# Contents

1 Introduction ......................................................................................................................... 1  
   1 Overview ....................................................................................................................... 1  
   2 Software license conditions ......................................................................................... 2  
   3 Trademarks ................................................................................................................... 2  

2 Getting started .................................................................................................................. 3  
   1 About the driver .............................................................................................................. 3  
   2 Installing the driver ....................................................................................................... 3  
   3 Connecting the logger .................................................................................................... 3  
   4 USB ADC-11 compatibility mode .................................................................................. 3  

3 Technical reference ............................................................................................................ 5  
   1 Capture modes ............................................................................................................... 5  
   2 Scaling ................................................................................................................................ 5  

4 Driver routines .................................................................................................................... 6  
   1 Summary ......................................................................................................................... 6  
   2 pl1000CloseUnit() - close the unit ............................................................................... 7  
   3 pl1000GetSingle() - get a single value from a specified channel ............................... 8  
   4 pl1000GetUnitInfo() - return information about the unit ........................................... 9  
   5 pl1000GetValues() - get a number of sample values after a run ............................... 10  
   6 pl1000MaxValue() - return the maximum ADC value .................................................. 11  
   7 pl1000OpenUnit() - open and enumerate the unit ......................................................... 12  
   8 pl1000OpenUnitAsync() - open the unit without waiting for completion .................. 13  
   9 pl1000OpenUnitProgress() - report progress of pl1000OpenUnitAsync() .................. 14  
  10 pl1000PingUnit() - check that the unit is responding ................................................... 15  
  11 pl1000Ready() - indicate when pl1000Run() has captured data .................................. 16  
  12 pl1000Run() - tell the unit to start capturing data .......................................................... 17  
  13 pl1000SetDo() - control the digital outputs on the unit ............................................... 18  
  14 pl1000SetInterval() - set the sampling speed of the unit ........................................... 19  
  15 pl1000SetPulseWidth() - configure the PWM output ................................................ 20  
  16 pl1000SetTrigger() - set the trigger on the unit ........................................................... 21  
  17 pl1000Stop() - abort data collection ............................................................................. 22  
  18 PICO_STATUS values ....................................................................................................... 23  

5 Glossary ............................................................................................................................ 25  

Index ..................................................................................................................................... 27
1 Introduction

1.1 Overview

The PicoLog 1000 Series PC Data Loggers are medium-speed, multichannel voltage-input devices for sampling analog data using a PC. This manual explains how to use the Application Programming Interface to write your own programs to control the unit. You should read it in conjunction with the PicoLog 1000 Series User’s Guide.

The following PicoLog 1000 Series Data Loggers are available:

<table>
<thead>
<tr>
<th>Version</th>
<th>Part No.</th>
<th>Resolution</th>
<th>Channels</th>
</tr>
</thead>
<tbody>
<tr>
<td>PicoLog 1012</td>
<td>PP543</td>
<td>10 bits</td>
<td>12</td>
</tr>
<tr>
<td>PicoLog 1216</td>
<td>PP544</td>
<td>12 bits</td>
<td>16</td>
</tr>
</tbody>
</table>

These devices can also be used with the PicoLog data logging software and the PicoScope oscilloscope software.
1.2 Software license conditions

The material contained in this release is licensed, not sold. Pico Technology Limited grants a license 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 Ltd. claims the copyright of, and retains the rights to, all material (software, documents, etc.) contained in this SDK except the example programs. You may copy and distribute the SDK without restriction, as long as you do not remove any Pico Technology copyright statements. The example programs in the SDK may be modified, copied and distributed for the purpose of developing programs to collect data using Pico products.

**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 license 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 14 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.

1.3 Trademarks

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

**PicoScope** and **Pico Technology** are registered in the U.S. Patent and Trademark Office.

**Windows** is a registered trademark of Microsoft Corporation in the USA and other countries.
2 Getting started

2.1 About the driver

The PicoLog 1000 Series units are supplied with a kernel driver and a DLL containing routines that you can call from your own programs. The drivers are supported by the following operating systems:

- Windows 7
- Windows 8
- Windows 10

The PicoSDK containing the drivers is available in 32-bit and 64-bit versions from www.picotech.com/downloads. The 32-bit driver will run on a 64-bit Windows system if you write a 32-bit application and run it under WoW64.

The DLL can be used with any programming language or application that can interface with DLLs: for example, C, Visual Basic for Applications (VBA) and LabVIEW. Example code is available in numerous repositories under the "picotech" organisation on GitHub. Some of these examples are fairly simple, but the C console mode example, pl1000con.c, demonstrates most of the facilities available in the driver.

The driver supports up to 64 USB units at one time.

2.2 Installing the driver

The driver is included in PicoSDK, which you can download from www.picotech.com/downloads. Select PicoLog Data Loggers > PicoLog 1012 or PicoLog 1216 > Software > PicoSDK.

2.3 Connecting the logger

Before you connect your logger, please install the driver software.

To connect the data logger, plug the cable provided into any available USB port on your PC. The first time you connect the unit, Windows may display a New Hardware Wizard. Follow any instructions in the Wizard and wait for the driver to be installed. Later versions of Windows display an Installing new hardware message and complete the process automatically. The unit is then ready for use.

2.4 USB ADC-11 compatibility mode

The PicoLog 1000 Series data loggers may be used as replacements for the USB ADC-11, an 11-channel data logger previously available from Pico Technology. The 1000 Series units have all the functions of the USB ADC-11 and some extra functions such as extra digital outputs, a PWM output and a sensor power output.

The 1000 Series units are API-compatible with the USB ADC-11. This means that any programs that you have already written do not need to be changed or recompiled - you simply need to update the usbadc11.dll to the latest version supplied in PicoSDK. The 1000 Series unit will behave like a USB ADC-11 and the extra outputs (pins 15 to 25) will be internally disconnected. You can continue to use the unit with an old ADC-11 terminal board if you have one, or you can switch to the new Small Terminal Board (PP545).
If you wish to use the extra functions of the 1000 Series units, you must rewrite your application to use the new PicoLog 1000 Series DLL (pl1000.dll), which is described in this manual and is available free of charge from Pico Technology. Example code is available to help you make the transition.
3 Technical reference

3.1 Capture modes

Three modes are available for capturing data:

- **BM_SINGLE**: collect a single block of data and exit
- **BM_WINDOW**: collect a series of overlapping blocks of data
- **BM_STREAM**: collect a continuous stream of data

**BM_SINGLE** is useful when you wish to collect data at high speed for a short period: for example, to collect 1000 readings in 50 milliseconds. The maximum capture size in this mode is 1 million samples.

**BM_WINDOW** is useful when collecting several blocks of data at low speeds - for example when collecting 10,000 samples over 10 seconds. Collecting a sequence of single blocks like this would take 10 seconds for each block, so displayed data would not be updated frequently. Using windowing, it is possible to ask for a new block more frequently, for example every second, and to receive a block containing 9 seconds of repeat data and 1 second of new data. The block is effectively a 10-second window that advances one second per cycle.

**BM_STREAM** is useful when you need to collect data continuously for long periods. In principle, it could be used to collect data indefinitely. Every time **pl1000GetValues()** is called, it returns the new readings since the last time it was called. The **noOfValues** argument passed to **pl1000Run()** must be sufficient to ensure that the buffer does not overflow between successive calls to **pl1000GetValues()**. For example, if you call **pl1000GetValues()** every second and you are collecting 500 samples per second, **noOfValues** must be at least 500, or preferably 1000, to allow for delays in the operating system.

3.2 Scaling

The PicoLog 1000 Series devices produce values in the range 0 to **maxValue**, where **maxValue** is the value returned by **pl1000MaxValue()**. To convert ADC readings to volts, multiply by 2.5 and divide by **maxValue**.

For example, **maxValue** for the PicoLog 1216 is 4095. Therefore, an ADC reading of 132 from this device from a represents $132 \times 2.5 / 4095 = \text{approx. 0.0806 volts.}$
4 Driver routines

4.1 Summary

The driver routines in the PicoLog 1000 Series API are listed, with short descriptions, in the Table of Contents at the start of this manual.

The driver allows you to do the following:

- Identify and open the logger
- Take a single reading from a particular channel
- Collect a block of samples at fixed time intervals from one or more channels
- Set up a trigger event for a particular channel

You can specify a sampling interval from 1 microsecond to 1 second. The shortest interval that the driver will accept depends on the capture mode selected.

The normal calling sequence to collect a block of data is as follows:

Check that the driver version is correct
Open the driver
Set trigger mode (if required)
Set sampling mode (channels and time per sample)

While you want to take measurements,
    Run
    While not ready
        Wait
    End while
    ... Get a block of data ...
End While

Close the driver (this happens automatically when the application terminates)
4.2 pl1000CloseUnit() - close the unit

```c
PICO_STATUS pl1000CloseUnit
(
    int16_t     handle
)
```

This function closes the unit.

**Arguments:**
- `handle`, device identifier returned by `pl1000OpenUnit()` or `pl1000OpenUnitProgress()`

**Returns:**
- PICO_OK
- PICO_HANDLE_INVALID
4.3 pl1000GetSingle() - get a single value from a specified channel

PICO_STATUS pl1000GetSingle
(
    int16_t    handle,
    PL1000_INPUTS channel,
    uint16_t   * value
)

This function returns a single sample value from the specified input channel.

Arguments:
handle, device identifier returned by pl1000OpenUnit() or pl1000OpenUnitProgress()

channel, which channel to sample:
[PL1000_CHANNEL_1 to PL1000_CHANNEL_12] (PicoLog 1012)
[PL1000_CHANNEL_1 to PL1000_CHANNEL_16] (PicoLog 1216)

value, a location where the function will write the sample value

Returns:
PICO_OK
PICO_INVALID_HANDLE
PICO_NO_SAMPLES_AVAILABLE
PICO_DEVICE_SAMPLING
PICO_NULL_PARAMETER
PICO_INVALID_PARAMETER
PICO_DATA_NOT_AVAILABLE
PICO_INVALID_CALL
PICO_NOT_RESPONDING
PICO_MEMORY
4.4 pl1000GetUnitInfo() - return information about the unit

```
PICO_STATUS pl1000GetUnitInfo
(
    int16_t handle,
    int8_t * string,
    int16_t stringLength,
    int16_t * requiredSize,
    PICO_INFO info
)
```

This function returns a string containing the specified item of information about the unit.

If you want to find out the length of the string before allocating a buffer for it, call the function with `string = NULL` first.

**Arguments:**
- `handle`, device identifier returned by `pl1000OpenUnit()` or `pl1000OpenUnitProgress()`
- `string`, a location where the function writes the requested information, or `NULL` if you are only interested in the value of `requiredSize`
- `stringLength`, the maximum number of characters that the function should write to `string`
- `requiredSize`, a location where the function writes the length of the information string before it was truncated to `stringLength`. If the string was not truncated, `requiredSize` will be less than or equal to `stringLength`
- `info`, the information that the driver should return. These values are specified in `PicoStatus.h`:
  - `PICO_DRIVER_VERSION`
  - `PICO_USB_VERSION`
  - `PICO_HARDWARE_VERSION`
  - `PICO_VARIANT_INFO`
  - `PICO BATCH AND SERIAL`
  - `PICO_CAL_DATE`
  - `PICO_KERNEL_DRIVER_VERSION`
  - `PICO_FIRMWARE_VERSION_1`

**Returns:**
- `PICO_OK`
- `PICO INVALID_HANDLE`
- `PICO NULL_PARAMETER`
- `PICO INVALID_INFO`
- `PICO_INFO UNAVAILABLE`
4.5  pl1000GetValues() - get a number of sample values after a run

PICO_STATUS pl1000GetValues

(  
    int16_t handle,  
    uint16_t * values,  
    uint32_t * noOfValues,  
    uint16_t * overflow,  
    uint32_t * triggerIndex
)

This function is used to get values after calling pl1000Run().

Arguments:

handle,  device identifier returned by pl1000OpenUnit() or pl1000OpenUnitProgress()

values,  an array of sample values returned by the function. The size of this buffer must be the number of enabled channels multiplied by the number of samples to be collected.

noOfValues,  on entry, the number of sample values per channel that the function should collect. On exit, the number of samples per channel that were actually written to the buffer.

overflow,  on exit, a bit field indicating which, if any, input channels overflowed the input range of the device. A bit set to 1 indicates an overflow. The least significant bit corresponds to channel 1. May be NULL if an overflow warning is not required.

triggerIndex,  on exit, a number indicating when the trigger event occurred. The number is a zero-based index to the values array, or 0xffffffff if the information is not available. On entry, the pointer may be NULL if a trigger index is not required.

Returns:
PICO_OK  
PICO_INVALID_HANDLE  
PICO_NO_SAMPLES_AVAILABLE  
PICO_DEVICE_SAMPLING  
PICO_NULL_PARAMETER  
PICO_INVALID_PARAMETER  
PICO_TOO_MANY_SAMPLES  
PICO_DATA_NOT_AVAILABLE  
PICO_INVALID_CALL  
PICO_NOT_RESPONDING  
PICO_MEMORY
4.6 pl1000MaxValue() - return the maximum ADC value

PICO_STATUS pl1000MaxValue
    (int16_t  handle,
     uint16_t * maxValue
    )

This function returns the maximum ADC value that the device will return. This value may be different on different models in the PicoLog 1000 Series.

**Arguments:**
handle, device identifier returned by [pl1000OpenUnit()](#) or [pl1000OpenUnitProgress()](#)

maxValue, a location where the function will write the maximum ADC value

**Returns:**
PICO_OK
PICO_INVALID_HANDLE
PICO_NULL_PARAMETER
PICO_INVALID_PARAMETER
4.7 pl1000OpenUnit() - open and enumerate the unit

```c
PICO_STATUS pl1000OpenUnit
(int16_t * handle)
```

This function opens and enumerates the unit.

**Arguments:**
- `handle`, the function will write a value here that uniquely identifies the data logger that was opened. Use this as the `handle` parameter when calling any other PicoLog 1000 Series API function.

**Returns:**
- `PICO_OK`
- `PICO_OS_NOT_SUPPORTED`
- `PICO_OPEN_OPERATION_IN_PROGRESS`
- `PICO_EEPROM_CORRUPT`
- `PICO_KERNEL_DRIVER_TOO_OLD`
- `PICO_FW_FAIL`
- `PICO_MAX_UNITS_OPENED`
- `PICO_NOT_FOUND`
- `PICO_NOT_RESPONDING`
4.8 pl1000OpenUnitAsync() - open the unit without waiting for completion

```c
PICO_STATUS pl1000OpenUnitAsync
(
    int16_t * status
)
```

This function opens a PicoLog 1000 Series data logger without waiting for the operation to finish. You can find out when it has finished by periodically calling `pl1000OpenUnitProgress()` until that function returns a non-zero value and a valid data logger handle.

The driver can support up to 64 data loggers.

**Arguments:**

- `status`, a location where the function writes a status flag:
  - 0: if there is already an open operation in progress
  - 1: if the open operation is initiated

**Returns:**

- `PICO_OK`
- `PICO_OPEN_OPERATION_IN_PROGRESS`
- `PICO_OPERATION_FAILED`
4.9 pl1000OpenUnitProgress() - report progress of pl1000OpenUnitAsync()

PICO_STATUS pl1000OpenUnitProgress(
    int16_t * handle,
    int16_t * progress,
    int16_t * complete
)

This function checks on the progress of pl1000OpenUnitAsync().

Arguments:
handle, a pointer to where the function should store the device identifier of the opened data logger, if the operation was successful. Use this as the handle parameter when calling any other PicoLog 1000 Series API function.

- 0: if no unit is found or the unit fails to open
- <>0: handle of unit (valid only if function returns PICO_OK)

progress, a location where the function writes an estimate of the progress towards opening the data logger. The value is between 0 and 100.

complete, a location where the function will write a non-zero value if the operation has completed

Returns:
PICO_OK
PICO_NULL_PARAMETER
PICO_OPERATION_FAILED
4.10 pl1000PingUnit() - check that the unit is responding

PICO_STATUS pl1000PingUnit
(
    int16_t handle
)

This function can be used to check that the already opened device is still connected to the USB port and communication is successful.

Arguments:
handle, device identifier returned by pl1000OpenUnit() or pl1000OpenUnitProgress()

Returns:
PICO_OK
PICO_INVALID_HANDLE
PICO_DRIVE_FUNCTION
PICO_BUSY
PICO_NOT_RESPONDING
4.11 pl1000Ready() - indicate when pl1000Run() has captured data

PICO_STATUS pl1000Ready
(
    int16_t handle,
    int16_t * ready
)

This function indicates when \texttt{pl1000Run()} has captured the requested number of samples.

\textbf{Arguments:}
handle, device identifier returned by \texttt{pl1000OpenUnit()} or \texttt{pl1000OpenUnitProgress()}

ready, TRUE if ready, FALSE otherwise

\textbf{Returns:}
PICO_OK
PICO_INVALID_HANDLE
PICO_NOT RESPONDING
4.12 pl1000Run() - tell the unit to start capturing data

```c
PICO_STATUS pl1000Run
(
    int16_t        handle,
    uint32_t       no_of_values,
    BLOCK_METHOD   method
)
```

This function tells the unit to start capturing data.

**Arguments:**
- `handle`, device identifier returned by `pl1000OpenUnit()` or `pl1000OpenUnitProgress()`
- `no_of_values`, the total number of samples to be collected per channel
- `method`, which method to use to collect the data, from the following list:
  - BM_SINGLE
  - BM_WINDOW
  - BM_STREAM

  See [Capture modes](#) for details.

**Returns:**
- PICO_OK
- PICO_INVALID_HANDLE
- PICO_USER_CALLBACK
- PICO_INVALID_CHANNEL
- PICO_TOO_MANY_SAMPLES
- PICO_INVALID_TIMEBASE
- PICO_NOT_RESPONDING
- PICO_CONFIG_FAIL
- PICO_INVALID_PARAMETER
- PICO_NOT_RESPONDING
- PICO_TRIGGER_ERROR
4.13 pl1000SetDo() - control the digital outputs on the unit

PICO_STATUS pl1000SetDo
(
    int16_t handle,
    int16_t do_value,
    int16_t doNo
)

This function controls the digital outputs DO0 to DO3 on the unit.

**Arguments:**
handle, device identifier returned by pl1000OpenUnit() or pl1000OpenUnitProgress()

do_value, whether to switch the output on or off:
  1 - turns the digital output on
  0 - turns the digital output off

doNo, which output to switch:
  [PL1000_DO_CHANNEL_0 to PL1000_DO_CHANNEL_3]

**Returns:**
PICO_OK
PICO_INVALID_HANDLE
PICO_NOT_RESPONDING
4.14 pl1000SetInterval() - set the sampling speed of the unit

PICO_STATUS pl1000SetInterval
(
    int16_t handle,
    uint32_t * us_for_block,
    uint32_t ideal_no_of_samples,
    int16_t * channels,
    int16_t no_of_channels
)

This function sets the sampling rate of the unit. Call this function with \texttt{us\_for\_block} set to the number of microseconds in which you wish to capture the entire requested data set. The function will return the actual number of microseconds the operation will take. You can then calculate the sampling interval $i$ as follows:

- in \texttt{BM\_SINGLE} mode*:
  
  \[
  i = \frac{1 \mu s \times \text{us\_for\_block}}{(\text{ideal\_no\_of\_samples} \times \text{no\_of\_channels})}
  \]

- in other modes:
  
  \[
  i = \frac{10 \mu s \times \text{us\_for\_block}}{(\text{ideal\_no\_of\_samples} \times \text{no\_of\_channels})}
  \]

*\texttt{BM\_SINGLE} mode can achieve sampling intervals down to 1 µs when \text{ideal\_no\_of\_samples} \times \text{no\_of\_channels} is no more than 8192. Under all other conditions, the fastest possible sampling interval is 10 µs per channel and \text{ideal\_no\_of\_samples} \times \text{no\_of\_channels} may be anything up to 1000000.

**Arguments:**

- \texttt{handle}, device identifier returned by \texttt{pl1000OpenUnit()} or \texttt{pl1000OpenUnitProgress()}

- \texttt{us\_for\_block}, on entry: the target total time in which to collect (\text{ideal\_no\_of\_samples} \times \text{no\_of\_channels}) samples, in microseconds; on exit: the time the driver will actually take to achieve this.

- \texttt{ideal\_no\_of\_samples}, the number of samples that you want to collect per channel. This number is used only for timing calculations.

- \texttt{channels}, an array of numbers identifying the channels from which you wish to capture:
  
  - [PL1000\_CHANNEL\_1 to PL1000\_CHANNEL\_12] (PicoLog 1012)
  - [PL1000\_CHANNEL\_1 to PL1000\_CHANNEL\_16] (PicoLog 1216)

  Sampling of multiple channels is sequential.

- \texttt{no\_of\_channels}, the number of channels in the \texttt{channels} array

**Returns:**

- \texttt{PICO\_OK}
- \texttt{PICO\_INVALID\_HANDLE}
- \texttt{PICO\_INVALID\_CHANNEL}
- \texttt{PICO\_INVALID\_TIMEBASE}
- \texttt{PICO\_NOT\_RESPONDING}
- \texttt{PICO\_CONFIG\_FAIL}
- \texttt{PICO\_INVALID\_PARAMETER}
- \texttt{PICO\_NOT\_RESPONDING}
- \texttt{PICO\_TRIGGER\_ERROR}
4.15  pl1000SetPulseWidth() - configure the PWM output

PICO_STATUS pl1000SetPulseWidth
(
    int16_t handle,
    uint16_t period,
    uint8_t cycle
)

This function sets the pulse width of the PWM (pulse-width modulated) output.

Arguments:
handle, device identifier returned by pl1000OpenUnit() or pl1000OpenUnitProgress()

period, the required period of the PWM waveform in microseconds, from 100 to 1800

cycle, the required duty cycle as a percentage from 0 to 100

Returns:
PICO_OK
PICO_INVALID_HANDLE
PICO_SIG_GEN_PARAM
PICO_NOT_RESPONDING
4.16 pl1000SetTrigger() - set the trigger on the unit

```c
PICO_STATUS pl1000SetTrigger
(
    int16_t handle,
    uint16_t enabled,
    uint16_t auto_trigger,
    uint16_t auto_ms,
    uint16_t channel,
    uint16_t dir,
    uint16_t threshold,
    uint16_t hysteresis,
    float delay
)
```

This function sets up the trigger, which controls when the unit starts capturing data.

**Arguments:**
- `handle`, device identifier returned by `pl1000OpenUnit()` or `pl1000OpenUnitProgress()`
- `enabled`, whether to enable or disable the trigger:
  - 0: disable the trigger
  - 1: enable the trigger
- `auto_trigger`, whether to rearm the trigger automatically after each trigger event:
  - 0: do not auto-trigger
  - 1: auto-trigger
- `auto_ms`, time in milliseconds after which the unit will auto-trigger if the trigger condition is not met
- `channel`, which channel to trigger on:
  - `[PL1000_CHANNEL_1 to PL1000_CHANNEL_12]` (PicoLog 1012)
  - `[PL1000_CHANNEL_1 to PL1000_CHANNEL_16]` (PicoLog 1216)
- `dir`, which edge to trigger on:
  - 0: rising edge
  - 1: falling edge
- `threshold`, trigger threshold (the level at which the trigger will activate) in ADC counts
- `hysteresis`, trigger hysteresis in ADC counts. This is the difference between the upper and lower thresholds. The signal must then pass through both thresholds in the same direction in order to activate the trigger, so that there are fewer unwanted trigger events caused by noise. The minimum value allowed is 1.
- `delay`, delay between the trigger event and the start of the block as a percentage of the block size. 0% means that the trigger event is the first data value in the block, and -50% means that the trigger event is in the middle of the block.

**Returns:**
- `PICO_OK`
- `PICO_INVALID_HANDLE`
- `PICO_USER_CALLBACK`
- `PICO_TRIGGER_ERROR`
- `PICO_MEMORY_FAIL`
4.17  pl1000Stop() - abort data collection

PICO_STATUS pl1000Stop
(
  int16_t   handle
)

This function aborts data collection. It is the normal method of terminating BM_WINDOW and BM_STREAM data collection. You can also call it to terminate a BM_SINGLE data collection early, but this will invalidate any data that has been captured.

Arguments:
handle,  device identifier returned by pl1000OpenUnit() or pl1000OpenUnitProgress()

Returns:
PICO_OK
PICO_INVALID_HANDLE
### 4.18 PICO_STATUS values

Every function in the PicoLog 1000 Series API returns an error code from the following list of PICO_STATUS values defined in PicoStatus.h:

<table>
<thead>
<tr>
<th>Code (hex)</th>
<th>Enum</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>PICO_OK</td>
<td>The Data Logger is functioning correctly</td>
</tr>
<tr>
<td>01</td>
<td>PICO_MAX_UNITS_OPENED</td>
<td>An attempt has been made to open more than 64 units</td>
</tr>
<tr>
<td>02</td>
<td>PICO_MEMORY_FAIL</td>
<td>Not enough memory could be allocated on the host machine</td>
</tr>
<tr>
<td>03</td>
<td>PICO_NOT_FOUND</td>
<td>No PicoLog 1000 device could be found</td>
</tr>
<tr>
<td>04</td>
<td>PICO_FW_FAIL</td>
<td>Unable to download firmware</td>
</tr>
<tr>
<td>05</td>
<td>PICO_OPEN_OPERATION_IN_PROGRESS</td>
<td>A request to open a device is in progress</td>
</tr>
<tr>
<td>06</td>
<td>PICO_OPERATION_FAILED</td>
<td>The operation was unsuccessful</td>
</tr>
<tr>
<td>07</td>
<td>PICO_NOT_RESPONDING</td>
<td>The device is not responding to commands from the PC</td>
</tr>
<tr>
<td>08</td>
<td>PICO_CONFIG_FAIL</td>
<td>The configuration information in the device has become corrupt or is missing</td>
</tr>
<tr>
<td>09</td>
<td>PICO_KERNEL_DRIVER_TOO_OLD</td>
<td>The picopp.sys file is too old to be used with the device driver</td>
</tr>
<tr>
<td>0A</td>
<td>PICO_EEPROM_CORRUPT</td>
<td>The EEPROM has become corrupt, so the device will use a default setting</td>
</tr>
<tr>
<td>0B</td>
<td>PICO_OS_NOT_SUPPORTED</td>
<td>The operating system on the PC is not supported by this driver</td>
</tr>
<tr>
<td>0C</td>
<td>PICO_INVALID_HANDLE</td>
<td>There is no device with the handle value passed</td>
</tr>
<tr>
<td>0D</td>
<td>PICO_INVALID_PARAMETER</td>
<td>A parameter value is not valid</td>
</tr>
<tr>
<td>0E</td>
<td>PICO_INVALID_TIMEBASE</td>
<td>The timebase is not supported or is invalid</td>
</tr>
<tr>
<td>0F</td>
<td>PICO_INVALID_VOLTAGE_RANGE</td>
<td>The voltage range is not supported or is invalid</td>
</tr>
<tr>
<td>10</td>
<td>PICO_INVALID_CHANNEL</td>
<td>The channel number is not valid on this device or no channels have been set</td>
</tr>
<tr>
<td>11</td>
<td>PICO_INVALID_TRIGGER_CHANNEL</td>
<td>The channel set for a trigger is not available on this device</td>
</tr>
<tr>
<td>12</td>
<td>PICO_INVALID_CONDITION_CHANNEL</td>
<td>The channel set for a condition is not available on this device</td>
</tr>
<tr>
<td>13</td>
<td>PICO_NO_SIGNAL_GENERATOR</td>
<td>The device does not have a signal generator</td>
</tr>
<tr>
<td>14</td>
<td>PICO_STREAMING_FAILED</td>
<td>Streaming has failed to start or has stopped without user request</td>
</tr>
<tr>
<td>15</td>
<td>PICO_BLOCK_MODE_FAILED</td>
<td>Block failed to start - a parameter may have been set wrongly</td>
</tr>
<tr>
<td>16</td>
<td>PICO_NULL_PARAMETER</td>
<td>A parameter that was required is NULL</td>
</tr>
<tr>
<td>18</td>
<td>PICO_DATA_NOT_AVAILABLE</td>
<td>No data is available from a run block call</td>
</tr>
<tr>
<td>19</td>
<td>PICO_STRING_BUFFER_TOO_SMALL</td>
<td>The buffer passed for the information was too small</td>
</tr>
<tr>
<td>1A</td>
<td>PICO_EETS_NOT_SUPPORTED</td>
<td>ETS is not supported on this device</td>
</tr>
<tr>
<td>1B</td>
<td>PICO_AUTO_TRIGGER_TIME_TOO_SHORT</td>
<td>The auto trigger time is less than the time it will take to collect the data</td>
</tr>
<tr>
<td>1C</td>
<td>PICO_BUFFER_STALL</td>
<td>The collection of data has stalled as unread data would be overwritten</td>
</tr>
<tr>
<td>1D</td>
<td>PICO_TOO_MANY_SAMPLES</td>
<td>The number of samples requested is more than available in the current memory segment</td>
</tr>
<tr>
<td>1E</td>
<td>PICO_TOO_MANY_SEGMENTS</td>
<td>Not possible to create number of segments requested</td>
</tr>
<tr>
<td>1F</td>
<td>PICO_PULSE_WIDTH_QUALIFIER</td>
<td>A null pointer has been passed in the trigger function or one of the parameters is out of range</td>
</tr>
<tr>
<td>20</td>
<td>PICO_DELAY</td>
<td>One or more of the hold-off parameters are out of range</td>
</tr>
<tr>
<td>21</td>
<td>PICO_SOURCE_DETAILS</td>
<td>One or more of the source details are incorrect</td>
</tr>
<tr>
<td>Code</td>
<td>Description</td>
<td>Additional Information</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>------------------------</td>
</tr>
<tr>
<td>22</td>
<td>PICO_CONDITIONS</td>
<td>One or more of the conditions are incorrect</td>
</tr>
<tr>
<td>24</td>
<td>PICO_DEVICE_SAMPLING</td>
<td>An attempt is being made to get stored data while streaming. Stop streaming by calling <code>pl1000Stop()</code></td>
</tr>
<tr>
<td>25</td>
<td>PICO_NO_SAMPLES_AVAILABLE</td>
<td>...because a run has not been completed</td>
</tr>
<tr>
<td>26</td>
<td>PICO_SEGMENT_OUT_OF_RANGE</td>
<td>The memory index is out of range</td>
</tr>
<tr>
<td>27</td>
<td>PICO_BUSY</td>
<td>Data cannot be returned yet</td>
</tr>
<tr>
<td>28</td>
<td>PICO_STARTINDEX_INVALID</td>
<td>The start time to get stored data is out of range</td>
</tr>
<tr>
<td>29</td>
<td>PICO_INVALID_INFO</td>
<td>The information number requested is not a valid number</td>
</tr>
<tr>
<td>2A</td>
<td>PICO_INFO_UNAVAILABLE</td>
<td>The handle is invalid so no information is available about the device. Only <code>PICO_DRIVER_VERSION</code> is available.</td>
</tr>
<tr>
<td>2B</td>
<td>PICO_INVALID_SAMPLE_INTERVAL</td>
<td>The sample interval selected for streaming is out of range</td>
</tr>
<tr>
<td>2C</td>
<td>PICO_TRIGGER_ERROR</td>
<td>Not used</td>
</tr>
<tr>
<td>2D</td>
<td>PICO_MEMORY</td>
<td>Driver cannot allocate memory</td>
</tr>
<tr>
<td>26</td>
<td>PICO_DELAY_NULL</td>
<td>NULL pointer passed as delay parameter</td>
</tr>
<tr>
<td>37</td>
<td>PICO_INVALID_BUFFER</td>
<td>The buffers for overview data have not been set while streaming</td>
</tr>
<tr>
<td>3A</td>
<td>PICO_CANCELED</td>
<td>A block collection has been canceled</td>
</tr>
<tr>
<td>3B</td>
<td>PICO_SEGMENT_NOT_USED</td>
<td>The segment index is not currently being used</td>
</tr>
<tr>
<td>3F</td>
<td>PICO_NOT_USED</td>
<td>The function is not available</td>
</tr>
<tr>
<td>41</td>
<td>PICO_INVALID_STATE</td>
<td>Device is in an invalid state</td>
</tr>
<tr>
<td>43</td>
<td>PICO_DRIVE_FUNCTION</td>
<td>You called a driver function while another driver function was still being processed</td>
</tr>
</tbody>
</table>
Glossary

**ADC.** Analog to Digital Converter. An ADC samples analog signals and converts them to digital data for storage and processing. It is an essential component of a data logger.

**DLL.** Dynamic Link Library. A file containing a collection of Windows functions designed to perform a specific class of operations. A DLL is supplied with the PicoLog Data Loggers to enable you to control the devices from your own programs.

**Driver.** A small program that acts as an interface, generally between a hardware component and a computer program. The PicoLog Data Loggers require a USB driver that runs in the Windows kernel, and a second driver in the form of a DLL that communicates with your application.

**Maximum sampling rate.** A figure indicating the maximum number of samples the ADC is capable of acquiring per second. Maximum sample rates are usually given in S/s (samples per second). The higher the sampling rate of the ADC, the more accurately it can represent the high-frequency details in a signal.

**Streaming.** An operating mode in which the ADC samples data and returns it to the computer in an unbroken stream.

**USB.** Universal Serial Bus. This is a standard port that enables you to connect external devices to PCs. A full-speed USB 2.0 port operates at up to 480 megabits per second. The PicoLog 1000 Series is also compatible with any USB port from USB 1.1 upwards.
Index

6
64-bit Windows  3

A
ADC value, maximum 11
ADC-11 compatibility 3
Asynchronous operation 5

B
BM_SINGLE mode 5
BM_STREAM mode 5
BM_WINDOW mode 5

C
Capture modes
  BM_SINGLE 5
  BM_STREAM 5
  BM_WINDOW 5
Closing a unit 7
Connecting to the PC 3

D
Data, reading 8, 10
Digital outputs, setting 18
DLLs 3
Driver routines
  pl1000CloseUnit 7
  pl1000GetSingle 8
  pl1000GetUnitInfo 9
  pl1000GetValues 10
  pl1000MaxValue 11
  pl1000OpenUnit 12
  pl1000OpenUnitAsync 13
  pl1000OpenUnitProgress 14
  pl1000Ready 16
  pl1000Run 17
  pl1000SetDo 18
  pl1000SetInterval 19
  pl1000SetPulseWidth 20
  pl1000SetTrigger 21
  pl1000Stop 22
  summary 6

G
Glossary 25

I
Information on unit, obtaining 9
Installation 3

L
Legal information 2

M
Maximum ADC value 11

N
New Hardware Wizard 3

O
Opening a unit 12, 13, 14, 16
Overview 1

P
PicoLog 1000 Series SDK 3
Programming 3
Pulse width, setting 20
PWM output, setting up 20

R
Running a unit 17

S
Sampling interval, setting 19
Scaling 5
SDK 3
Software license conditions 2
Stopping a unit 22
Streaming 5

T
Trademarks 2
Trigger, setting 21

U
Unit information, obtaining 9
USB ADC-11 compatibility  3

W

WoW64  3