PicoLog® 1000 Series

USB Data Loggers

Programmer's Guide

pl1000pg r7

pico®
Technology

Contents

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:

Version	Part No.	Resolution	Channels
PicoLog 1012	PP543	10 bits	12
PicoLog 1216	PP544	12 bits	16

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 x 2.5 / 4095 = 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

```
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

```
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

```
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 `pl1000Run()` has captured the requested number of samples.

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

`ready,` `TRUE` if ready, `FALSE` otherwise

Returns:
`PICO_OK`
`PICO_INVALID_HANDLE`
`PICO_NOT_RESPONDING`

4.12 pl1000Run() - tell the unit to start capturing data

```
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 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 BM_SINGLE mode*:
i = 1 µs x us_for_block / (ideal_no_of_samples x no_of_channels)
in other modes:
i = 10 µs x us_for_block / (ideal_no_of_samples x no_of_channels)

* BM_SINGLE mode can achieve sampling intervals down to 1 µs when ideal_no_of_samples x no_of_channels is no more than 8192. Under all other conditions, the fastest possible sampling interval is 10 µs per channel and ideal_no_of_samples x no_of_channels may be anything up to 1000000.

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

us_for_block, on entry: the target total time in which to collect (ideal_no_of_samples * no_of_channels) samples, in microseconds; on exit: the time the driver will actually take to achieve this.

ideal_no_of_samples, the number of samples that you want to collect per channel. This number is used only for timing calculations.

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.

no_of_channels, the number of channels in the channels array

Returns:
PICO_OK
PICO_INVALID_HANDLE
PICO_INVALID_CHANNEL
PICO_INVALID_TIMEBASE
PICO_NOT_RESPONDING
PICO_CONFIG_FAIL
PICO_INVALID_PARAMETER
PICO_NOT_RESPONDING
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

```
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`:

Code (hex)	Enum	Description
00	PICO_OK	The Data Logger is functioning correctly
01	PICO_MAX_UNITS_OPENED	An attempt has been made to open more than 64 units
02	PICO_MEMORY_FAIL	Not enough memory could be allocated on the host machine
03	PICO_NOT_FOUND	No PicoLog 1000 device could be found
04	PICO_FW_FAIL	Unable to download firmware
05	PICO_OPEN_OPERATION_IN_PROGRESS	A request to open a device is in progress
06	PICO_OPERATION_FAILED	The operation was unsuccessful
07	PICO_NOT_RESPONDING	The device is not responding to commands from the PC
08	PICO_CONFIG_FAIL	The configuration information in the device has become corrupt or is missing
09	PICO_KERNEL_DRIVER_TOO_OLD	The `picopp.sys` file is too old to be used with the device driver
0A	PICO_EEPROM_CORRUPT	The EEPROM has become corrupt, so the device will use a default setting
0B	PICO_OS_NOT_SUPPORTED	The operating system on the PC is not supported by this driver
0C	PICO_INVALID_HANDLE	There is no device with the handle value passed
0D	PICO_INVALID_PARAMETER	A parameter value is not valid
0E	PICO_INVALID_TIMEBASE	The timebase is not supported or is invalid
0F	PICO_INVALID_VOLTAGE_RANGE	The voltage range is not supported or is invalid
10	PICO_INVALID_CHANNEL	The channel number is not valid on this device or no channels have been set
11	PICO_INVALID_TRIGGER_CHANNEL	The channel set for a trigger is not available on this device
12	PICO_INVALID_CONDITION_CHANNEL	The channel set for a condition is not available on this device
13	PICO_NO_SIGNAL_GENERATOR	The device does not have a signal generator
14	PICO_STREAMING_FAILED	Streaming has failed to start or has stopped without user request
15	PICO_BLOCK_MODE_FAILED	Block failed to start - a parameter may have been set wrongly
16	PICO_NULL_PARAMETER	A parameter that was required is `NULL`
18	PICO_DATA_NOT_AVAILABLE	No data is available from a run block call
19	PICO_STRING_BUFFER_TOO_SMALL	The buffer passed for the information was too small
1A	PICO_ETS_NOT_SUPPORTED	ETS is not supported on this device
1B	PICO_AUTO_TRIGGER_TIME_TOO_SHORT	The auto trigger time is less than the time it will take to collect the data
1C	PICO_BUFFER_STALL	The collection of data has stalled as unread data would be overwritten
1D	PICO_TOO_MANY_SAMPLES	The number of samples requested is more than available in the current memory segment
1E	PICO_TOO_MANY_SEGMENTS	Not possible to create number of segments requested
1F	PICO_PULSE_WIDTH_QUALIFIER	A null pointer has been passed in the trigger function or one of the parameters is out of range
20	PICO_DELAY	One or more of the hold-off parameters are out of range
21	PICO_SOURCE_DETAILS	One or more of the source details are incorrect

22	PICO_CONDITIONS	One or more of the conditions are incorrect
24	PICO_DEVICE_SAMPLING	An attempt is being made to get stored data while streaming. Stop streaming by calling pl1000Stop().
25	PICO_NO_SAMPLES_AVAILABLE	...because a run has not been completed
26	PICO_SEGMENT_OUT_OF_RANGE	The memory index is out of range
27	PICO_BUSY	Data cannot be returned yet
28	PICO_STARTINDEX_INVALID	The start time to get stored data is out of range
29	PICO_INVALID_INFO	The information number requested is not a valid number
2A	PICO_INFO_UNAVAILABLE	The handle is invalid so no information is available about the device. Only PICO_DRIVER_VERSION is available.
2B	PICO_INVALID_SAMPLE_INTERVAL	The sample interval selected for streaming is out of range
2C	PICO_TRIGGER_ERROR	Not used
2D	PICO_MEMORY	Driver cannot allocate memory
36	PICO_DELAY_NULL	NULL pointer passed as delay parameter
37	PICO_INVALID_BUFFER	The buffers for overview data have not been set while streaming
3A	PICO_CANCELLED	A block collection has been canceled
3B	PICO_SEGMENT_NOT_USED	The segment index is not currently being used
3F	PICO_NOT_USED	The function is not available
41	PICO_INVALID_STATE	Device is in an invalid state
43	PICO_DRIVE_FUNCTION	You called a driver function while another driver function was still being processed

5 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

UK headquarters:

Pico Technology
James House
Colmworth Business Park
St. Neots
Cambridgeshire
PE19 8YP
United Kingdom

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

US regional office:

Pico Technology
320 N Glenwood Blvd
Tyler
Texas 75702
United States

Tel: +1 800 591 2796
Fax: +1 620 272 0981

Asia-Pacific regional office:

Pico Technology
Room 2252, 22/F, Centro
568 Hengfeng Road
Zhabei District
Shanghai 200070
PR China

Tel: +86 21 2226-5152

sales@picotech.com
support@picotech.com

pico.china@picotech.com

www.picotech.com

pl1000pg r7 2018-03-01