PicoScope 3425
PC Oscilloscope

User's Manual

ps3425.en-2

Contents

1 Introduction

1.1 Overview

The **PicoScope 3425** PC Oscilloscope 57 is a high-precision differential oscilloscope. (What is a differential oscilloscope?) 10 It is fully USB 2.0 58-capable and backwards-compatible with USB 1.1 58. There is no need for an external power supply as power is supplied from the USB port, making these oscilloscopes highly portable.

With the PicoScope 57 software, the PicoScope 3425 can be used as a PC Oscilloscope 57 and spectrum analyser 57. Alternatively, using the API functions, you can develop your own programs to collect and analyse data from the oscilloscope.

1.2 Minimum PC requirements

For the PicoScope 3425 PC Oscilloscope to operate correctly, you must connect it to a computer with the minimum requirements to run Windows or the following (whichever is the higher specification):

Processor	Pentium-class processor or equivalent minimum.
Memory	256 MB minimum.
Disk space	10 MB minimum.
Operating system	Microsoft Windows XP SP2 or Vista.
Ports	USB 1.1 58 compliant port minimum. USB 2.0 58 compliant port recommended. Must be connected directly to the port or a powered USB hub. Will not work on a passive hub.

1.3 Safety information

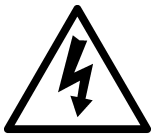
We strongly recommend that you read the general safety information below before using your oscilloscope for the first time. If you use the oscilloscope in a manner contrary to these instructions, safety protection built in to the equipment may cease to function. This could cause damage to your computer or other equipment, or lead to injury to yourself and others.

Maximum input range

The PicoScope 3425 PC Oscilloscope is designed to measure voltages in the range stated in the Specifications 9 table. The oscilloscope can withstand the **Overvoltage** stated in the Specifications 9 table, and operation with voltages exceeding this range may cause physical damage.

Mains voltages and measurement category

The PicoScope 3425 PC Oscilloscope is designed only for **CAT I** measurements as defined by **BS EN61010-1:2001,** which permits measurements on circuits that are not directly connected to the mains (line power). The oscilloscope is not designed for measurements on CAT II, III or IV circuits.



The PicoScope 3425 PC Oscilloscope must not be directly connected to the mains (line power).

Failure to heed this warning may lead to injury or death.

Safety grounding

The PicoScope 3425 PC Oscilloscope connects directly to the ground of a computer through the USB cable provided. This connection is intended only to minimise interference, and therefore you **must not** rely on it as a protective safety ground.

Do not connect the ground sockets 7 on the front panel to any source other than ground. If in doubt, use a meter to check that there is no significant AC or DC voltage between the oscilloscope's ground socket and point to which you intend to connect it. Failure to check may cause damage to your computer, or injury to yourself and others.

Repairs

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

Cleaning and decontamination

- Remove all connections from the unit
- Clean the external surfaces of the oscilloscope with a soft damp cloth. Do not use chemical cleaners.
- Make sure that the instrument is completely dry before using again.

1.4 Safety symbols

Symbol 1: Caution: risk of electric shock

This symbol indicates that a safety hazard exists on the indicated connections if you do not take correct precautions. Ensure that you read in detail all safety documentation associated with the product before using it.

Symbol 2: Equipotentiality

This symbol indicates that the indicated connectors are all at the same potential (i.e. are shorted together). You must therefore take necessary precautions to avoid applying a potential across the indicated terminals as this may result in a large current, causing damage to the product and connected equipment.

1.5 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 2.

1.6 CE notice

The PicoScope 3425 PC Oscilloscope meets the intent of the EMC directive **89/336/EEC** and is designed to the **EN61326-1 (1997) Class B Emissions and Immunity** standard.

The oscilloscope also meets the intent of the **Low Voltage Directive** and is designed to the **BS EN 61010-1:2001 / IEC 61010-1:2001** (safety requirements for electrical equipment for measurement, control, and laboratory use) standard.

1.7 Legal information

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

As 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 use in mission-critical applications, for example life support systems.

Viruses

This software was continuously monitored for viruses during production, but you are responsible for virus-checking the software once it is installed.

Support

If you are dissatisfied with the performance of this software, please contact our technical support staff, who will try to fix the problem within a reasonable time. If you are still dissatisfied, please return the product and software to your supplier within 28 days of purchase for a full refund.

Upgrades

We provide upgrades, free of charge, from our web site at www.picotech.com. We reserve the right to charge for updates or replacements sent out on physical media.

Trademarks

Windows is a trademark or registered trademark of Microsoft Corporation. Pico Technology Limited and PicoScope are internationally registered trademarks.

1.8 Company details

You can obtain technical assistance from Pico Technology at the following address:

Address: Pico Technology
James House
Colmworth Business Park
Eaton Socon
St. Neots
PE19 8YG
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

2 Product information

2.1 Pack contents

The PicoScope 3425 pack contains the following items:

Reorder code	Description	Quantity
PR090	PicoScope 3425 Differential PC Oscilloscope	1
MI106	USB cable, for use with USB 1.1 and USB 2.0 ports	1
TA039	Screened twisted-pair lead, 1.3 metre	4
TA038	Current clamp adaptor	4
TA001	Black test probe	1
TA002	Red test probe	1
TA005	Black dolphin clip	4
TA006	Red dolphin clip	4
DO115	Quick Start Guide	1
DI025	PicoScope software and reference CD	1

The accessories supplied with the PicoScope 3425 are rated for safe working at the maximum voltages stated in the Specifications. 9 For your safety, if you use your own accessories with this oscilloscope, you **must** ensure that they are rated for the voltage you are measuring.

2.2 Installation instructions

Important

You must install the PicoScope software 57 before connecting a PicoScope 3425 PC Oscilloscope to your PC for the first time.

Install the software by following the steps in the quick start guide supplied with your oscilloscope. You can then connect your oscilloscope to the PC.

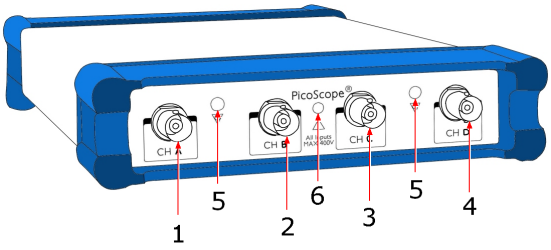
There is no need for an additional power supply, as the unit draws its power from the USB port.

Checking the installation

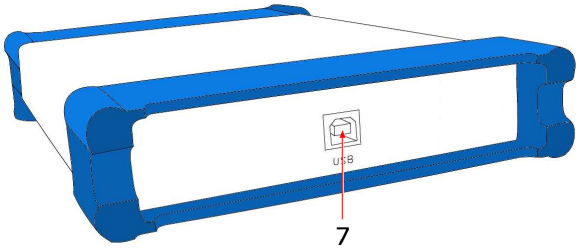
Once the software has been installed, ensure that the oscilloscope is connected to the PC and then start the PicoScope software. The software should now display the voltage of any signal that is connected to the oscilloscope. If you are using the differential cable and test probes supplied, you should see a small 50 or 60 hertz noise signal in the oscilloscope window when you touch the test probes with your fingers.

See Connector diagrams 7.

2.3 Connector diagrams



Front view (3425)



Rear view

1. **Ch A.** Input channel A. As this is a differential oscilloscope, 10 the channels have non-standard input impedances and **cannot be used with passive attenuated scope probes** such as the x1/x10 probes supplied with conventional oscilloscopes.

2. **Ch B.** Input channel B. Has the same characteristics as **Ch A**.

3. **Ch C.** Input channel C. Has the same characteristics as **Ch A**.

4. **Ch D.** Input channel D. Has the same characteristics as **Ch A**.

5. **Ground.** Can be used with the current clamp adaptor supplied to convert one or more of the differential inputs to single-ended inputs. Can also be used to ground the screen of the shielded twisted-pair cable supplied.

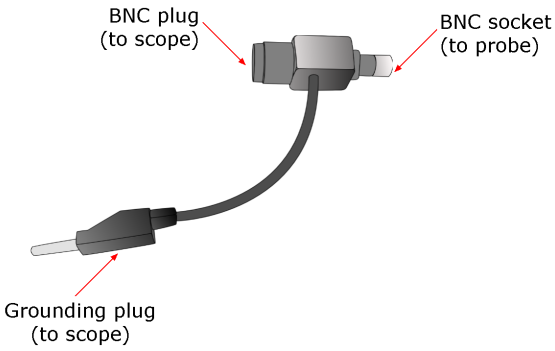
   The scope's ground is connected to the PC's ground through the USB cable. You MUST NOT rely on the scope's ground as a protective safety ground.

6. **LED.** Lights up when the oscilloscope is first powered up, switches off when the PicoScope software begins to run, and then flashes while the oscilloscope is capturing data.

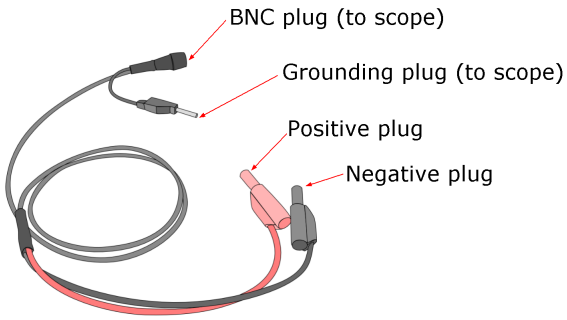
7. **USB port.** Compatible with USB 58 1.1 and USB 2.0 ports.

2.4 Connecting the oscilloscope

The oscilloscope is supplied with the following cables and adaptors:



TA038 Current clamp adaptor



TA039 Screened twisted-pair cable

TA038 Current clamp adaptor

Use this adaptor to connect a current clamp to the oscilloscope. First fit the adaptor to one of the BNC inputs on the oscilloscope's front panel, then insert the adaptor's grounding plug into one of the ground sockets on the front panel.

It is important to ground the current clamp using the procedure just described. Most current clamps have an internal metal case that readily picks up electromagnetic noise from the environment, and if the case were not grounded, this noise would interfere with the signal.

TA039 Screened twisted-pair cable

Use this cable to connect a signal directly to the oscilloscope. A positive signal will be displayed when you connect the red BNC plug to the more positive signal and the black BNC plug to the more negative signal. If you connect these the wrong way round, the oscilloscope will not be damaged but the signal will appear inverted on the scope display.

Insert the grounding plug into one of the ground sockets on the front of the oscilloscope. This grounds the screen of the cable to prevent it from picking up electromagnetic noise that might otherwise interfere with the signal.

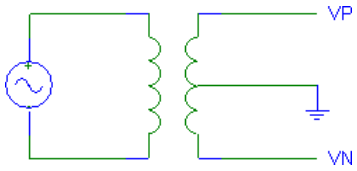
2.5 Specifications

Specification	Value
Vertical resolution 58	12 bits
Analogue bandwidth 56	5 MHz (3 MHz on 100 mV range)
Channels	4
Maximum sampling rate 57	
Single channel Two channels Three or four channels	20 MS/s 10 MS/s 5 MS/s
Capture memory 56	
One channel enabled Two channels enabled Three or four channels enabled	512 k samples 256 k samples 128 k samples
Input type	Differential voltage 10 Selectable AC or DC coupling Touch-proof BNC connectors with 4 mm GND sockets
Input impedance	12.4 MΩ (on 100 mV to 5 V ranges) 10.1 MΩ (on 10 V to 400 V ranges)
Input capacitance	12 pF
Common-mode voltage range 56 to ensure measurement accuracy	30 V (on 100 mV to 5 V ranges) 400 V (on 10 V to 400 V ranges)
Maximum safe voltages	
Differential 57 Any input above scope GND 56	400 V 400 V (600 V transient)
Measurement category rating	CAT I
Voltage ranges 58	100 mV, 200 mV, 500 mV, 1 V, 2 V, 5 V, 10 V, 20 V, 50 V, 100 V, 200 V, 400 V
Accuracy	Voltage: ±1% Time: 50 ppm
Linearity	12 bits
Noise	< 10 LSB
Operating environment	
Temperature range Humidity range	0 °C to 40 °C (20 °C to 30 °C for quoted accuracy) Minimum 5% RH non-condensing Maximum 80% RH non-condensing, decreasing linearly to 50% at 40 °C
Storage environment	
Temperature range Humidity range	-20 °C to 60 °C 5% to 90% RH non-condensing
Other environmental conditions	Dry environments Altitude up to 2000 m No pollution, or only dry, non-conductive pollution
PC connection	USB 2.0 58 Compatible with USB 1.1
Power supply	From USB port 4.6 V to 5.25 V DC @ approx. 500 mA No external power supply required
Dimensions	255 mm x 170 mm x 40 mm (approximately 10.0 in x 6.7 in x 1.6 in)
Weight	920 g (approximately 2 lb)

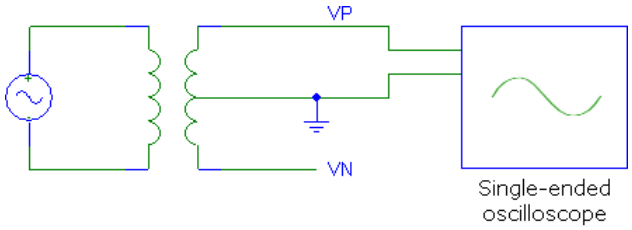
2.6 What is a differential oscilloscope?

The PicoScope 3425 is a **differential PC Oscilloscope.** A differential oscilloscope measures the voltage difference between two points, regardless of the voltage of either point with respect to ground. This is unlike a conventional single-ended oscilloscope, which requires one of the points to be at ground potential.

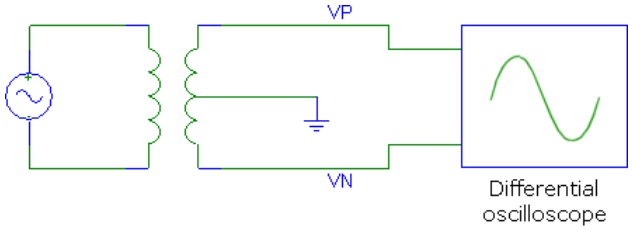
For example, suppose that you want to measure the output voltage of a transformer with a grounded centre tap, as in the diagram below:



A single-ended oscilloscope can only measure a signal with reference to ground, so you need to connect the scope's ground clip to the centre of the secondary. You can then measure either V_P or V_N with respect to ground, but not the total voltage across the secondary. The diagram below shows a single-ended scope connected between V_P and ground:



With a differential oscilloscope, you can directly measure the secondary voltage by connecting the positive probe to V_P and the negative to V_N. This is shown in the diagram below. A ground connection is not essential, although it is a good idea to use a shielded cable grounded at one end in order to prevent electromagnetic noise coupling into the cable.

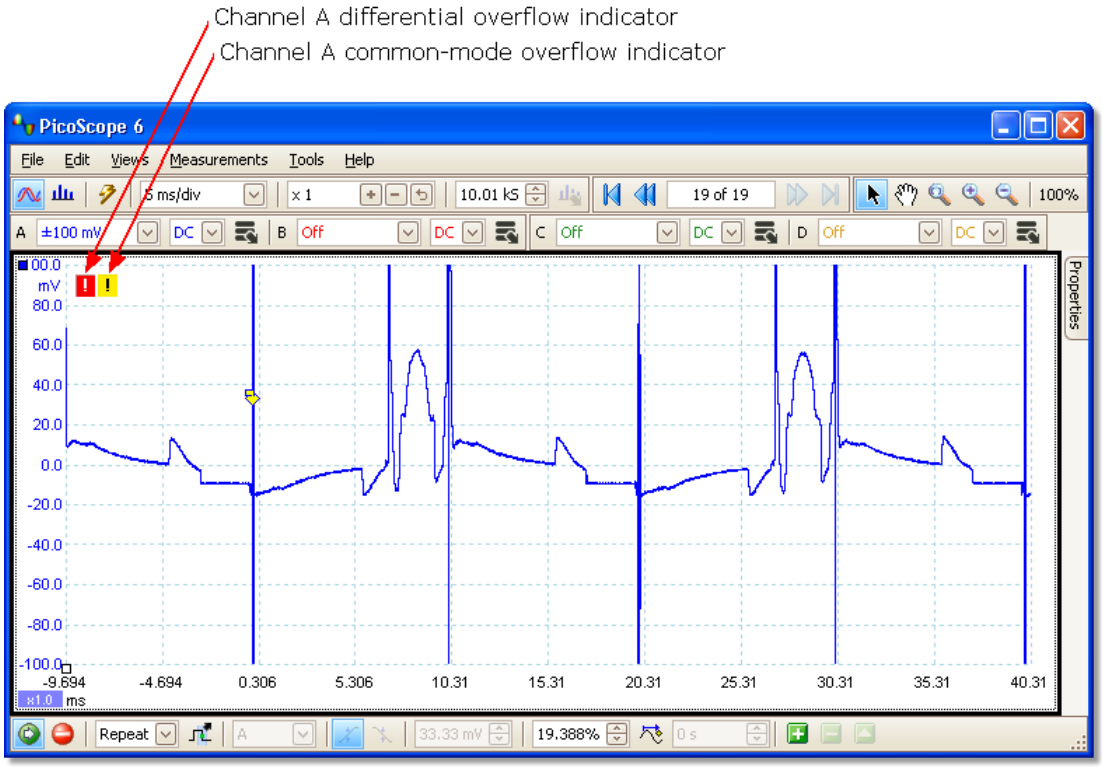


Differential oscilloscopes have many other applications where the voltage to be measured is not referenced to ground, or where it is not desirable to connect the scope ground to the ground of the device under test.

2.7 Overflow indicators

PicoScope 6, the PC Oscilloscope software supplied with the PicoScope 3425, displays a yellow **common-mode overflow indicator** for each channel when either the positive or the negative input voltage with respect to ground is outside the range stated in the Specifications. 9 Exceeding the common-mode range of the scope causes inaccurate measurements and can lead to severe signal distortion.

The red **differential overflow indicator** is used to warn when the differential voltage (the difference between the positive and negative inputs) on each channel exceeds the selected voltage range. This condition causes clipping of the displayed signal.



2.8 AC/DC coupling

Each channel can be set to use either AC or DC coupling. When AC coupling is used, any DC component of the signal below about 1 hertz is filtered out.

To change the coupling mode, select AC or DC from the control on the oscilloscope toolbar of the PicoScope software. The setting should be chosen to suit the characteristics of the input signal.

2.9 Resolution enhance

The hardware resolution of the oscilloscope is 12 bits, but you can obtain an effective resolution of up to 16 bits using the Resolution Enhance feature built in to the PicoScope software. See the PicoScope 6 User's Guide for details.

3 Programmer's reference

3.1 Driver

The Windows XP/Vista 32-bit **driver**, `picopp.sys`, is installed under the control of a Setup 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 3425 PC Oscilloscope for the first time.

3.2 Programming with the PicoScope 3000 Series

The `ps3000.dll` library in your PicoScope installation directory allows you to program a PicoScope 3425 oscilloscope using standard C function calls. 17

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

- Open 18 the scope unit
- Set up the input channels with the required voltage ranges 12 and coupling mode 12
- Set up triggering 12
- Start capturing data. (See Sampling modes 13 for a more detailed discussion of programming.)
- Wait until the scope unit is ready
- Copy data to a buffer
- Stop capturing data
- Close the scope unit

Numerous sample programs 51 are installed with your PicoScope software. These show how to use the functions of the driver software in each of the modes available.

3.2.1 Voltage ranges

It is possible to set the gain for each channel with the ps3000_set_channel() 22 function. This will give an input **voltage range** between ±100 mV and ±400 V.

3.2.2 AC/DC coupling

Using the ps3000_set_channel() 22 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.

3.2.3 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 on any of the conditions available in the simple and advanced triggering modes.

Applicability	Available in block mode 13 and fast streaming mode 15 only. Calls to the ps3000_set_trigger() 25 function have no effect in compatible streaming mode 14.

3.2.4 Sampling modes

PicoScope 3425 PC Oscilloscopes can run in various **sampling modes**.

- **Block mode.** 13 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.** 14 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.

3.2.4.1 Block mode

In **block mode**, the computer prompts a PicoScope 3425 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 3425 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. This feature is 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() 35, ps3000_ready() 37, ps3000_stop() 38 and ps3000_get_values() 39.

See Using block mode 13 for programming details.

3.2.4.2 Using block mode

This is the general procedure for reading and displaying data in block mode: 13

1. Open the oscilloscope using ps3000_open_unit() 18
2. Select channel ranges and AC/DC coupling using ps3000_set_channel() 22
3. Using ps3000_set_trigger() 25, set the trigger if required
4. Using ps3000_get_timebase() 23, select timebases until the required number of nanoseconds per sample is located
5. Start the oscilloscope running using ps3000_run_block() 35
6. Wait until the oscilloscope says it is ready using ps3000_ready() 37
7. Transfer the block of data from the oscilloscope using ps3000_get_values() 39 or ps3000_get_times_and_values() 40
8. Display the data
9. Repeat steps 5 to 8
10. Stop the oscilloscope using ps3000_stop() 38

3.2.4.3 Streaming modes

The **streaming modes** are alternatives to block mode 13 that can capture data without gaps between blocks.

In a streaming mode, the computer prompts the PicoScope 3425 PC Oscilloscope to start collecting data. The data is then transferred back to the PC without being stored in oscilloscope memory. Data can be sampled with a period between 1 µs and 60 s, and the maximum number of samples is limited only by the amount of free space on the PC's hard disk.

There are two streaming modes:

- Compatible streaming mode 14
- Fast streaming mode 15

3.2.4.4 Compatible streaming mode

Compatible streaming mode is a basic streaming mode 14 that works with all scope units, at speeds from one sample per minute to a 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 56 setting) will cause a restart of the data stream. The driver can buffer up to 32 K samples of data per channel, but the user must ensure that the ps3000_get_values() 39 function is called frequently enough to avoid buffer overrun.

See Using compatible streaming mode 14 for programming details.

Applicability	Does not support triggering 12. The ps3000_get_times_and_values() 40 function will always return FALSE (0) in streaming mode.

3.2.4.5 Using compatible streaming mode

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

1. Open the oscilloscope using ps3000_open_unit() 18
2. Select channel ranges and AC/DC switches using ps3000_set_channel() 22
3. Start the oscilloscope running using ps3000_run_streaming() 36
4. Transfer the block of data from the oscilloscope using ps3000_get_values() 39
5. Display the data
6. Repeat steps 4 and 5 as necessary
7. Stop the oscilloscope using ps3000_stop() 38

3.2.4.6 Fast streaming mode

Fast streaming mode is an advanced streaming mode 14 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 56, which allows your application to zoom in and out of the data with the minimum of effort.

Applicability	Works with triggering 12

See Using fast streaming mode 15 for programming details.

3.2.4.7 Using fast streaming mode

This is the general procedure for reading and displaying data in fast streaming mode: 15

1. Open the oscilloscope using ps3000_open_unit() 18
2. Select channel ranges and AC/DC switches using ps3000_set_channel() 22
3. Set the trigger using ps3000_set_trigger() 25
4. Start the oscilloscope running using ps3000_run_streaming_ns() 41
5. Get a block of data from the oscilloscope using ps3000_get_streaming_last_values() 42
6. Display or process the data
7. If required, check for overview buffer overruns by calling ps3000_overview_buffer_status() 48
8. Repeat steps 5 to 7 as necessary or until auto_stop is TRUE
9. Stop fast streaming using ps3000_stop() 38
10. Retrieve any part of the data at any time scale by calling ps3000_get_streaming_values() 44
11. If you require raw data, retrieve it by calling ps3000_get_streaming_values_no_aggregation() 46
12. Repeat steps 10 to 11 as necessary
13. Close the oscilloscope by calling ps3000_close_unit() 50

3.2.5 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 58 of the oscilloscope by the amount given by the equation below:

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

Applicability	Available in block mode 13 only

3.2.6 Scaling

The PicoScope 3425 PC Oscilloscope has a resolution of 12 bits, but the oscilloscope driver normalises all readings to 16 bits. This enables it to take advantage of noise reduction from oversampling 11, 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 15 mode.
PS3000_MIN_VALUE	-32 767	Negative full scale
0	0	Zero volts
PS3000_MAX_VALUE	32 767	Positive full scale

3.2.7 Combining oscilloscopes

It is possible to collect data using up to four PicoScope 3000 Series PC Oscilloscopes 57 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() 18 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: It is not possible to synchronise the collection of data between oscilloscopes that are being used in combination.

3.2.8 Functions

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

The following user-defined functions are also described:

3.2.8.1 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

3.2.8.2 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() 20 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

3.2.8.3 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() 19.

Applicability	All modes Use only with ps3000_open_unit_async() 19
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

3.2.8.4 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 of the device from which information 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 0 if one of the parameters is out of range, or a null pointer is passed for `string`

`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 PC Oscilloscope that is attached to the computer.	`"3425"`
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.	`"21Oct07"`
6	`PS3000_ERROR_CODE`	One of the Error codes 55.	`"4"`

3.2.8.5 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 56 mode and the input range.

Applicability	All modes
Arguments	`handle`, the handle of the required device. `channel`, an enumerated type. Use `PS3000_CHANNEL_A (0)` or `PS3000_CHANNEL_B (1)`. `enabled`, specifies if the channel is active: `TRUE`=active, `FALSE`=inactive. `dc`, specifies the AC/DC coupling 56 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
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
11	PS3000_50V	±50 V
12	PS3000_100V	±100 V
13	PS3000_200V	±200 V
14	PS3000_400V	±400 V

3.2.8.6 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 [57] are available on the oscilloscope. You should set up the channels using ps3000_set_channel() [22] 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 nanoseconds, between readings at the selected timebase. If a null pointer is passed, nothing will be written here. `time_units,` a pointer to the most suitable units that the results should be measured in. This value should also be passed when calling ps3000_get_times_and_values() [40]. 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 [11] 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 0 on error

3.2.8.7 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 required device.
Returns	`1` if a valid handle is passed `0` if handle is invalid

3.2.8.8 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 56 and fast streaming mode 15.
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)` or `PS3000_NONE(5)`. `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 `1` if successful

3.2.8.9 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 56 and fast streaming mode 15 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)` or `PS3000_NONE (5)`. `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 ±20 V. `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 1 if successful

3.2.8.10 ps3000SetAdvTriggerChannelProperties

```
short ps3000SetAdvTriggerChannelProperties(
  short                          handle,
  TRIGGER_CHANNEL_PROPERTIES * channelProperties,
  short                          nChannelProperties,
  long                           autoTriggerMilliseconds);
```

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

Applicability	All modes
Arguments	`handle`, the handle of the required device. `channelProperties`, a pointer to a `TRIGGER_CHANNEL_PROPERTIES` structure describing the requested properties. If NULL, triggering is switched off. `nChannelProperties`, should be set to 1 if `channelProperties` is non-null, otherwise 0. `autoTriggerMilliseconds`, the time in milliseconds for which the scope device will wait before collecting data if no trigger event occurs. If this is set to zero, the scope device will wait indefinitely for a trigger.
Returns	`0` if unsuccessful, or if one or more of the arguments are out of range `1` if successful

3.2.8.10.1 TRIGGER_CHANNEL_PROPERTIES structure

A structure of this type is passed to ps3000SetAdvTriggerChannelProperties() 27 in the `channelProperties` argument to specify the trigger mechanism, and is defined as follows: -

```
typedef struct tTriggerChannelProperties
{
  short          thresholdMajor;
  short          thresholdMinor;
  unsigned short hysteresis;
  short          channel;
  THRESHOLD_MODE thresholdMode;
} TRIGGER_CHANNEL_PROPERTIES;
```

Applicability	All modes

Members	`thresholdMajor`, the upper threshold at which the trigger event is to take place. This is scaled in 16-bit ADC counts at the currently selected range for that channel.
	
	`thresholdMinor`, the lower threshold at which the trigger event is to take place. This is scaled in 16-bit ADC counts at the currently selected range for that channel.
	
	`hysteresis`, the hysteresis that the trigger has to exceed before it will fire. It is scaled in 16-bit counts.
	
	`channel`, the channel to which the properties apply.
	
	`thresholdMode`, either a level or window trigger. Use one of these constants: -
	`LEVEL (0)`
	`WINDOW(1)`

3.2.8.11 ps3000SetAdvTriggerChannelConditions

```
short ps3000SetAdvTriggerChannelConditions(
  short                handle,
  TRIGGER_CONDITIONS * conditions,
  short                nConditions);
```

This function sets up trigger conditions on the scope's inputs. The trigger is set up by defining a TRIGGER_CONDITIONS 30 structure. Each structure is the AND of the states of one scope input.

Applicability	All modes
Arguments	`handle,` the handle of the required device. `conditions,` a pointer to a `TRIGGER_CONDITIONS` structure specifying the conditions that should be applied to the current trigger channel. If NULL, triggering is switched off. `nConditions,` should be set to 1 if conditions is non-null, otherwise 0.
Returns	`0` if unsuccessful, or if one or more of the arguments are out of range `1` if successful

3.2.8.11.1 TRIGGER_CONDITIONS structure

A structure of this type is passed to ps3000SetAdvTriggerChannelConditions() 29 in the `conditions` argument to specify the trigger conditions, and is defined as follows: -

```
typedef struct tTriggerConditions
{
  TRIGGER_STATE channelA;
  TRIGGER_STATE channelB;
  TRIGGER_STATE channelC;
  TRIGGER_STATE channelD;
  TRIGGER_STATE external;
  TRIGGER_STATE pulseWidthQualifier;
} TRIGGER_CONDITIONS;
```

Applicability	All modes

Members	`channelA, channelB, channelC, channelD, pulseWidthQualifier:` the type of condition that should be applied to each channel. Use these constants: -
	
	`CONDITION_DONT_CARE (0)`
	`CONDITION_TRUE (1)`
	`CONDITION_FALSE (2)`
	
	`external,` not used

Remarks

The channels that are set to `CONDITION_TRUE` or `CONDITION_FALSE` must all meet their conditions simultaneously to produce a trigger. Channels set to `CONDITION_DONT_CARE` are ignored.

The oscilloscope can only use a single channel for the trigger source. Therefore you must define `CONDITION_TRUE` or `CONDITION_FALSE,` and the pulse width qualifier if required, for only one channel at a time.

3.2.8.12 ps3000SetAdvTriggerChannelDirections

```
short ps3000SetAdvTriggerChannelDirections(
  short handle,
  THRESHOLD_DIRECTION channelA,
  THRESHOLD_DIRECTION channelB,
  THRESHOLD_DIRECTION channelC,
  THRESHOLD_DIRECTION channelD,
  THRESHOLD_DIRECTION ext);
```

This function sets the direction of the trigger for each channel.

Applicability	All modes
Arguments	`handle`, the handle of the required device `channelA`, `channelB`, `channelC`, `channelD`, all specify the direction in which the signal must pass through the threshold to activate the trigger. The allowable values for a `THRESHOLD_DIRECTION` variable are listed in the table below. `ext`, not used
Returns	`0` if unsuccessful, or if one or more of the arguments are out of range `1` if successful

THRESHOLD_DIRECTION constants

ABOVE	for gated triggers: above a threshold
BELOW	for gated triggers: below a threshold
RISING	for threshold triggers: rising edge
FALLING	for threshold triggers: falling edge
RISING_OR_FALLING	for threshold triggers: either edge
INSIDE	for window-qualified triggers: inside window
OUTSIDE	for window-qualified triggers: outside window
ENTER	for window triggers: entering the window
EXIT	for window triggers: leaving the window
ENTER_OR_EXIT	for window triggers: either entering or leaving the window
NONE	no trigger

3.2.8.13 ps3000SetPulseWidthQualifier

```
short ps3000SetPulseWidthQualifier(
  short handle,
  PNQ_CONDITIONS      * conditions,
  short                 nConditions,
  THRESHOLD_DIRECTION   direction,
  unsigned long         lower,
  unsigned long         upper,
  PULSE_WIDTH_TYPE      type);
```

This function sets up pulse width qualification, which can be used on its own for pulse width triggering or combined with other triggering to produce more complex triggers. The pulse width qualifier is set by defining a `conditions` structure.

Applicability	All modes
Arguments	`handle`, the handle of the required device `conditions`, a pointer to a `PWQ_CONDITIONS` structure specifying the conditions that should be applied to the trigger channel. If `conditions` is NULL then the pulse width qualifier is not used. `nConditions`, should be set to 1 if `conditions` is non-null, otherwise 0. `direction`, the direction of the signal required to trigger the pulse. `lower`, the lower limit of the pulse width counter. `upper`, the upper limit of the pulse width counter. This parameter is used only when the `type` is set to `PW_TYPE_IN_RANGE` or `PW_TYPE_OUT_OF_RANGE`. `type`, the pulse width type, one of these constants: - `PW_TYPE_NONE` — do not use the pulse width qualifier `PW_TYPE_LESS_THAN` — pulse width less than lower `PW_TYPE_GREATER_THAN` — pulse width greater than lower `PW_TYPE_IN_RANGE` — pulse width between lower and upper `PW_TYPE_OUT_OF_RANGE` — pulse width not between lower and upper
Returns	`0` if unsuccessful, or if one or more of the arguments are out of range `1` if successful

3.2.8.13.1 PWQ_CONDITIONS structure

A structure of this type is passed to ps3000SetPulseWidthQualifier() 32 in the `conditions` argument to specify the pulse-width qualifier conditions, and is defined as follows: -

```
typedef struct tPwqConditions
{
  TRIGGER_STATE channelA;
  TRIGGER_STATE channelB;
  TRIGGER_STATE channelC;
  TRIGGER_STATE channelD;
  TRIGGER_STATE external;
} PWQ_CONDITIONS;
```

Applicability	Pulse-width qualified triggering

Members	`channelA, channelB, channelC, channelD:` the type of condition that should be applied to each channel. Use these constants: -
	
	`CONDITION_DONT_CARE (0)`
	`CONDITION_TRUE (1)`
	`CONDITION_FALSE (2)`
	
	`external`, not used

3.2.8.14 ps3000SetAdvTriggerDelay

```
short ps3000SetAdvTriggerDelay(
  short         handle,
  unsigned long delay,
  float         preTriggerDelay);
```

This function sets the post-trigger delay, which causes capture to start a defined time after the trigger event.

Applicability	All modes
Arguments	`handle,` the handle of the required device `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%. For example, 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.
Returns	`0` if unsuccessful, or if one or more of the arguments are out of range `1` if successful

3.2.8.15 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 13.

Applicability	Block mode 13 only
Arguments	`handle`, the handle of 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 11 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 1 if successful

3.2.8.16 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, use ps3000_run_streaming_ns() [41] instead.

Applicability	Compatible streaming [14] mode only
Arguments	`handle`, the handle of 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() [39] are returned. If this is 1, the number of values requested by ps3000_get_values() [39] are returned, even if they have already been read by ps3000_get_values() [39].
Returns	`1` if streaming has been enabled correctly, `0` if a problem occurred or a value was out of range.

3.2.8.17 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 13 only. Does nothing if the oscilloscope is in streaming mode 14.
Arguments	`handle`, the handle of 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.

3.2.8.18 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 of the required device.
Returns	`0` if an invalid handle is passed `1` if successful

3.2.8.19 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() 36, or in block mode 13 after calling ps3000_run_block() 35.

Applicability	Compatible streaming mode 14 and block mode 13 only. Do not use in fast streaming mode 15 - use ps3000_get_streaming_last_values() 42 instead.
Arguments	`handle`, the handle of 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. The bit assignments are as follows: 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 D C B A D C B A common-mode overflow differential overflow `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.

3.2.8.20 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 [13] after calling ps3000_run_block() [35].

Applicability	Block mode [13] only. It will not return any valid times if the oscilloscope is in streaming mode [14].
Arguments	`handle`, the handle of 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. The bit assignments are as follows: 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 D C B A — common-mode overflow D C B A — differential overflow `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() [23] to acquire the most suitable `time_units`.

3.2.8.21 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 15. The function returns immediately without waiting for data to be captured. After calling this function, you should next call ps3000_get_streaming_last_values() 42 to copy the data to your application's buffer.

Applicability	Fast streaming 15 mode only
Arguments	`handle`, the handle of 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 56) 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() 48 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.

3.2.8.22 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 [15] is running. You must have called ps3000_run_streaming_ns() [41] beforehand to set up fast streaming.

Applicability	Fast streaming [15] mode only
Arguments	`handle`, the handle of the required device. `lpGetOverviewBuffersMaxMin`, a pointer to the callback function [43] 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(). [41] FALSE if one of the parameters is out of range.

3.2.8.23 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 [15] mode. You pass a pointer to this function to ps3000_get_streaming_last_values() [42], 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() [42].

For an example of a suitable callback function, see the C++ sample code [52] included in your PicoScope installation.

Applicability	Fast streaming [15] mode only
Arguments	`overviewBuffers`, a pointer to a location where ps3000_get_streaming_last_values() [42] will store a pointer to its overview buffers [57] that contain the sampled data. The driver creates the overview buffers when you call ps3000_run_streaming_ns() [41] to start fast streaming. `overflow`, a bit field that indicates whether there has been a voltage overflow on any channel. The bit assignments are as follows: 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 D C B A D C B A common-mode overflow differential overflow `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.

3.2.8.24 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. 15 It allows you to retrieve data with different aggregation 56 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(), 38 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 16 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 15 mode only

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.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				D	C	B	A					D	C	B	A
				common-mode overflow								differential overflow			

`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 56 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

3.2.8.25 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 [15] has stopped.

Before calling the function, capture some data using fast streaming, stop streaming using ps3000_stop(), [38] 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 [16] for more details of data values.)

Applicability	Fast streaming [15] mode only
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. 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 D C B A D C B A common-mode overflow / differential overflow `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

3.2.8.26 ps3000_save_streaming_data

```
short ps3000_save_streaming_data (
  short                   handle,
  PS3000_CALLBACK_FUNC    lpCallbackFunc,
  short                 * dataBuffers,
  short                   dataBufferSize )
```

This function sends all available streaming data to the my_save_streaming_data() 49 callback function in your application. Your callback function decides what to do with the data.

Applicability	Fast streaming 15 mode only
Arguments	`handle,` the handle of the required device. `lpCallbackFunc,` a pointer to the my_save_streaming_data() 49 callback function in your application that handles the saving of streaming data. `dataBuffers,` a pointer to the data. `dataBufferSize,` the size of the buffer, in samples.
Returns	

3.2.8.27 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() 41 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() 41.

Applicability	Fast streaming 15 mode only
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

3.2.8.28 Callback function to save data

```
short my_save_streaming_data (
  short * dataBuffer,
  short   noOfBuffers )
```

This is a callback function in your application that receives data from ps3000_save_streaming_data() 47.

The function name my_save_streaming_data() 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; it refers to it only by the address that you pass to ps3000_save_streaming_data() 42.

Applicability	Fast streaming 15 mode only
Arguments	`dataBuffer,` a pointer to the buffer where the values are stored. `noOfBuffers,` tells your function how many buffers there are.
Returns	

3.2.8.29 ps3000_close_unit

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

Shuts down a PicoScope 3425 oscilloscope.

Applicability	All modes
Arguments	`handle`, the handle, returned by ps3000_open_unit() 18, of the oscilloscope being closed.
Returns	`1` if a valid handle is passed, `0` if not.

3.3 Programming examples

Programming examples are optionally available when you install PicoScope 5. Select the **Custom** installation option and then enable **Programming Examples**, selecting either the whole tree or just the examples you are interested in. There are examples for the following languages and development environments:

3.3.1 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 file must be in the compilation directory:

- `ps3000.h`

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

- `ps3000.dll`

3.3.2 C++

The C++ example program shows how to use the fast streaming mode [15] in the driver, with and without triggering [12], and demonstrates the auto_stop feature. It runs in console mode.

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

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

You will also need the following library for Microsoft C++:

- `ps3000.lib` (Microsoft Visual C 32-bit applications)

Ensure that the program directory contains a copy of:

- `ps3000.dll`

from the PicoScope installation directory.

A Visual Studio 2005 (VC8) project file, `faststreaming.vcproj`, is provided.

3.3.3 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.

3.3.4 Delphi

The program:

- ps3000.dpr

in the Examples\ps3000\ subdirectory of your PicoScope installation demonstrates how to operate PicoScope 3000 Series PC Oscilloscopes. 57 The file:

- ps3000.inc

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

- ps3000fm.res
- ps3000fm.dfm
- ps3000fm.pas

This has been tested with Delphi version 3.

3.3.5 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.

3.3.6 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.

3.3.7 LabView

The 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_fastStream.vi
- ps3000_runBlock.vi
- ps3000_runStream.vi
- ps3000wrap.c
- ps3000wrap.dll

You will also need this file from the installation directory:

- PS3000.dll

4 Troubleshooting

4.1 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 55 section.

Error code	Meaning
1	More than 4 PicoScope 3000 Series oscilloscopes are opened on one machine. It is not possible to use more than 4 oscilloscopes in the same application.
2	The driver cannot allocate enough of the computer's memory to operate the oscilloscope. Consult the system requirements 1 section for more information.
3	A PicoScope 3000 Series PC Oscilloscope 57 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 3425 PC Oscilloscope. Consult the system requirements 1 section for more information.

4.2 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 54 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 XP SP2 or Vista.

5 Glossary

AC/DC control. Each channel can be set to either AC coupling or DC coupling. With DC coupling, the voltage displayed on the screen is equal to the true voltage of the signal across the differential inputs. With AC coupling, any DC component of the signal is filtered out, leaving only the variations in the signal (the AC component).

Aggregated values. In fast streaming mode 15, the PicoScope 3000 57 driver can use a method called **aggregation** to reduce the amount of data your application needs to process. This means that it replaces a number of consecutive raw samples with a pair of values, which are the minimum and maximum values of all the raw samples. You can set the aggregation parameter when you call ps3000_run_streaming_ns 41 for real-time capture, and when you call ps3000_get_streaming_values 44 to obtain post-processed data.

Aliasing. An effect that can cause digital oscilloscopes to display fast-moving waveforms incorrectly, by showing spurious low-frequency signals ("aliases") that do not exist in the input. To avoid this problem, choose a sampling rate that is at least twice the frequency of the fastest-changing input signal.

Analogue bandwidth. All oscilloscopes have an upper limit to the range of frequencies at which they can measure accurately. The analog bandwidth of an oscilloscope is defined as the frequency at which a displayed sine wave has half the power of the input sine wave (or, equivalently, about 71% of the amplitude).

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. This mode of operation is effective when the input signal being sampled is high frequency. Note: To avoid aliasing 56 effects, the maximum input frequency must be less than half the sampling rate.

Buffer size. The size, in samples, of the oscilloscope buffer memory. The buffer memory is used by the oscilloscope to temporarily store data before transferring it to the PC.

Common-mode overflow. The scope measures the difference between the positive and negative input voltages on each channel. The voltage of each input with respect to ground does not affect the measurement as long as it does not exceed the common-mode voltage limit. If this limit is exceeded, a common-mode overflow occurs and the scope will not measure the signal accurately. PicoScope 6 shows a warning indicator 11 when this happens.

Common-mode voltage. The common-mode voltage of two points is the average voltage of the two points with respect to ground. A differential oscilloscope accurately measures the voltage difference between its two inputs and ignores their common-mode voltage, as long as the input voltages with respect to ground remain within a defined range. Outside this range the accuracy of the measurement cannot be guaranteed.

Differential oscilloscope. A differential oscilloscope measures the voltage difference between two points, regardless of the voltage of either point with respect to ground. This is unlike a conventional oscilloscope, which requires one of the two points to be at ground potential. More details. 10

Differential overflow. Occurs when the difference between the positive and negative inputs on one channel exceeds the selected measuring range. The result is an inaccurate measurement. PicoScope 6 shows a warning indicator 11 when this happens.

Differential voltage limit. The differential voltage (the voltage difference between the positive and negative inputs on one channel) must not exceed this limit, or the oscilloscope may be permanently damaged.

Maximum sampling rate. A figure indicating the maximum number of samples the oscilloscope is capable of acquiring per second. Maximum sample rates are given in MS/s (megasamples per second). The higher the sampling capability of the oscilloscope, the more accurate the representation of the high frequencies in a fast signal.

Overview buffer. A buffer in which the PicoScope 3000 Series driver temporarily stores data on its way from the scope device to the application's buffer.

PC Oscilloscope. A measuring instrument consisting of a Pico Technology scope device and the PicoScope software. It provides all the functions of a bench-top oscilloscope without the cost of a display, hard disk, network adapter and other components that your PC already has.

PicoScope 3000 Series. A PC Oscilloscope range comprising the PicoScope 3204, 3205, 3206 general-purpose scopes, the PicoScope 3223 and 3423 automotive scopes, the PicoScope 3224 and 3424 high-resolution scopes and the 3425 differential scope.

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 on an oscilloscope 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 down 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.

Spectrum analyser. An instrument that measures the energy content of a signal in each of a large number of frequency bands. It displays the result as a graph of energy (on the vertical axis) against frequency (on the horizontal axis). The PicoScope software includes a spectrum analyser.

Streaming mode. A sampling mode in which the oscilloscope samples data and returns it to the computer in an unbroken stream. This mode of operation is effective when the input signal being sampled contains only low frequencies.

Timebase. The timebase controls the time interval across the scope display. There are ten divisions across the screen and the timebase is specified in units of time per division, so the total time interval is ten times the timebase.

USB 1.1. USB (Universal Serial Bus) 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 Mbps (12 megabits per second), and is much faster than a serial port.

USB 2.0. USB (Universal Serial Bus) is a standard port that enables you to connect external devices to PCs. A typical USB 2.0 port supports a data transfer rate that is 40 times faster than that supported by USB 1.1. USB 2.0 is backwards-compatible with USB 1.1.

Vertical resolution. A value, in bits, indicating the degree of precision with which the oscilloscope can turn input voltages into digital values. Calculation techniques can improve the effective resolution.

Voltage range. The voltage range is the difference between the maximum and minimum voltages that can be accurately captured by the oscilloscope.

Index

A

B

C

D

E

F

G

H

I

L

M

N

O

P

R

S

T

U

V

W

Pico Technology Ltd

James House
Colmworth Business Park
Eaton Socon
St. Neots
PE19 8YG
United Kingdom
Tel: +44 (0) 1480 396 395
Fax: +44 (0) 1480 396 296
Web: www.picotech.com

ps3425.en-2

7.12.07