MATLAB® Instrument Driver

for PicoScope® 5000A/B Series

Programmer's Guide

ps5000ab.matlab.en r1


pico
Technology

Contents


pico
Technology

1 Introduction

This document outlines the functions defined in the MATLAB® Instrument Driver for the PicoScope® 5000A/B Series oscilloscopes.

Driver version: 1.1.16

1.1 License agreement

Grant of license. The material contained in this release is licensed, not sold. Pico Technology Limited ('Pico') 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 and agree to abide by these conditions.

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

Copyright. The software in this release is for use only with Pico products or with data collected using Pico products. 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 and its agents shall not be liable for any loss or damage, howsoever caused, related to the use of Pico equipment or software, unless excluded by statute.

Fitness for purpose. No two applications are the same, so Pico cannot guarantee that its equipment or software is suitable for a given application. It is therefore the user's responsibility to ensure that the product is suitable for the user's application.

Mission-critical applications. Because the software runs on a computer that may be running other software products, and may be subject to interference from these other products, this license specifically excludes usage in 'mission-critical' applications, for example life-support systems.

Viruses. This software was continuously monitored for viruses during production. However, the user is responsible for virus checking the software once it is installed.

Support. No software is ever error-free, but if you are dissatisfied with the performance of this software, please contact our technical support staff.

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.

1.2 Trademarks

Pico Technology and *PicoScope* are internationally registered trademarks of Pico Technology. *Pico Technology* is registered at the U.S. Patents and Trademarks Office. *MATLAB* is a registered trademark of The Mathworks, Inc. *Instrument Control Toolbox* is a trademark of The Mathworks, Inc. *Windows* is a trademark or registered trademark of Microsoft Corporation.

1.3 Company details

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

Address: Pico Technology
James House
Colmworth Business Park
ST NEOTS
Cambridgeshire
PE19 8YP
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 MATLAB® Driver Package

MATLAB is a numerical computing environment and high-level language developed by Mathworks that is used in industry and academia for a wide variety of applications.

This package is provided in two ways:

- A zip file (`PS5000a_MATLAB_IC_Generic_Driver.zip`) downloaded from the MathWorks File Exchange site:

  http://www.mathworks.com/matlabcentral/fileexchange/42820-picoscope%C2%AE-5000-series-matlab%C2%AE-generic-instrument-driver

- In the MATLAB folder in the root directory of the Software Development Kit (SDK) for the PicoScope 5000 Series (A API).

The package contains the following:

- a MATLAB Generic Instrument Driver that is used with Instrument Control Toolbox
- scripts that demonstrate how to call various functions in order to capture data in block and streaming mode, as well as using the signal generator

For MATLAB product information, please contact:

The MathWorks
3 Apple Hill Drive
Natick, MA, 01760-2098
USA

Tel: +1 508-647-7000
Fax: +1 508-647-7101

E-mail: info@mathworks.com

Web: www.mathworks.com

2.1 Prerequisites

The following hardware and software will be required:

Hardware:

One of the following devices:

- PicoScope 5000A or 5000B Series device

Software:

- MATLAB v8.0 (R2012b or later) – 32-bit/64-bit for Microsoft Windows
- Instrument Control Toolbox (v3.2 or later)
- Latest version of PicoScope PS5000A Software Development Kit (filename beginning `PS5000asdk`) – available to download from http://www.picotech.com/software.html

  Note: Be careful not to confuse this with the PS5000 SDK for the original PicoScope 5000 Series oscilloscopes (PicoScope 5203 and 5204) which has a filename beginning `PS5000sdk`).

- Microsoft Windows SDK 7.1 (for MATLAB 64-bit only) - please also refer to "How do I install Microsoft Windows SDK 7.1?" on MATLAB CENTRAL (http://www.mathworks.com/matlabcentral/answers/101105).
- Testing has been carried out with 32-bit and 64-bit versions of MATLAB R2012b.

2.2 Installation procedure

Extract the contents of the `PS5000a_MATLAB_IC_Generic_Driver.zip` file or copy the MATLAB directory from the SDK to a directory on the computer.

The following dynamic link library (DLL) files from the SDK will also be required:

- `ps5000a.dll` (Rename from `PS5000a.dll` to `ps5000a.dll` if necessary)
- `ps5000aWrap.dll`
- `PicoIpp.dll`

For 64-bit versions, please ensure that you obtain these DLL files from the x64 directories in the root and `Wrapper` directories in the SDK. The files from the `ps5000a/x64` directory in the Instrument Driver Package will also be required.

In the root directory are two folders:

- `Functions`
- `ps5000A`

In addition to the above two folders there are two script files in the root directory:

- `PicoConstants.m` – contains common constant values used across Pico Technology devices.
- `PicoStatus.m` – contains common status codes and information ID numbers used between various PicoScope ranges as well as the PicoLog 1000 devices.

The locations of the DLL files and the above folders (including their contents) must be added to the *MATLAB Path* (use the *addpath* command if necessary).

The folder `Functions` contains the following files:

- `adc2mv.m` – converts ADC counts to millivolts.
- `freq2delta.m` - Converts frequency to a delta value for PicoScope AWG.
- `importAWGFile.m` – Allows a PicoScope AWG comma-separated values (CSV) file to be imported
- `mv2adc.m` – converts millivolt values to ADC counts.
- `j2m.m` – converts a Java object to a MATLAB object.
- `normalise.m` - Normalise a vector (used for the AWG).

The `ps5000a` folder contains the generic instrument driver, prototype files, and a number of scripts:

Driver

- `picotech_ps5000a_generic.mdd`

Prototype Files

- `PS5000aMFile.m`
- `PS5000aWrapMFile.m`

The prototype files are used in place of the C header files in the SDK to define functions, enumerations and structures.

The prototype files to use with 64-bit MATLAB can be found in the x64 directory located in the `ps5000a` directory of the Instrument Driver package. Copy these files into the same directory as the Instrument Driver file or add them to the MATLAB path, ensuring that the prototype files generated using the 32-bit DLL are moved elsewhere.

Example Scripts

The files below provide examples of using the driver to capture data using different modes and simple triggering:

- `PS5000a_IC_Generic_Driver_Block.m`
- `PS5000a_IC_Generic_Driver_FFT.m`
- `PS5000a_IC_Generic_Driver_Rapid_Block.m`
- `PS5000a_IC_Generic_Driver_Rapid_Block_Plot3D.m`
- `PS5000a_IC_Generic_Driver_Streaming.m`
- `PS5000a_IC_Generic_Driver_Sig_Gen.m`

Some of the scripts are modified versions of scripts generated using the Instrument Control Toolbox Test and Measurement Tool.

2.3 Selecting C Compiler for MATLAB

Before using the Instrument Driver, you must select a C compiler.

1. At the MATLAB command prompt type:

   ```
   mex -setup
   ```

   and press Enter.

2. Press 'y' when prompted for mex to locate installed compilers.

3. Select the following compiler:

   MATLAB 32-bit: lcc-win32 C 2.4.1

   MATLAB 64-bit: Microsoft Software Development Kit (SDK) 7.1

4. Verify the choice selected.

2.4 Running the examples

The example scripts can be run either from the MATLAB command window or from the script Editor window.

To run a script, either:

- Type the script name in the MATLAB command window and press **ENTER**

or

- Open the script in the MATLAB Editor and click **Run**.

NOTE: The `m` files are structured in cells allowing you to run a small set of functions at a time.

3 MATLAB Generic Instrument Driver

The MATLAB Generic Instrument Driver for the PicoScope 5000A/B Series consists of a number of properties and functions which are listed below.

3.1 Properties

Beside the standard Instrument Control Device properties, the following additional properties are stored internally within the driver for each instance of an object:

Property Name	Data type	Default value	Can be changed by user
autoStop	Double - enumeration	1	Y
	A flag to specify if streaming should stop when all of maxSamples have been taken.		
awgBufferSize	Double	Device-dependent	N
	Size of the AWG waveform buffer ('B' models only).		
bandwidth	Double	Device/resolution-dependent	N
	Represents the maximum bandwidth (3 dB cut-off point) of the oscilloscope in hertz (Hz).		
bufferMemory	Double	Device-dependent	N
	Defines the size of the memory buffer on the device (in samples).		
channelCount	Double	Device-dependent	N
	The number of analog input channels on the device.		
channelSettings	Any	N/A	N
	Represents a set of arrays containing the channel settings for analog channels.		
dacFrequency	Double	Device-dependent (20e6)	N
	dacFrequency is the update frequency of the arbitrary waveform generator.		
digitalPorts	Double	N/A	N
	Not used		
digitalSettings	Any (struct)	N/A	N
	Not used		
firstRange	Double - enumeration	0	N
	Enumeration corresponding to the lowest voltage range supported by the device (10 mV).		
hasHardwareEts	Double - enumeration	Device-dependent	N
	Indicates if the device has hardware-based equivalent time sampling.		
lastRange	Double - enumeration	11	N
	Enumeration corresponding to highest voltage range supported by device (20 V).		
maxADCValue	Double	Resolution-dependent (32512)	N
	The maximum ADC count value for the device.		
maxSamplingRate	Double	Device-dependent	N
	The maximum single shot sampling rate for the device (samples per second).		
minADCValue	Double	Resolution-dependent (–32512)	N

	The minimum ADC count value for the device.		
numPostTriggerSamples	Double	1024	Y
	The number of samples to be taken after a trigger event.		
numPreTriggerSamples	Double	0	Y
	The number of samples to return before the trigger event.		
resolution	Double - enumeration	8	N
	The resolution of the device, in bits.		
sigGenType	Double - enumeration	Device-dependent	N
	The type of signal generator on the device: 0 - None 1 - Function generator 2 - Arbitrary waveform generator		
startFrequency	Double	1000	Y
	The frequency that the signal generator will initially produce (hertz).		
stopFrequency	Double	1000	Y
	The frequency at which the sweep reverses direction or returns to the initial frequency (hertz).		
streamingInterval	Double	1e–6	Y
	The requested time interval between samples (in seconds) when capturing data in streaming mode.		
timebase	Double	65	Y
	The timebase used for the scope when capturing data in block mode.		
unitHandle	Double	0 – handle assigned by the driver.	N
	The handle assigned by the underlying driver for the device.		
unitSerial	String	Device-dependent	N
	The device batch/serial number.		

Further information for a property where applicable, may be found using the `instrhelp` function. For example:

```
instrhelp(ps5000aDeviceObj, 'autoStop')
```

3.2 Functions

The PicoScope 5000 MATLAB Instrument Driver provides a number of functions to control the device from within the MATLAB environment. Most of the functions mirror the core API functions defined in the *PicoScope 5000 Series (A API) Programmer's Guide*.

3.2.1 'handle' parameter

Note that as Instrument Control Toolbox makes use of device objects to represent an instance of a device, the `handle` parameter defined in the core API and wrapper functions is not required as this is stored internally as the `unitHandle` property in the Instrument Driver.

Instead of the `handle` parameter, the argument `obj,` which represents the device object, is used.

3.2.2 Function return values

The `status` values returned from some function calls or through an error message correspond to `PICO_STATUS` values defined in the `PicoStatus.m` file. Use the `dec2hex()` MATLAB function to convert the value to a hexadecimal value, the definitions of which may be found in the main *PicoScope 5000 Series (A API) Programmer's Guide*.

In addition to the status, the output from a function call in MATLAB may consist of a number of parameters – this is due to the fact that MATLAB outputs variables corresponding to arguments that are pointers in the API function call.

For further information on functions in the Instrument Driver, please call the `instrhelp` function from the MATLAB command window e.g.

```
instrhelp(ps5000aDeviceObj, 'runBlock');
```

3.2.3 Function descriptions

3.2.3.1 Core API functions

The following core API functions are available within the Instrument Driver. Unless otherwise stated, the parameters match those defined in the main Programmer's Guide.

- `ps5000aChangePowerSource`
- `ps5000aCurrentPowerSource`
- `ps5000aFlashLed`
- `ps5000aGetAnalogueOffset`
  `maximumVoltage` and `minimumVoltage` input arguments not required.
- `ps5000aGetChannelInformation`
  `info, probe, ranges` and `length` input arguments are not required.
- `ps5000aGetDeviceResolution`
  `resolution` input argument not required, the resolution output argument is in bits.
- `ps5000aGetMaxDownSampleRatio`
  `maxDownSampleRatio` input argument not required.
- `ps5000aGetMaxSegments`
  `maxsegments` input argument not required.
- `ps5000aGetNoOfCaptures`
  `nCaptures` input argument not required.
- `ps5000aGetNoOfProcessedCaptures`
  `nProcessedCaptures` input argument not required.
- `ps5000aGetTimebase`
  `noSamples, timeIntervalNanoseconds` and `maxSamples` input arguments are not required.
- `ps5000aGetTimebase2`
  `noSamples, timeIntervalNanoseconds` and `maxSamples` input arguments are not required.
- `ps5000aGetTriggerTimeOffset`
  `timeUpper, timeLower` and `timeUnits` input arguments are not required.
- `ps5000aGetTriggerTimeOffset64`
  `time` and `timeUnits` input arguments are not required.
- `ps5000aGetValues`
  `overflow` input argument is not required.
- `ps5000aGetValuesBulk`
  `overflow` input argument is not required.
- `ps5000aGetValuesOverlapped`
  `overflow` input argument is not required.
- `ps5000aGetValuesOverlappedBulk`

overflow input argument is not required.
- ps5000aGetValuesTriggerTimeOffsetBulk
  timesUpper, timesLower and timeUnits input arguments are not required.
- ps5000aGetValuesTriggerTimeOffsetBulk64
  times and timeUnits input arguments are not required.
- ps5000aIsReady
  ready argument is not required.
- ps5000aIsTriggerOrPulseWidthQualifierEnabled
  triggerEnabled and pulseWidthQualifierEnabled input arguments are not required.
- ps5000aMaximumValue
  value input argument is not required.
- ps5000aMemorySegments
  nMaxSamples input argument is not required.
- ps5000aMinimumValue
  value input argument is not required.
- ps5000aNoOfStreamingValues
  noOfValues input argument is not required.
- ps5000aPingUnit
- ps5000aRunStreaming
  sampleInterval, sampleIntervalTimeUnits, maxPreTriggerSamples, maxPostTriggerSamples and autoStop input parameters are not required.
- ps5000aSetBandwidthFilter
- ps5000aSetChannel
- ps5000aSetDataBuffer
- ps5000aSetDataBuffers
- ps5000aSetDeviceResolution
  The resolution input argument is specified in bits. This function will also return the actual resolution set (determined by the number of channels enabled).
- ps5000aSetEts
  sampleTimePicoseconds input argument is not required.
- ps5000aSetEtsTimeBuffer
  buffer input argument is not required.
- ps5000aSetEtsTimeBuffers
  timeUpper and timeLower input arguments are not required.
- ps5000aSetNoOfCaptures
- ps5000aSetPulseWidthQualifier
- ps5000aSetTriggerChannelDirections
  aux input argument not required.
- ps5000aSigGenSoftwareControl
- ps5000aStop

3.2.3.2 Wrapper functions

A number of functions are provided in ps5000aWrap.dll which is loaded by the Instrument Driver. These provide a method of obtaining streaming data and information without having to use the callback function required by the ps5000aGetStreamingLatestValues function in the core API. Additionally there are functions that avoid the need to use structures for advanced trigger functionality.

- autoStopped
- availableData
- clearTriggerReady
- getStreamingLatestValues
- isReady
- isTriggerReady

- setAppAndDriverBuffers
- setMaxMinAppAndDriverBuffers
- setPulseWidthQualifer
- setTriggerConditions
- setTriggerProperties

3.2.3.3 Other functions

The following functions are either additional functions or modifications of the core API functions:

- getBlockData
- getRapidBlockData
- getUnitInfo - replaces ps5000aGetUnitInfo
- resetDevice
- runBlock - replaces the need to call ps5000aRunBlock and ps5000aIsReady
- setAdvancedTrigger
- setChannelDefaults
- setSigGenArbitrary - replaces ps5000aSetSigGenArbitrary
- setSigGenArbitrarySimple
- setSigGenBuiltIn - replaces ps5000aSetSigGenBuiltIn
- setSigGenBuiltInSimple
- setSigGenOff
- setSimpleTrigger - replaces ps5000aSetSimpleTrigger
- setTriggerOff

Note that the changes include the use of millivolts as opposed to ADC counts to define trigger thresholds and the use of frequency in hertz as opposed to a delta phase for the arbitrary waveform generator.

3.2.4 Connecting to/disconnecting from a PicoScope 5000A/B Oscilloscope

Connecting to a device

Prior to establishing a connection, load the PS5000aConfig.m file into the MATLAB environment.

Use the icdevice command to create an instance of an object representing the PicoScope 5000A/B device. This loads the required library files.

It is possible to specify the batch/serial number of the device to connect to when creating the device object:

```
ps5000aDeviceObject = icdevice('picotech_ps5000a_generic',
'TEST/001');
```

Alternatively, to proceed without specifying a batch/serial number:

```
ps5000aDeviceObject = icdevice('picotech_ps5000a_generic', '');
```

To connect to the device, use the connect command:

```
connect(ps5000aDeviceObject);
```

Connecting to multiple units

Connecting to multiple units is not fully supported in this release – it may be possible for block and rapid-block mode captures and signal generator operation.

Disconnecting from a device

To disconnect a device from the PC use the disconnect command:

```
disconnect(ps5000aDeviceObject);
```

For further information on the process of creating as well as connecting and disconnecting to/from a device object, please refer to the MATLAB Help files.

3.3 Enumerations and structures

The prototype files supplied provide the user with the ability to access the enumerations and structures defined in the `ps5000aApi.h` header file provided in the Software Development Kit.

The example scripts show that the prototype file is loaded and assigned to a number of variables via the `PS5000aConfig` script:

```
[ps5000aMethodinfo, ps5000aStructs, ps5000aEnuminfo,
ps5000aThunkLibName] = PS5000aMFile;
```

3.3.1 Enumerations

Enumerations are accessed from the `ps5000aEnuminfo` structure e.g.:

```
channelSettings(1).range = ps5000aEnuminfo.enPS5000ARange.PS5000A_1V;
```

3.3.2 Structures

Structures are accessed from the `structs` structure:

```
trigger_channel_a_properties =
ps5000aStructs.tPS5000ATriggerChannelProperties.members;
```

3.4 Sampling modes and signal generation

As described in the main *PicoScope 5000 Series (A API) Programmer's Guide*, the oscilloscopes can run in various sampling modes.

Below are descriptions of the sequence of function calls to use in order to acquire data or control the signal generator.

3.4.1 Block mode

The sequence of steps to follow is as in the *Using block mode* section of the *Programmer's Guide* with the following exceptions:

- Step 1 – connect to the oscilloscope as described above or using the Test and Measurement Tool.
- Step 4 – Advanced Triggering is not fully supported. Use the `setSimpleTrigger` function.
- Steps 5 and 6 – use the `runBlock` function which calls the underlying `ps5000aRunBlock` function and polls the driver using the `ps5000aIsReady` function.
- Steps 7 and 8 – if non-aggregated data is required call `getBlockData` instead which takes care of setting up the data buffers and returns the data values in arrays.

3.4.2 Rapid block

The sequence of steps to follow is as in the 'Using rapid block' section of the *Programmer's Guide* with the following exceptions:

Without aggregation:

- Step 1 – connect to the oscilloscope as described above or using the Test and Measurement Tool.
- Step 4 – Advanced Triggering is not fully supported. Use the `setSimpleTrigger` function.
- Steps 6 and 7 – Use the `runBlock` function which calls the underlying `ps5000aRunBlock` function and polls the driver using the `ps5000aIsReady` function.
- Steps 8 and 9 – call `getRapidBlockData` instead which takes care of setting up the data buffers and returns the data values in arrays.

3.4.3 ETS (equivalent-time sampling)

The sequence of steps to follow is as in the 'Using ETS mode' section of the *Programmer's Guide* with the following exceptions:

- Step 1 – connect to the oscilloscope as described above or using the Test and Measurement Tool.
- Step 4 – Advanced Triggering is not fully supported. Use the `setSimpleTrigger` function.
- Steps 5 and 6 – use the `runBlock` function which calls the underlying `ps5000aRunBlock` function and polls the driver using the `ps5000aIsReady` function.
- Steps 7 and 8 – if non-aggregated data is required call `getBlockData` instead, which takes care of setting up the data buffers and returns the data values in arrays.

3.4.4 Streaming

The sequence of steps to follow is as in the 'Using streaming mode' section of the *Programmer's Guide* with the following exceptions:

- Step 1 – connect to the oscilloscope as described above

  Prior to step 5, call `setAppAndDriverBuffers` or `setAppAndDriverBuffers` to inform the wrapper driver of the driver buffer from step 4, and the application buffer into which the data should be copied into. The length of the driver and application buffer *must* be the same. This ensures that data is correctly copied inside the streaming callback function in the wrapper driver.

- Step 6 – call `getStreamingLatestValues` then poll the driver using the `isReady` function

  Once the driver indicates that it is ready, call `availableData` to verify if any data has been collected, and if so:

  o Call `isTriggerReady` to verify if the scope has triggered (if a trigger has been setup) and the zero-based index in the buffer where the first sample is located.

- Call `autoStopped` if the `autoStop` property has been set to 1 to verify if the scope has stopped automatically.

3.4.5 Signal generator

Once a connection to the device has been established, the following functions may be used to output waveforms from the signal generator:

- `setSigGenArbitrary`
- `setSigGenArbitrarySimple`
- `setSigGenBuiltIn`
- `setSigGenBuiltInSimple`

Calling the `setSigGenOff` function will output a 0V DC signal.

The `setSigGenArbitrarySimple` and `setSigGenBuiltInSimple` functions allow you to output a waveform at constant frequency, minimising the number of parameters to provide to the function.

The trigger functions may be called if a trigger source has been specified and the `ps5000aSigGenSoftwareControl` function may be called to provide a software trigger.

Arbitrary Waveforms

Arbitrary waveforms may be created as an array of data values in the MATLAB environment as in the Signal Generator example script. Data values must be in the range +1 to −1 where these correspond to the maximum and minimum peak values for the signal, although the supplied `normalise` function may be used to convert the data values.

Additionally, files generated using the PicoScope 6 software may be read in using the supplied `importAWGfile` function or through the MATLAB Import tool.

Note that only the models with a 'B' suffix, e.g. the PicoScope 5242B, have an arbitrary waveform generator.

3.5 Test and Measurement Tool

The MATLAB Instrument and Control Toolbox Test and Measurement Tool can be used to control the device and is suited for the following modes:

- Block data capture
- Rapid block data capture
- Equivalent time sampling (ETS) data capture
- Signal generator operation

Data in the MATLAB Workspace may be used as arguments (for example a defined waveform for the AWG) – use the following syntax in the inputs field:

```
evalin('base', 'variable name')
```

The variable name should be entered in quotes.

3.5.1 Connecting to the PicoScope device

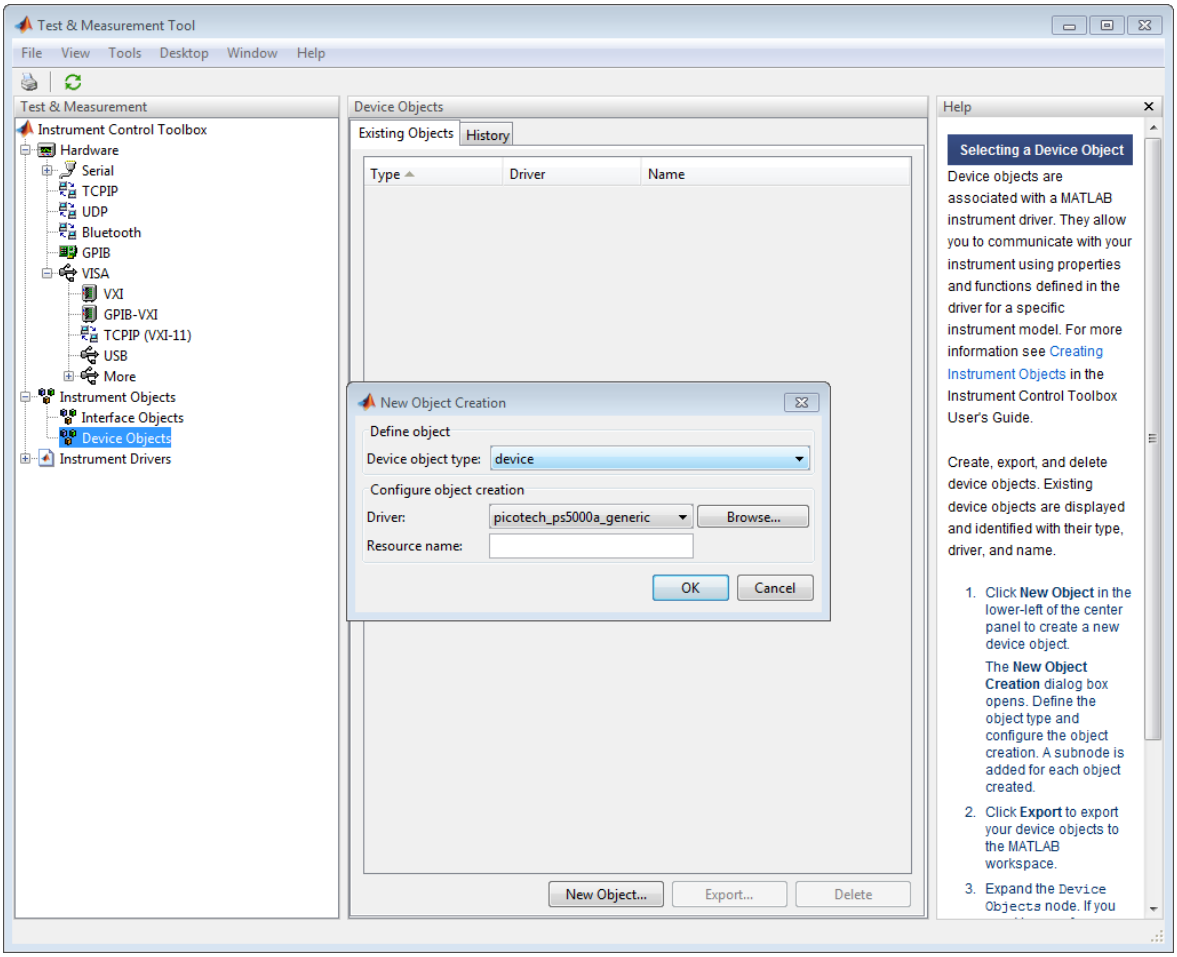
1. Load the `PS5000aConfig.m` file in MATLAB.

2. At the command prompt, type `tmtool` or with MATLAB R2012b or later, click the Instrument Control App from the **Apps** toolbar.

3. In the **Test & Measurement Tool** window that appears, expand the **Instrument Object** in the **Test & Measurement** pane and left-click on **Device Objects**.

4. Click the **New Object...** button at the bottom of the **Existing Objects** tab in the **Device Objects** pane on the right.

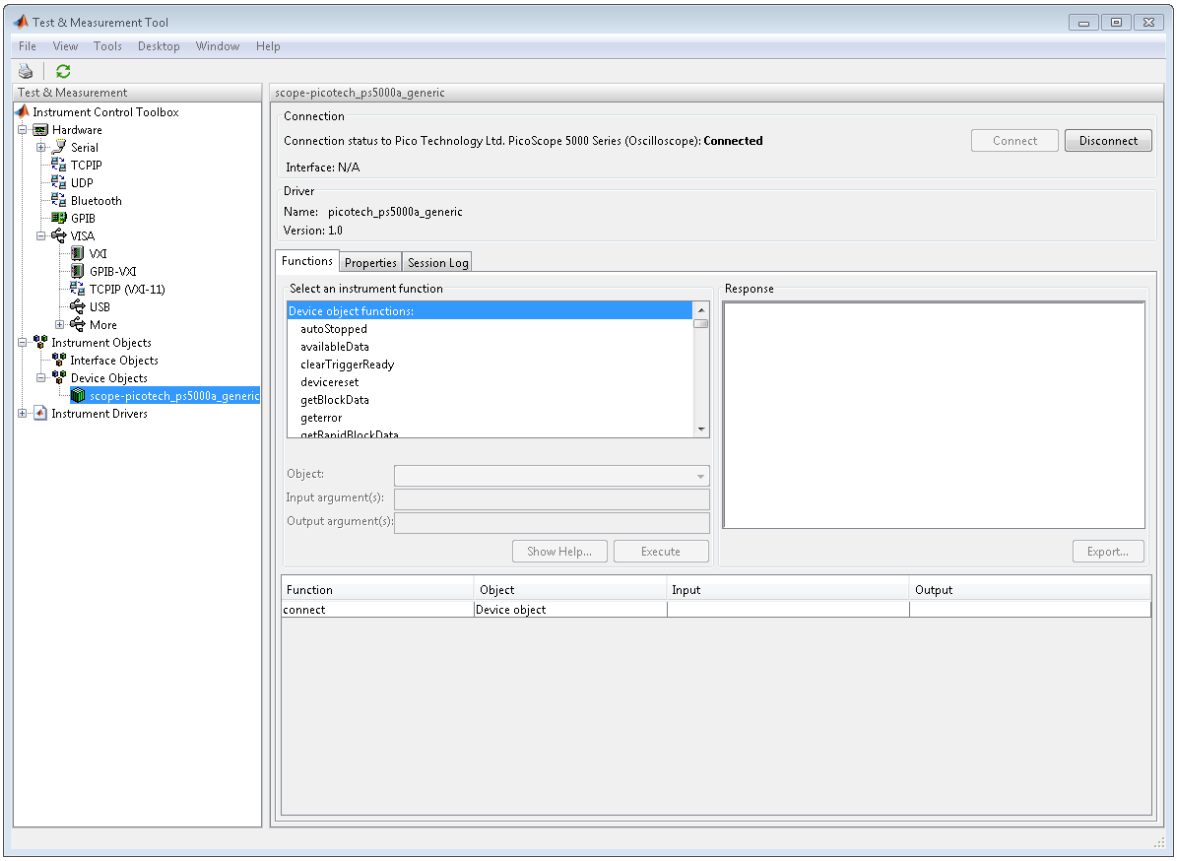
5. In the **New Object Creation** dialog that appears, ensure that the parameters are shown as in the figure below. The batch/serial number of the device may be entered in the **Resource Name** field.



New Object Creation dialog

6. Click **OK**.

7. Click the 'scope-picotech_ps5000a_generic' item that appears under the **Device Objects**, and then click **Connect**.



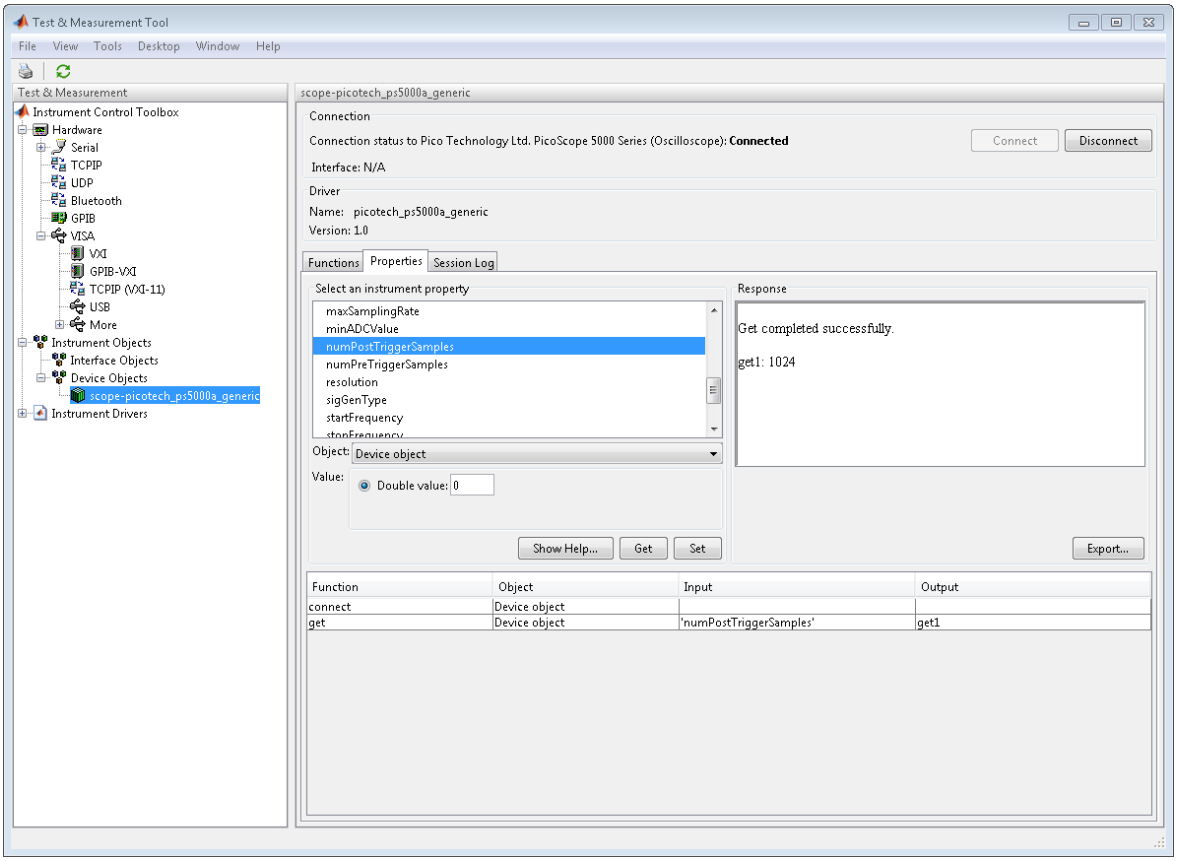
Device connected

3.5.2 Getting and setting properties

1. Click on the **Properties** tab in the **scope-picotech_ps5000a_generic** pane.

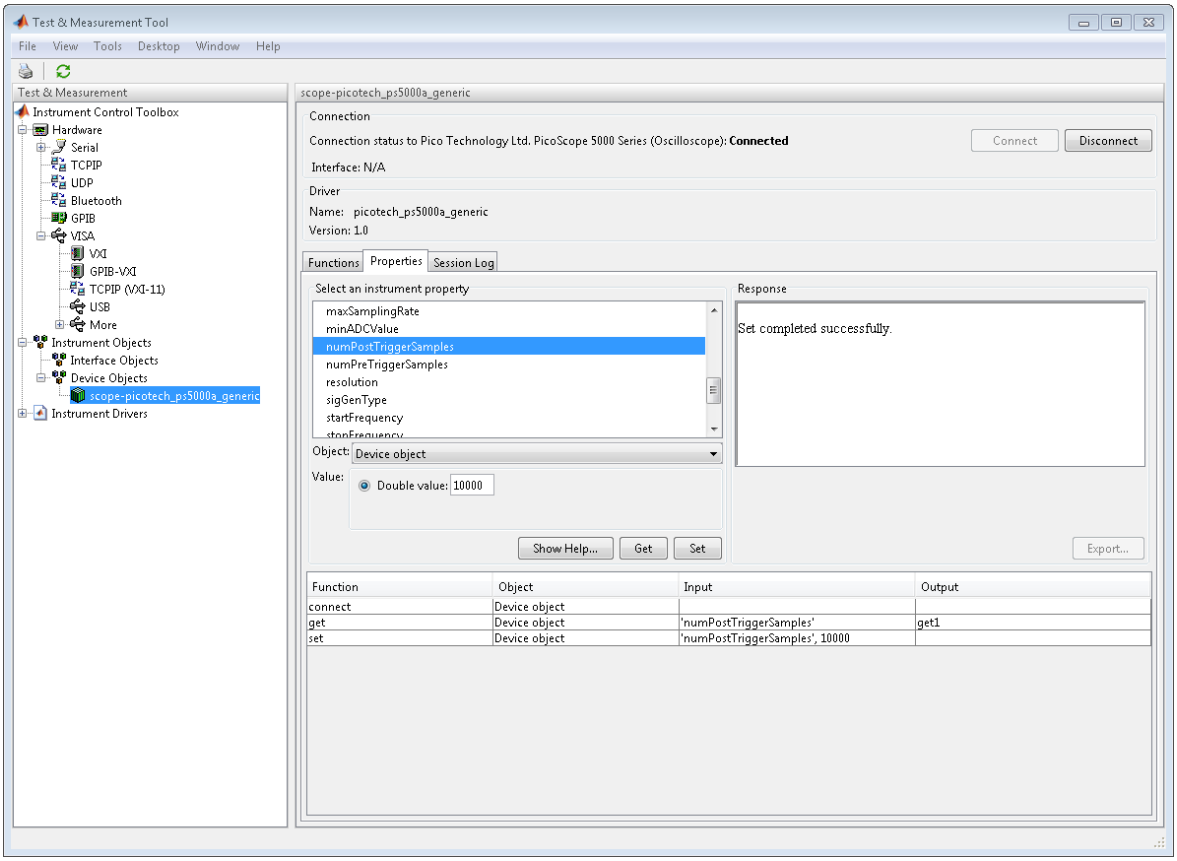
2. Select an instrument property, such as **numPostTriggerSamples**.

3. To retrieve the value, click **Get** – the value will appear in the **Response** pane to the right.



Retrieving a property

4. To set a value for the property, enter a value in the textbox and click **Set**.



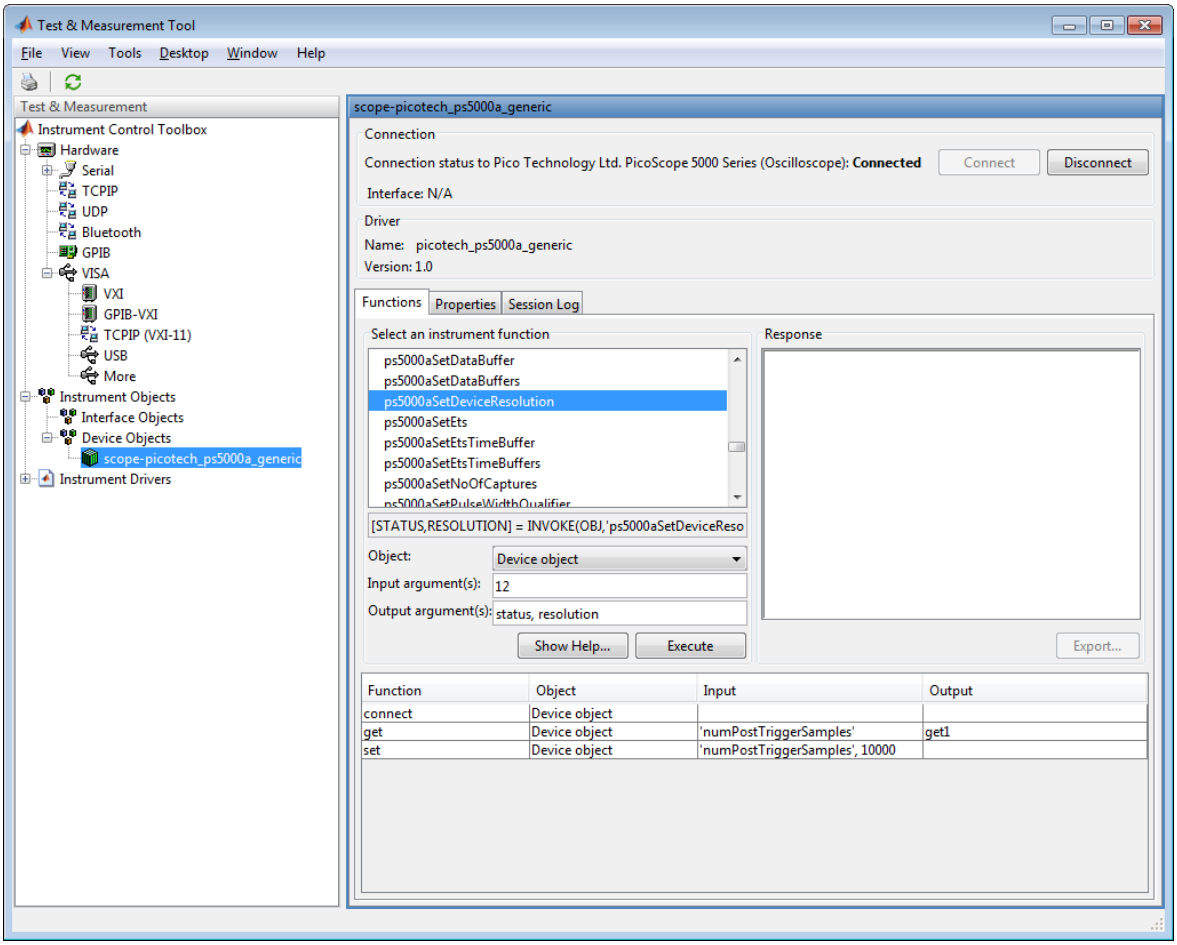
Setting a property

3.5.3 Calling functions

1. Click the **Functions** tab.

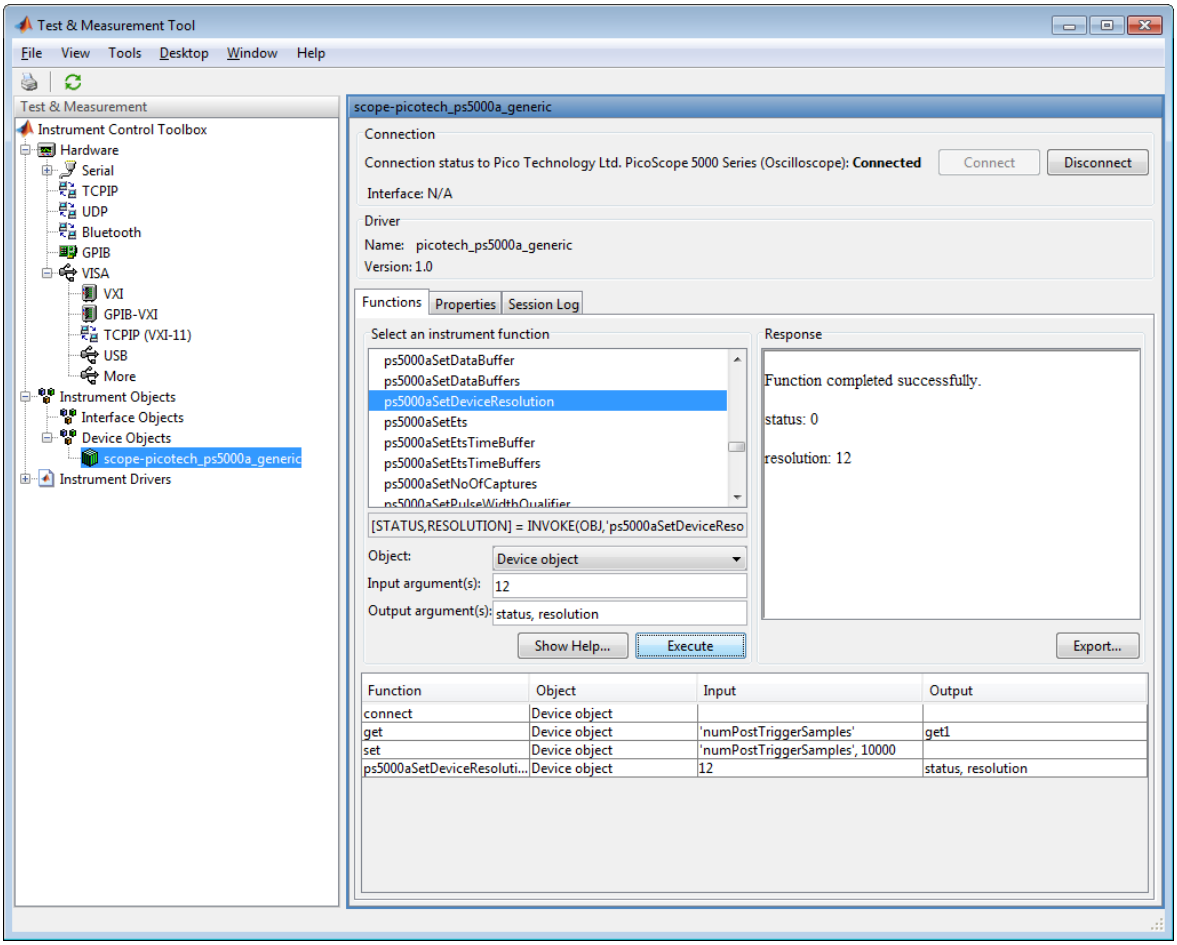
2. Select an instrument function from the list, such as **ps5000aSetDeviceResolution**

3. Set the input and output arguments - note that only the basic data types are supported by the Test and Measurement Tool.



Setting function parameters

4. Click **Execute**.



Completed function execution

Data returned from the function can be exported to the MATLAB workspace using the **Export...** button.

3.5.4 Disconnecting from the PicoScope device

1. Click the **Disconnect** button at the top right of the **scope-picotech_ps5000a_generic** pane – the **Connection status** should change to **Disconnected**.

4 Troubleshooting

This section covers some troubleshooting information for common errors that may be encountered.

Unable to connect to a device

Check that the device is connected to the PC and listed in the Windows Device Manager.

Confirm that the LED on the front of the device is illuminated.

Ensure that no other instance of a device object corresponding to a previous connection is present – if so, delete it.

Ensure that the `PS5000aConfig` file has been loaded.

Attempt to connect to a device having previously terminated a script with Ctrl-C results in a lockup

Disconnect the USB cable from the oscilloscope and close MATLAB using the Windows Task Manager.

Connect the USB lead again.

'Not a valid Win32 application' error when loading the DLLs

Ensure that the correct 32-bit or 64-bit dlls are on your MATLAB path or in the same directory as the Instrument Driver file depending on whether using MATLAB 32-bit or 64-bit.

Warnings relating to FcnPtr when running a script

This is expected and refers to the C-style callback functions for block and streaming mode.

General

NOTE: Always call the disconnect function on the device object on completion of data acquisition/signal generator output

If assistance is still required, please contact support@picotech.com or call using the Pico Technology contact details provided earlier in this guide.

Index


pico
Technology

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
www.picotech.com

ps5000ab.matlab.en r1 2014-02-19