indefinitely.
Returns	`0` if one of the parameters is out of range, otherwise `1`.

4.2.9.11 ps3000_set_trigger2

```
short ps3000_set_trigger2 (
      short    handle,
      short    source,
      short    threshold,
      short    direction,
      float    delay,
      short    auto_trigger_ms )
```

This function is used to enable or disable triggering and its parameters. It has the same behaviour as ps3000_set_trigger() 25, except that the `delay` parameter is a floating-point value.

Applicability	Triggering is available in block mode 11 and fast streaming mode 14 only.
Arguments	`handle`, the handle of the required device. `source`, specifies where to look for a trigger. Use `PS3000_CHANNEL_A (0)`, `PS3000_CHANNEL_B (1)`, `PS3000_CHANNEL_C (2)`, `PS3000_CHANNEL_D (3)`, `PS3000_EXTERNAL(4)` or `PS3000_NONE(5)`. Channels C, D and External are not available on all models. `threshold`, the threshold for the trigger event. This is scaled in 16-bit ADC counts at the currently selected range. If an external trigger is enabled the range is fixed at +/-20V. `direction`, use `PS3000_RISING (0)` or `PS3000_FALLING (1)`. `delay`, specifies the delay, as a percentage of the requested number of data points, between the trigger event and the start of the block. It should be in the range -100% to +100%. Thus, 0% means that the trigger event is at the first data value in the block, and -50% means that it is in the middle of the block. If you wish to specify the delay as an integer, use ps3000_set_trigger() 25 instead. `auto_trigger_ms`, the delay in milliseconds after which the oscilloscope will collect samples if no trigger event occurs. If this is set to zero the oscilloscope will wait for a trigger indefinitely.
Returns	`0` if one of the parameters is out of range, otherwise `1`.

4.2.9.12 ps3000_run_block

```
short ps3000_run_block (
      short    handle,
      long     no_of_samples,
      short    timebase,
      short    oversample,
      long     *time_indisposed_ms )
```

This function tells the oscilloscope to start collecting data in block mode 12.

Applicability	Block mode 11 only.
Arguments	`handle`, the handle to the required device. `no_of_samples`, the number of samples to return. `timebase`, a code between 0 and the maximum timebase available (consult the driver header file). Timebase 0 gives the maximum sample rate available, timebase 1 selects a sample rate half as fast, timebase 2 is half as fast again and so on. For the maximum sample rate, see the specifications 9. Note that the number of channels enabled may affect the availability of the fastest timebases. `oversample`, the oversampling factor, a number between 1 and 256. See Oversampling 15 for details. `time_indisposed_ms`, a pointer to the approximate time, in milliseconds, over which the ADC will collect data. If a trigger is set, it is the amount of time the ADC takes to collect a block of data after a trigger event, calculated as sample interval x number of points required. Note: The actual time may differ from computer to computer, depending on how fast the computer can respond to I/O requests.
Returns	`0` if one of the parameters is out of range, otherwise `1`.

4.2.9.13 ps3000_run_streaming

```
short ps3000_run_streaming (
      short    handle,
      short    sample_interval_ms,
      long     max_samples,
      short    windowed )
```

This function tells the oscilloscope to start collecting data in compatible streaming mode 14. If this function is called when a trigger has been enabled, the trigger settings will be ignored.

For faster streaming with the PicoScope 3224 and 3424 variants, use ps3000_run_streaming_ns() 32 instead.

Applicability	Compatible streaming 13 mode only.
Arguments	`handle,` the handle to the required device. `sample_interval_ms,` the time interval, in milliseconds, between data points. This can be no shorter than 1 ms. `max_samples,` the maximum number of samples that the driver is to store. This can be no greater than 60 000. It is the caller's responsibility to retrieve data before the oldest values are overwritten. `windowed,` if this is 0, only the values taken since the last call to ps3000_get_values() 30 are returned. If this is 1, the number of values requested by ps3000_get_values() 30 are returned, even if they have already been read by ps3000_get_values() 30.
Returns	`1` if streaming has been enabled correctly, `0` if a problem occurred or a value was out of range.

4.2.9.14 ps3000_ready

```
short ps3000_ready ( short handle )
```

This function checks to see if the oscilloscope has finished the last data collection operation.

Applicability	Block mode 11 only. Does nothing if the oscilloscope is in streaming mode 13.
Arguments	`handle`, the handle to the required device.
Returns	`1` if ready. The oscilloscope has collected a complete block of data or the auto trigger timeout has been reached. `0` if not ready. An invalid handle is passed, or the oscilloscope is in streaming mode, or the scope is still collecting data in block mode. `-1` if device not attached. The endpoint transfer fails, indicating that the unit may well have been unplugged.

4.2.9.15 ps3000_stop

```
short ps3000_stop ( short handle )
```

Call this function to stop the oscilloscope sampling data. If this function is called before a trigger event occurs, the oscilloscope may not contain valid data.

Applicability	All modes.
Arguments	`handle`, the handle to the required device.
Returns	`0` if an invalid handle is passed, `1` otherwise.

4.2.9.16 ps3000_get_values

```
long ps3000_get_values (
    short handle
    short *buffer_a,
    short *buffer_b,
    short *buffer_c,
    short *buffer_d,
    short *overflow,
    long  no_of_values )
```

This function is used to get values in compatible streaming mode 14 after calling ps3000_run_streaming() 28, or in block mode 11 after calling ps3000_run_block() 27.

Applicability	Compatible streaming mode 13 and block mode 11 only. Does nothing if ETS triggering is enabled. Do not use in fast streaming mode 14 - use ps3000_get_streaming_last_values() 33 instead.
Arguments	`handle`, the handle to the required device. `buffer_a, buffer_b, buffer_c, buffer_d,` pointers to the buffers that receive data from the specified channels (A, B, C or D). A pointer is unused if the oscilloscope is not collecting data from that channel. If a pointer is `NULL`, nothing will be written to it. `overflow`, a bit pattern indicating whether an overflow has occurred on a channel. Bit 0 is the least significant bit. Bit 0 --> channel A Bit 1 --> channel B Bit 2 --> channel C Bit 3 --> channel D `no_of_values,` the number of data points to return. In streaming mode, this is the maximum number of values to return.
Returns	The actual number of data values per channel returned, which may be less than `no_of_values` if streaming. `FALSE` if one of the parameters is out of range.

4.2.9.17 ps3000_get_times_and_values

```
long ps3000_get_times_and_values (
      short    handle
      long     *times,
      short    *buffer_a,
      short    *buffer_b,
      short    *buffer_c,
      short    *buffer_d,
      short    *overflow,
      short    time_units,
      long     no_of_values )
```

This function is used to get values and times in block mode 11 after calling ps3000_run_block() 27.

Applicability	Block mode 11 only. It will not return any valid times if the oscilloscope is in streaming mode 14. Essential for ETS operation.
Arguments	`handle`, the handle to the required device. `times`, a pointer to the buffer for the times in `time_units`. Each time is the interval between the trigger event and the corresponding sample. Times before the trigger event are negative, and times after the trigger event are positive. `buffer_a, buffer_b, buffer_c, buffer_d`, pointers to the buffers that receive data from the specified channels (A, B, C or D). A pointer is unused if the oscilloscope is not collecting data from that channel. If a pointer is `NULL`, nothing will be written to it. `overflow`, a bit pattern indicating whether an overflow has occurred on a channel. Bit 0 is the LSB. Bit 0 --> channel A Bit 1 --> channel B Bit 2 --> channel C Bit 3 --> channel D `time_units`, which can be one of: `PS3000_FS` (=0, femtoseconds), `PS3000_PS` (=1, picoseconds), `PS3000_NS` (=2, nanoseconds, default), `PS3000_US` (=3, microseconds), `PS3000_MS` (=4, milliseconds) or `PS3000_S` (=5, seconds). `no_of_values`, the number of data points to return. In streaming mode, this is the maximum number of values to return.
Returns	The actual number of data values per channel returned, which may be less than `no_of_values` if streaming. `0` if one or more of the parameters are out of range or if the times will overflow with the `time_units` requested. Use ps3000_get_timebase() 21 to acquire the most suitable `time_units`.

4.2.9.18 ps3000_run_streaming_ns

```
short ps3000_run_streaming_ns (
      short                 handle,
      unsigned long         sample_interval,
      PS3000_TIME_UNITS     time_units,
      unsigned long         max_samples,
      short                 auto_stop,
      unsigned long         noOfSamplesPerAggregate,
      unsigned long         overview_buffer_size )
```

This function tells the scope unit to start collecting data in fast streaming mode 14. The function returns immediately without waiting for data to be captured. After calling this function, you should next call ps3000_get_streaming_last_values() 33 to copy the data to your application's buffer.

Applicability	Fast streaming 14 mode only. PicoScope 3224 and 3424 variants only.
Arguments	`handle`, the handle to the required device. `sample_interval`, the time interval, in `time_units`, between data points. `time_units`, the units in which `sample_interval` is measured. `max_samples`, the maximum number of samples that the driver should store from each channel. Your computer must have enough physical memory for this many samples, multiplied by the number of channels in use, multiplied by the number of bytes per sample. `auto_stop`, a boolean to indicate whether streaming should stop automatically when `max_samples` is reached. Set to any non-zero value for `TRUE`. `noOfSamplesPerAggregate`, the number of incoming samples that the driver will merge together (or aggregate: see aggregation 44) to create each value pair passed to the application. The value must be between 1 and `max_samples`. `overview_buffer_size`, the size of the overview buffers, temporary buffers used by the driver to store data before passing it to your application. You can check for overview buffer overruns using the ps3000_overview_buffer_status() 38 function and adjust the overview buffer size if necessary. We recommend using an initial value of 15,000 samples.
Returns	`1` if streaming has been enabled correctly, `0` if a problem occurred or a value was out of range.

4.2.9.19 ps3000_get_streaming_last_values

```
short ps3000_get_streaming_last_values (
      short                       handle
      GetOverviewBuffersMaxMin    lpGetOverviewBuffersMaxMin )
```

This function is used to collect the next block of values while fast streaming 14 is running. You must have called ps3000_run_streaming_ns() 32 beforehand to set up fast streaming.

Applicability	Fast streaming 14 mode only. PicoScope 3224 and 3424 variants only. Not compatible with ETS 12 triggering - function has no effect in ETS mode.
Arguments	`handle`, the handle of the required device. `lpGetOverviewBuffersMaxMin`, a pointer to the callback function 34 in your application that receives data from the streaming driver.
Returns	The actual number of data values returned per channel, which may be less than `max_samples` if streaming, where `max_samples` is a parameter passed to ps3000_run_streaming_ns(). 32 `FALSE` if one of the parameters is out of range.

4.2.9.20 Callback function to copy data to buffer

```
void my_get_overview_buffers (
      short            **overviewBuffers,
      short            overflow,
      unsigned long    triggeredAt,
      short            triggered,
      short            auto_stop,
      unsigned long    nValues )
```

This is the callback function in your application that receives data from the driver in fast streaming [14] mode. You pass a pointer to this function to ps3000_get_streaming_last_values() [33], which then calls it back when the data is ready. Your callback function should do nothing more than copy the data to another buffer within your application. To maintain the best application performance, the function should return as quickly as possible without attempting to process or display the data.

The function name `my_get_overview_buffers()` is just for illustration. When you write this function, you can give it any name you wish. The PicoScope driver does not need to know your function's name, as it refers to it only by the address that you pass to ps3000_get_streaming_last_values() [33].

For an example of a suitable callback function, see the C++ sample code [40] included in your PicoScope installation.

Applicability	Fast streaming [14] mode only. PicoScope 3224 and 3424 variants only. Not compatible with ETS [12] triggering - has no effect in ETS mode.
Arguments	`overviewBuffers`, a pointer to a location where ps3000_get_streaming_last_values() [33] will store a pointer to its overview buffers that contain the sampled data. The driver creates the overview buffers when you call ps3000_run_streaming_ns() [32] to start fast streaming. `overflow`, a bit field that indicates whether there has been a voltage overflow on any channel. Bit 0 --> channel A, Bit 1 --> channel B, Bit 2 --> channel C, Bit 3 --> channel D `triggeredAt`, an index into the overview buffers, indicating the sample at the trigger event. Valid only when `triggered` is `TRUE`. `triggered`, a boolean indicating whether a trigger event has occurred and `triggeredAt` is valid. Any non-zero value signifies `TRUE`. `auto_stop`, a boolean indicating whether streaming data capture has automatically stopped. Any non-zero value signifies `TRUE`. `nValues`, the number of values in each overview buffer.

4.2.9.21 ps3000_get_streaming_values

```
unsigned long ps3000_get_streaming_values (
      short            handle,
      double           *start_time,
      short            *pbuffer_a_max,
      short            *pbuffer_a_min,
      short            *pbuffer_b_max,
      short            *pbuffer_b_min,
      short            *pbuffer_c_max,
      short            *pbuffer_c_min,
      short            *pbuffer_d_max,
      short            *pbuffer_d_min,
      short            *overflow,
      unsigned long    *triggerAt,
      short            *triggered,
      unsigned long    no_of_values,
      unsigned long    noOfSamplesPerAggregate )
```

This function is used after the driver has finished collecting data in fast streaming mode. 14 It allows you to retrieve data with different aggregation 44 ratios, and thus zoom in to and out of any region of the data.

Before calling this function, first capture some data in fast streaming mode, stop fast streaming by calling ps3000_stop(), 29 then allocate sufficient buffer space to receive the requested data. The function will store the data in your buffer with values in the range PS3000_MIN_VALUE to PS3000_MAX_VALUE. The special value PS3000_LOST_DATA is stored in the buffer when data could not be collected because of a buffer overrun. (See Scaling 15 for more on data values.)

Each sample of aggregated data is created by processing a block of raw samples. The aggregated sample is stored as a pair of values: the minimum and the maximum values of the block of raw samples.

Applicability	Fast streaming 14 mode only. PicoScope 3224 and 3424 variants only. Not compatible with ETS 12 triggering - function has no effect in ETS mode.

Arguments	`handle,` the handle of the required device. `start_time,` the time in nanoseconds, relative to the trigger point, of the first data sample required. `pbuffer_a_max,pbuffer_a_min,` pointers to two buffers into which the function will write the maximum and minimum aggregated sample values from channel A. `pbuffer_b_max, pbuffer_b_min, pbuffer_c_max, pbuffer_c_min, pbuffer_d_max, pbuffer_d_min,` as the two parameters above but for channels B, C and D `overflow,` a pointer to where the function will write a bit field indicating whether the voltage on each of the input channels has overflowed. Bit 0 --> Channel A, Bit 1 --> Channel B, Bit 2 --> Channel C, Bit 3 --> Channel D `triggerAt,` a pointer to where the function will write an index into the buffers. The index is the number of the sample at the trigger reference point. Valid only when `triggered` is `TRUE`. `triggered,` a pointer to a boolean indicating that a trigger has occurred and `triggerAt` is valid. `no_of_values,` the number of values required. `noOfSamplesPerAggregate,` the number of samples that the driver should combine to form each aggregated value pair. The pair consists of the maximum and minimum values of all the samples that were aggregated. For channel A, the minimum value is stored in the buffer pointed to by `pbuffer_a_min` and the maximum value in the buffer pointed to by `pbuffer_a_max.`
Returns	the number of values written to each buffer, or `0` if a parameter was out of range

4.2.9.22 ps3000_get_streaming_values_no_aggregation

```
unsigned long ps3000_get_streaming_values_no_aggregation (
      short          handle,
      double         *start_time,
      short          *pbuffer_a,
      short          *pbuffer_b,
      short          *pbuffer_c,
      short          *pbuffer_d,
      short          *overflow,
      unsigned long  *triggerAt,
      short          *trigger,
      unsigned long  no_of_values )
```

This function retrieves raw streaming data from the driver's data store after fast streaming [14] has stopped.

Before calling the function, capture some data using fast streaming, stop streaming using ps3000_stop(), [29] and then allocate sufficient buffer space to receive the requested data. The function will store the data in your buffer with values in the range `PS3000_MIN_VALUE` to `PS3000_MAX_VALUE`. The special value `PS3000_LOST_DATA` is stored in the buffer when data could not be collected because of a buffer overrun. (See Scaling [15] for more details of data values.)

Applicability	Fast streaming [14] mode only. PicoScope 3224 and 3424 variants only. Not compatible with ETS [12] triggering - has no effect in ETS mode.
Arguments	`handle,` the handle of the required device. `start_time,` the time in nanoseconds of the first data sample required. `pbuffer_a, pbuffer_b, pbuffer_c, pbuffer_d,` pointers to buffers into which the function will write the raw sample values from channels A, B, C and D. `overflow,` a pointer to where the function will write a bit field indicating whether the voltage on each of the input channels has overflowed. Bit 0 --> Channel A, Bit 1 --> Channel B, Bit 2 --> Channel C, Bit 3 --> Channel D `triggerAt,` a pointer to where the function will write an index into the buffers. The index is the number of the the sample at the trigger reference point. Valid only when `trigger` is `TRUE`. `trigger,` a pointer to a boolean indicating that a trigger has occurred and `triggerAt` is valid. `no_of_values,` the number of values required.
Returns	the number of values written to each buffer, or `0` if a parameter was out of range

4.2.9.23 ps3000_overview_buffer_status

```
short ps3000_overview_buffer_status (
      short    handle,
      short    *previous_buffer_overrun )
```

This function indicates whether or not the overview buffers used by ps3000_run_streaming_ns() 32 have overrun. If an overrun occurs, you can choose to increase the `overview_buffer_size` argument that you pass in the next call to ps3000_run_streaming_ns() 32.

Applicability	Fast streaming 14 mode only. PicoScope 3224 and 3424 variants only. Not compatible with ETS 12 triggering - function has no effect in ETS mode.
Arguments	`handle`, the handle of the required device. `previous_buffer_overrun`, a pointer to a boolean indicating whether the overview buffers have overrun. Any non-zero value indicates a buffer overrun.
Returns	`0` if the function was successful `1` if the function failed due to an invalid `handle`

4.2.9.24 ps3000_close_unit

```
short ps3000_close_unit ( short handle )
```

Shuts down a PicoScope 3000 Series PC Oscilloscope.

Applicability	All modes.
Arguments	`handle`, the handle, returned by ps3000_open_unit() 17, of the oscilloscope being closed.
Returns	`1` if a valid handle is passed, `0` if not.

4.3 Programming examples

Your PicoScope installation includes programming examples in the following languages and development environments:

4.3.1 C

There are two **C** example programs: one is a simple GUI application, and the other is a more comprehensive console mode program that demonstrates all of the facilities of the driver.

The GUI example program is a generic Windows application - that is, it does not use Borland AppExpert or Microsoft AppWizard. To compile the program, create a new project for an Application containing the following files from the `Examples/ps3000/` subdirectory of your PicoScope installation:

- `ps3000.c`;
- `ps3000.rc`; and

- `ps3000bc.lib` (Borland 32-bit applications); or
- `ps3000.lib` (Microsoft Visual C 32-bit applications)

The following files must be in the compilation directory:

- `ps3000.rch`;
- `ps3000.h`;

and the following file must be in the same directory as the executable.

- `ps3000.dll`

The console example program is a generic windows application - that is, it does not use Borland AppExpert or Microsoft AppWizard. To compile the program, create a new project for an Application containing the following files:

- `ps3000con.c`; and

- `ps3000bc.lib` (Borland 32-bit applications); or
- `ps3000.lib` (Microsoft Visual C 32-bit applications).

The following files must be in the compilation directory:

- `ps3000.h`;

and the following file must be in the same directory as the executable.

- `ps3000.dll`

4.3.2 C++

The **C++** example program shows how to use the fast streaming mode [14] in the driver, with and without triggering [11], and demonstrates the `auto_stop` feature. It runs in console mode and requires a PicoScope 3224 or 3424 scope unit.

You will need to compile the following files that are supplied in the `Examples/ps3000/` subdirectory of your PicoScope installation:

- `ps3000.h`
- `small.ico`
- `stdafx.cpp`
- `stdafx.h`
- `streamingTests.cpp`
- `streamingTests.h`
- `streamingTests.ico`
- `streamingTests.rc`
- `streamingTestsResource.h` (rename to `resource.h` before compiling)

You will also need one of the following libraries, depending on whether you are using Borland or Microsoft C++:

- `ps3000.lib` (Microsoft Visual C 32-bit applications)
- `ps3000bc.lib` (Borland 32-bit applications); or

Ensure that the program directory contains a copy of

- `ps3000.dll`

from the PicoScope installation directory.

4.3.3 Visual Basic

The `Examples/ps3000/` subdirectory of your PicoScope installation contains the following files:

- `ps3000.vbp` - project file
- `ps3000.bas` - procedure prototypes
- `ps3000.frm` - form and program

Note**:** The functions which return a `TRUE`/`FALSE` value, return 0 for `FALSE` and 1 for `TRUE`, whereas Visual Basic expects 65 535 for `TRUE`. Check for `>0` rather than `=TRUE`.

4.3.4 Delphi

The program

- ps3000.dpr

in the Examples/ps3000/subdirectory of your PicoScope installation demonstrates how to operate PicoScope 3000 Series PC Oscilloscopes. The file

- ps3000.inc

contains procedure prototypes that you can include in your own programs. Other required files are:

- ps300fm.res,
- ps300fm.dfm and
- ps3000fm.pas.

This has been tested with Delphi versions 3.

4.3.5 Excel

1 Load the spreadsheet ps3000.xls
2 Select **Tools | Macro**
3 Select **GetData**
4 Select **Run**

Note**:** The Excel macro language is similar to Visual Basic. The functions which return a TRUE/FALSE value, return 0 for FALSE and 1 for TRUE, whereas Visual Basic expects 65 535 for TRUE. Check for >0 rather than =TRUE.

4.3.6 Agilent VEE

The example function ps3000.vee is in the Examples/ps3000/ subdirectory of your PicoScope installation. It uses procedures that are defined in ps3000.vh. It was tested using Agilent VEE version 5.

4.3.7 LabView

The PS3000.vi example in the Examples/ps3000/ subdirectory of your PicoScope installation shows how to access the driver functions using LabVIEW. It was tested using version 6.1 of LabVIEW for Windows. To use the example, copy these files to your LabVIEW directory:

- PS3000.vi
- open_unit.vi
- set_channel.vi
- setup_data_collection.vi
- signal_generator.vi

You will also need

- PS3000.dll

from the installation directory.

5 Troubleshooting

5.1 Software error codes

Consult this section if your are a PicoScope or PicoLog user. If you are writing your own program, refer to the driver error codes 42 section.

Error code	Meaning
1	More than 4 PicoScope 3000 Series oscilloscopes are opened on one machine using PicoLog. It is not possible to use more than 4 oscilloscopes with PicoLog.
2	The driver cannot allocate enough of the computer's memory to operate the oscilloscope. Consult the system requirements 7 section for more information.
3	A PicoScope 3000 Series PC Oscilloscope could not be found on your machine. Make sure the software is installed before the oscilloscope is plugged into the USB socket and restart your computer.
4, 5 or 6	There is a problem with the oscilloscope itself. These problems could arise from configuration settings being corrupted, or a firmware or hardware error.
7	The operating system is not recent enough to support the PicoScope 3000 Series PC Oscilloscope. Consult the system requirements 7 section for more information.

5.2 Driver error codes

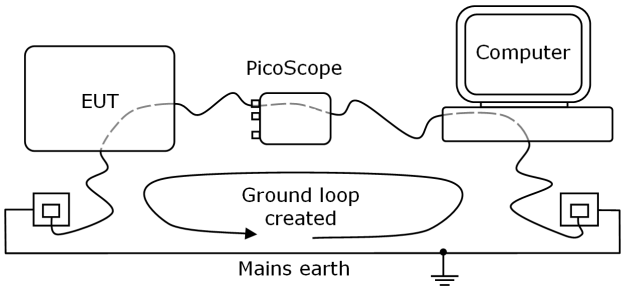
This description of the **driver error codes** is aimed at those people who intend to write their own programs for use with the driver. If the PicoScope or PicoLog software reports an error, refer to the Troubleshooting 42 section.

Code	Name	Description
0	PS3000_OK	The oscilloscope is functioning correctly.
1	PS3000_MAX_UNITS_OPENED	Attempts have been made to open more than PS3000_MAX_UNITS.
2	PS3000_MEM_FAIL	Not enough memory could be allocated on the host machine.
3	PS3000_NOT_FOUND	An oscilloscope could not be found.
4	PS3000_FW_FAIL	Unable to download firmware.
5	PS3000_NOT_RESPONDING	The oscilloscope is not responding to commands from the PC.
6	PS3000_CONFIG_FAIL	The configuration information in the oscilloscope has become corrupt or is missing.
7	PS3000_OS_NOT_SUPPORTED	The operating system is not Windows 98SE, ME, 2000 or XP.

5.3 Powering the 3204, 3205 and 3206

The **PicoScope 3204, 3205 and 3206** PC Oscilloscopes are normally powered from the USB port of the computer. If the computer and the equipment under test (labelled "EUT" in the diagram below) are both referenced to the same ground, a "ground loop" may be created. This may degrade the DC accuracy and noise performance when measuring small signals.

Typically, a ground loop is created when the oscilloscope is connected to a mains-powered computer and is used to measure a signal on another mains-powered device. In this case the ground loop is created through mains earth, as illustrated below:



The majority of laptop power supplies (chargers) are floating and have no ground reference. If, however, connecting your grounded laptop power supply causes noise/offset problems, you can either use the oscilloscope with the laptop running on its batteries or power the oscilloscope using the supplied mains adaptor.

If necessary, you should plug the mains adaptor into the socket on the back of the oscilloscope (near the USB socket). It can be safely connected and disconnected during operation without risk of damage to the oscilloscope.

6 Glossary

AC/DC switch
To switch from AC coupling to DC coupling, or vice versa, select AC or DC from the control on the oscilloscope toolbar of the PicoScope software application. The AC setting filters out any DC component of the input signal, and is suitable for viewing small AC signals superimposed on a DC or slowly-changing offset. In this mode you can measure the peak-to-peak amplitude of an AC signal but not its absolute value. Use the DC setting for measuring the absolute value of a signal.

Aggregation
In fast streaming mode 14, the PicoScope 3000 driver can use a method called aggregation to reduce the amount of data your application needs to process. This means that for every block of consecutive samples, it stores only the minimum and maximum values. You can set the number of samples in each block, called the aggregation parameter, when you call ps3000_run_streaming_ns() 32 for real-time capture, and when you call ps3000_get_streaming_values() 35 to obtain post-processed data.

Analog bandwidth
The input frequency at which the measured signal amplitude is 3 decibels below its true value.

API
Application Programming Interface. A set of function calls that give programmers access to the PicoScope 3000 Series driver.

Block mode
A sampling mode in which the computer prompts the oscilloscope to collect a block of data into its internal memory before stopping the oscilloscope and transferring the whole block into computer memory. Choose this mode of operation when the input signal being sampled contains high frequencies. Note: To avoid sampling errors, the maximum input frequency must be less than half the sampling rate.

Buffer size
The size of the oscilloscope buffer memory, measured in samples. In block mode, the buffer memory is used by the oscilloscope to store data temporarily. This allows the oscilloscope to sample data independently of the speed at which it can transfer data to the computer.

Device Manager
Device Manager is a Windows program that displays the current hardware configuration of your computer. On Windows 98SE or Windows ME, right click on 'My Computer' and choose the 'Device Manager' tab. On Windows 2000 or XP, right-click on 'My Computer,' choose 'Properties', then click the 'Hardware' tab and the 'Device Manager' button.

Driver
A program that controls a piece of hardware. The driver for the PicoScope 3000 Series PC Oscilloscopes is supplied in the form of a 32-bit Windows DLL, `ps3000.dll`. This is used by the PicoScope and PicoLog software, and by user-designed applications, to control the oscilloscopes.

ETS
Equivalent Time Sampling. ETS constructs a picture of a repetitive signal by accumulating information over many similar wave cycles. This means the oscilloscope can capture fast-repeating signals that have a higher frequency than the maximum sampling rate. Note: ETS should not be used for one-shot or non-repetitive signals.

External trigger
This is the BNC socket marked **E** on the PicoScope 3204/5/6 PC Oscilloscopes. It can be used to start a data collection run but cannot be used to record data. As it shares the same connector as the signal generator output, these two functions cannot be used at the same time. It is possible, however, to use the output from the signal generator as a trigger.

Maximum sampling rate
A figure indicating the maximum number of samples the oscilloscope can acquire per second. Maximum sample rates are usually given in MS/s (megasamples per second) or GS/s (gigasamples per second.) The higher the sampling rate of the oscilloscope, the more accurate the representation of the high-frequency details in a fast signal.

Oversampling
Oversampling is taking more than one measurement during a time interval and returning an average. If the signal contains a small amount of noise, this technique can increase the effective vertical resolution of the oscilloscope.

PC Oscilloscope
The instrument formed by connecting a PicoScope 3000 Series PC Oscilloscope to a computer running the PicoScope software application.

PicoLog software
This is a software product that accompanies all our oscilloscopes. It turns your PC into a data logger and chart recorder.

PicoScope 3000 Series
An oscilloscope range comprising the PicoScope 3204, 3205, 3206, 3224 and 3424 PC Oscilloscopes.

PicoScope software
This is a software product that accompanies all our oscilloscopes. It turns your PC into an oscilloscope, spectrum analyser, and meter display.

Signal generator
This is a feature of some oscilloscopes which allows a signal to be generated without an external input device being present. The signal generator output is the BNC socket marked **E** on the oscilloscope. If you connect a BNC cable between this and one of the channel inputs, you can send a signal into one of the channels. On some units, the signal generator can generate a simple TTL square wave, while on others it can generate a sine, square or triangle wave that can be swept back and forth. Consult the specifications 9 for further details.

Note: The signal generator output is physically the same as the external trigger input, so these two functions cannot be used at the same time. It is possible, however, to use the output from the signal generator as a trigger.

Streaming mode
A sampling mode in which the oscilloscope samples data and returns it to the computer in an unbroken stream. This mode allows the capture of data sets whose size is not limited by the size of the scope's memory buffer, at sampling rates up to a few million samples per second.

Temperature range
The minimum and maximum temperatures between which the oscilloscope is guaranteed to meet its specifications. The 3204/5/6 PC Oscilloscopes are specified at a nominal temperature of 25°C, and the 3224/3424 are specified over the range 20°C to 30°C.

Timebase
The timebase controls the time interval that the width of the scope display represents. If you select **Timebase is time per division** in the **Preferences** dialog box, it works like a traditional bench top scope. There are ten divisions across the screen, so the total time interval is ten times the timebase.

Trigger bandwidth
The maximum frequency at which the trigger circuit will reliably generate a trigger event.

USB 1.1
Universal Serial Bus (Full Speed). This is a standard port that enables you to connect external devices to PCs. A typical USB 1.1 port supports a data transfer rate of 12 megabits per second, and is much faster than a serial port.

USB 2.0
Universal Serial Bus (High Speed). This is a standard port that enables you to connect external devices to PCs. A typical USB 2.0 port supports a data transfer rate 40 times faster than USB 1.1, and all USB 2.0 ports are backwards-compatible with USB 1.1.

Vertical resolution
A value, in bits, indicating the degree of precision with which the oscilloscope can convert input voltages to digital values. Oversampling 15 can improve the effective resolution.

Voltage range
The range of input voltages that the oscilloscope will measure in a given mode.

Index

A

B

C

D

E

F

G

H

I

L

M

N

O

P

R

S

T

U

V

W

Pico Technology Ltd

The Mill House
Cambridge Street
St Neots PE19 1QB
United Kingdom
Tel: +44 (0)1480 396 395
Fax: +44 (0)1480 396 296
Web: www.picotech.com

PS3000044-1.5 14.7.06