CompuScope Software Development Kit (SDK) for Python for Windows

User's Guide

Revision 1.04 – 10/14/2021

Gage Contact Information	
Toll Free:	1-800-567-GAGE
Tel:	1-514-633-7447
Fax:	1-514-633-0770
Sales Email:	sales@gage-applied.com
Support Email:	tech-support@gage-applied.com
Web:	http://www.gage-applied.com

Gage is a Product Brand of:

Vitrek, LLC
900 North State Street
Lockport, Illinois 60441-2200
USA

Toll Free: 1-800-DATA-NOW
Tel: 1-815-838-005
Fax: 1-815-838-4424

http://www.vitrek.com

How to reach Gage Product Support

Toll-free phone: (800) 567-4243
Toll-free fax: (800) 780-8411

To reach Gage Product Support from outside North America

Tel: +1-514-633-7447
Fax: +1-514-633-0770

E-mail: prodinfo@gage-applied.com
Web site: http://www.gage-applied.com
On-line Support Request Form: http://www.gage-applied.com/support/support-form.php

Table of Contents

TABLE OF CONTENTS3

PREFACE5

REQUIREMENTS5

INSTALLATION5

OVERVIEW6

PYTHON SDK FUNCTIONS6

Initialize6
GetSystem (board_type, channels, sample_bits, index)7
FreeSystem (handle)7
GetSystemInfo (handle)7
GetBoardsInfo (handle)7
GetAcquisitionConfig (handle, config)8
SetAcquisitionConfig (handle, dict)8
GetChannelConfig (handle, chan, config)8
SetChannelConfig (handle, chan, dict)9
GetTriggerConfig (handle, trig, config)9
SetTriggerConfig (handle, trig, dict)9
GetExtendedBoardOptions (handle)9
GetStreamTotalDataSizeInBytes (handle)10
GetStreamSegmentDataSizeInSamples (handle)10
GetTimeStampFrequency (handle)10
GetDataPackingMode (handle)10
GetStreamingCaptureMode (handle)10
GetDataFormatInfo (handle)10
GetTriggeredInfo (handle, start, count)11
GetFftConfig (handle)11
GetSegmentTailSizeInBytes (handle)11
GetMulRecAverageCount(handle)11
SetDataPackingMode (handle, mode)11
SetStreamingCaptureMode (handle, mode)12
SetFftMode (handle, dict)12
SetFftWindowConfig (handle, windowSize, list)12
SetMulRecAverageCount (handle, count)12
SetIdentityLed (handle)12
SetOneSampleResolution (handle)13
GetSystemCaps (handle, capsID)13
Commit (handle)17
StartCapture (handle)17
ForceCapture (handle)17
AbortCapture (handle)17
GetStatus (handle)18
TransferData (handle, channel, mode, segment, start, length)18
GetStreamingBuffer (handle, cardIndex, size)18
FreeStreamingBuffer (handle, cardIndex, buffer)19
TransferStreamingData (handle, cardIndex, buffer, transferSizeInSample)19
GetStreamingTransferStatus (handle, cardIndex, waitTimeout)19
ConvertToSigHeader (dict, comment, name)19
GetErrorString (errorCode)20

SUPPORT FUNCTIONS IN GAGESUPPORT.PY20

CalculateChannelIndexIncrement (mode, channel_count, board_count)20
LoadAcquisitionConfiguration (handle, iniFile)20
LoadChannelConfiguration (handle, channel, iniFile)21
LoadTriggerConfiguration (handle, trigger, iniFile)21
LoadApplicationConfiguration (iniFile)21
SaveFile (filename, channel, buffer, format, stHeader)21

TECHNICAL SUPPORT23

Preface

This manual is meant to serve as an aid to engineers using CompuScope series of high-speed data acquisition cards in the Python programming language.

Throughout this manual, it is assumed that you are familiar with the Python programming environment. It is also assumed that you have correctly installed and configured the CompuScope drivers and that you have installed the Python programming environment of your choice. The CompuScope Python SDK has been tested with Python 3.6 and Python 2.7, in both 32-bit and 64-bit version. Other versions, provided they are fairly recent, should work, but cannot be guaranteed to work.

The Python SDK consists of a Python library (with a .PYD extension), several Python support files and one or more sample programs. The library is a wrapper over the CompuScope C/C# SDK, so further information can be found in the CompuScope C/C# SDK User's Guide.

Requirements

The Gage CompuScope SDK for Python supports the following versions of Python:

Python version 2.7 (32-bit and 64-bit).
Python 3.6 or newer (32-bit and 64-bit).

The Gage Python SDK does require some additional files, which may or may not come with your Python distribution. If not, they are easily available from the Python Package Index using pip. If you are using Anaconda Python, they can be installed with the Conda utility. The required files are:

configparser	(ConfigParser for Python versions < 3.0) library used for reading and writing INI files.
numpy	(version 1.16 or greater) library with support for large, contiguous arrays. Used in the CompuScope Python SDK for transferring data from the CompuScope board to user memory.
future	Only required for python 2. Provides __future__ and builtins. Allows python 3 compatibility.

Installation

If you purchased the CompuScope C/C# SDK you will have been shipped a software key that allow installation of the SDK from the Gage CompuScoppe CD. Simple select the installation of the C/C# SDK from the CompuScope CD and enter the key when prompted. The Python SDK will be installed along with the C/C# SDK.

Overview

By default, the Python SDK will install itself in the OS system drive:
C:\Program Files\Gage\CompuScope\CompuScope C_C# SDK\Python.
It is recommended that you use the default installation. All the necessary files to use the Python SDK will be in this folder.

The files provided are:

GageSupport.py	support python file with some commonly used functions and routines, particularly for reading INI files and configuring the CompuScope board
GageConstants.py	support file containing constants used with the Python SDK
GageErrrors.py	support file containing error constants used with the Python SDK
PyGage2_32.py	Python C extension for use with 32-bit Python 2.7
PyGage2_64.py	Python C extension for use with 64-bit Python 2.7
PyGage3_32.py	Python C extension for use with 32-bit Python 3.6
PyGage3_64.py	Python C extension for use with 64-bit Python 3.6
GageAcquire.py	sample program to demonstrate basic functionality

Several other sample programs may be provided. If they are, they will mirror the respective C version's functionality so the description in the C SDK User's Guide will be applicable to the Python sample.

The sample programs all have code in the beginning to determine what platform (32-bit or 64-bit) and which version of Python (2 or 3) they are running on and import the appropriate pyd file as PyGage. If you know the platform and version you're running on and wish to skip this code you can just use the original name to import the library.

For a more complete description of the sample programs and the general program structure, refer to the CompuScope C/C# SDK User's Guide. The Python examples also use INI files to configure the board. A description of these files can be found in the INI_FILE_DESCRIPTION document that comes with the SDK. Note that not all of the C/C# sample programs are available in the Python SDK, though all the functionality demonstrated in the C sample programs is available in the Python SDK.

Python SDK Functions

Initialize

Initializes all the CompuScope systems in the computer. The return value is an integer which, if positive, is the number of CompuScope systems found. Otherwise, the return value is a negative integer which denotes a CompuScope error code, which can be found in GageErrors.py.

GetSystem (board_type, channels, sample_bits, index)

Gets ownership of a CompuScope system. The parameters board_ type, channels, sample_ bits are used to specify certain systems. For example, if the value of 'sample bits' is 8, then the function will retrieve the first 8-bit system it finds. The index parameter can be used to retrieve a certain system in the list, for example index = 2 would retrieve the second system. If all parameters are 0, the function gets the first free system it finds. The return value upon success is a handle, an unsigned integer which is a unique identifier for the CompuScope system and is used in subsequent calls. If the call fails, it returns a CompuScope error code.

FreeSystem (handle)

Frees the system associated with the handle. No other application can use a system until the associated handle is freed. Return a 1 if successful, a CompuScope error code if not.

GetSystemInfo (handle)

Retrieves static information for the CompuScope system associated with handle. If the function fails, the return value is an integer representing a CompuScope error code. If the function succeeds it returns a dictionary with the following fields:

MaxMemory	Memory size, in samples of each board in a CompuScope system.
SampleBits	Vertical resolution, in bits, of the system.
SampleResolution	Sample resolution of the system, used for voltage conversion.
SampleSize	Sample size in bytes.
SampleOffset	ADC output that corresponds to the middle voltage of the input range. Used for voltage conversion.
BoardType	Numeric constant that indicates the CompuScope model.
BoardName	String containing the CompuScope model name.
AddonOptions	Options encoding features like, gate, trigger enable, etc.
BaseBoardOptions	Options encoding features like gate, trigger enable, etc.
TriggerMachineCount	Number of trigger engines available in the CompuScope system.
ChannelCount	Number of channels in the CompuScope system.
BoardCount	Number of boards in the CompuScope system.

GetBoardsInfo (handle)

Retrieves static information for each board in the CompuScope system associated with handle. If the function fails, the return value is an integer representing a Gage error code. If the function succeeds the return value is a list of dictionaries, one for each board in the system. Each dictionary will have the following fields:

BoardType	Constant (defined in GageConstants.py) representing the board model.
SerialNumber	String containing the serial number of the board.
BaseBoardVersion	Version number of the base board.
BaseBoardFirmwareVersion	Version number of the firmware on the base board.
AddonBoardVersion	Version number of the Addon board.
AddonBoardFirmwareVersion	Version number of the firmware on the Addon board.
AddonFwOptions	Options of the current firmware image of the Addon board.

BaseBoardFwOptions	Options of the current firmware image of the base board.
AddonHwOptions	Options of the Addon board pcb.
BaseBoardHwOptions	Options of the base board pcb.

GetAcquisitionConfig (handle, config)

Retrieves the current acquisition parameters for the CompuScope system associated with handle. If config is set to CS_CURRENT_CONFIGURATION (defined in GageConstants.py) the current settings of the driver are retrieved. If config is set to CS_ACQUISITION_CONFIG the current settings of the hardware are retrieved. Once Commit is called, the driver settings are sent to the hardware so they will be the same. The config parameter is optional, and if not used the settings from the driver are returned. If the function fails, the return value is an integer representing a CompuScope error code. If it succeeds, the return value is a dictionary with the following fields:

SampleRate	Sample rate value in Hz.
ExternalClock	External clocking status. 0 = “inactive”, otherwise “active”.
Mode	Acquisition mode of the system.
SampleBits	Actual vertical resolution, in bits, of the CompuScope system.
SampleResolution	Actual sample resolution of the system.
SampleSize	Actual sample size, in bytes, of the CompuScope system.
SegmentCount	Number of segments per acquisition.
Depth	Number of samples to capture after the trigger event.
SegmentSize	Maximum number of points stored for one segment acquisition.
TriggerTimeout	Amount of time to wait (in 100 ns units) after start of segment acquisition before forcing a trigger event. Timeout counter is reset for every segment in multiple record acquisition
TriggerDelay	Number of samples to skip after the trigger event before starting to decrement the depth counter.
TriggerHoldoff	Number of samples to acquire before enabling the trigger circuitry. The amount of pre-trigger data is determined by TriggerHoldoff and has a maximum value of SegmentSize – Depth.
SampleOffset	Actual sample offset for the CompuScope system.
TimeStampConfig	Time stamp mode. Multiple selections may be ORed together. Available values (defined inGageConstants.py) are TIMESTAMP_GCLK, TIMESTAMP_MCLK, TIMESTAMP_SEG_RESET, TIMESTAMP_FREERUN.

SetAcquisitionConfig (handle, dict)

Sends the acquisition parameters contained dict to the CompuScope system indentified by handle. The variable dict is a dictionary as described in GetAcquisitionConfig. If any parameters are missing from dict, they will retain their previous values. If the function succeeds the return value is 1, otherwise a negative integer which represents a CompuScope error code is returned.

GetChannelConfig (handle, chan, config)

Retrieves channel parameters for the channel specified by chan for the CompuScope system associated with handle. If config is set to CS_CURRENT_CONFIGURATION (defined in GageConstants.py) the current settings of the driver are retrieved. If config is set to CS_ACQUISITION_CONFIG the current settings of

the hardware are retrieved. Once Commit is called, the driver settings are sent to the hardware so they will be the same. The config parameter is optional, and if not used the settings from the driver are returned. Channel numbers begin from 1. If the function fails, the return value is an integer representing a CompuScope error code. If it succeeds, the return value is a dictionary with the following fields:

InputRange	Channel full scale input range in mV peak-to-peak.
Impedance	Channel impedance in Ohms (50, 1000000 or Hi-Z).
Filter	Index of the filter to be used. Default is 0 (no filter).
DcOffset	Channel DC offset, in mV.
Coupling	Channel coupling, either DC (1) or AC (2).

SetChannelConfig (handle, chan, dict)

Sends the channel parameters contained in dict for the channel specified in chan to the CompuScope system identified by handle. The variable dict is a dictionary as described in GetChannelConfig(). If any parameters are missing from dict, they will retain their previous values. If the function succeeds the return value is 1, otherwise a negative integer which represents a CompuScope error is returned.

GetTriggerConfig (handle, trig, config)

Retrieves trigger parameters for the trigger engine specified by trig for the CompuScope system associated with handle. If config is set to CS_CURRENT_CONFIGURATION (defined in GageConstants.py) the current settings of the driver are retrieved. If config is set to CS_ACQUISITION_CONFIG the current settings of the hardware are retrieved. Once Commit is called, the driver settings are sent to the hardware so they will be the same. The config parameter is optional, and if not used the settings from the driver are returned. Trigger engine numbers begin from 1. If the function fails, the return value is an integer representing a CompuScope error code. If it succeeds, the return value is a dictionary with the following fields:

Level	Trigger level as a percentage of the trigger source input range.
Source	Trigger source (see constants in GageConstants.py).
ExtCoupling	External trigger coupling, either DC (1) or AC (2).
ExtRange	External trigger full scale input range in mV.
ExtImpedance	External trigger impedance, I Ohms.
Relation	Logical relation applied to the trigger engine outputs (default OR).

SetTriggerConfig (handle, trig, dict)

Sends the trigger parameters contained in dict for the trigger engine specified in trig to the CompuScope system identified by handle. The variable dict is a dictionary as described in GetTriggerConfig(). If any parameters are missing from dict, they will retain their previous values. If the function succeeds the return value is 1, otherwise a negative integer which represents a CompuScope error is returned.

GetExtendedBoardOptions (handle)

Retrieves information about the 2nd and 3rd Baseboard FPGA images for the CompuScope system associated with handle. If the call is successful, the return value is a 64-bit integer with any available options OR'd in. otherwise a negative integer with represents a CompuScope error is returned.

GetStreamTotalDataSizeInBytes (handle)

Queries for the total amount of data in expert streaming for the CompuScope system associated with handle. If successful, the return value is a 64-bit integer that represents the total amount of data in bytes. If unsuccessful the return value will be a negative integer which represents a CompuScope error code.

GetStreamSegmentDataSizeInSamples (handle)

Queries for the segment data size in expert streaming for the CompuScope system associated with handle. If successful, the return value is a 64-bit integer that represents the total amount of data in a segment in samples. If unsuccessful the return value will be a negative integer which represents a CompuScope error code.

GetTimeStampFrequency (handle)

Returns the frequency of the timestamp counter in Hertz for the CompuScope system associated with handle. The timestamp counter frequency can be used to calculate the time between consecutive timestamps. If the call is successful, the return value is a 64-bit integer containing the counter frequency in Hertz. If unsuccessful, the return value is a negative integer representing a CompuScope error code.

GetDataPackingMode (handle)

Returns the current data packing mode for expert streaming for the CompuScope system associated with handle. If successful, the return value will be one of the following: 0 – Unpacked, 1 – Data Packed 8 bit, 2 – Data Packed 12-bit. If unsuccessful, the return value is a negative integer representing a CompuScope error code.

GetStreamingCaptureMode (handle)

Returns the current mode for expert streaming for the CompuScope system associated with handle. If successful, the return value will be either 0 (Memory Mode) or 1 (Streaming Mode). If unsuccessful, the return value is a negative integer representing a CompuScope error code.

GetDataFormatInfo (handle)

Retrieves information about the format of the data in expert streaming mode for the CompuScope system associated with handle. If successful, the return value is a dictionary with the following fields:

Signed	1 means data is signed, 0 means data is unsigned.
Packed	1 means data is packed, 0 means unpacked.
Sample Bits	Actual vertical resolution, in bits, of the CompuScope system.
SampleSizeBits	Actual sample size, in bits, of the CompuScope system.
SampleOffset	Actual sample offset of the CompuScope system.
SampleResolution	Actual sample resolution of the CompuScope system.

If unsuccessful, the return value is a negative integer representing a CompuScope error code.

GetTriggeredInfo (handle, start, count)

Query for the channel that caused the trigger event for the system identified by handle. Start and count are optional parameters which tell the function which segment to start with and how many to return values for. If they are not included, the default values are start = 1 and count is how many segments were captured. If the function is successful, the return value is a list of channels, one for each segment requested. If unsuccessful, the return value is a negative integer representing a CompuScope error code.

GetFftConfig (handle)

Query for the configuration parameters for FFT expert firmware for the system identified by handle. Can only be used with the FFT firmware option. If successful, the function returns a dictionary with the following values:

FftSize	Size of the FFT (read-only)
Enable	1 enables the FFT filter, 0 disables it
Average	1 enables FFT averaging, 0 disables. Currently not implemented.
RealOnly	1 means inputs are Real only, 0 means Real and Imaginary. Currently not implemented.
Windowing	Turns on Windowing. Window coefficients must be loaded with SetFftWindowConfig()
IFft	1 means perform an inverse FFT
FftMr	1 means one FFT block per acquisition segment, 0 means all FFT blocks are processed with the same acquisition segment. Currently not implemented.

If unsuccessful, the return value is a negative integer representing a CompuScope error code.

GetSegmentTailSizeInBytes (handle)

Retrieve the size (in bytes) of the segment tail size for the CompuScope system identified by handle. Some CompuScope boards have some data (tail) at the end of each segment which may contain extra information about the capture. If successful the return value is an unsigned integer containing the tail size in bytes. If unsuccessful, the return value is a negative integer representing a CompuScope error code.

GetMulRecAverageCount(handle)

Retrieve the number of averages set in the expert Signal Averaging firmware for the CompuScope system identified with handle. Can only be used with expert Signal Averaging firmware. If successful, the return value is an unsigned integer with the number of averages

SetDataPackingMode (handle, mode)

Sets the current data packing mode for expert streaming for the CompuScope system associated with handle. Mode is an integer. The available modes are:

0	Unpacked
1	8-bit Packed Data
2	12-bit Packed Data

Note: Not all modes are available for all systems, consult your CompuScope Hardware manual. If the function is successful, a 1 is returned. If unsuccessful, the return value is a negative integer representing a CompuScope error code.

SetStreamingCaptureMode (handle, mode)

Sets the current capture mode for expert streaming for the CompuScope system associated with handle. Mode is an integer. Available modes are 0 (Memory mode) or 1 (Streaming Mode). If the function is successful, a 1 is returned. If unsuccessful, the return value is a negative integer representing a CompuScope error code.

SetFftMode (handle, dict)

Sets the configuration parameters for the FFT expert firmware for the system identified by handle. Can only be used with the FFT firmware option. The dict parameter is a dictionary as described below:

FftSize	Size of the FFT (read-only)
Enable	1 enables the FFT filter, 0 disables it
Average	1 enables FFT averaging, 0 disables. Currently not implemented.
RealOnly	1 means inputs are Real only, 0 means Real and Imaginary. Currently not implemented.
Windowing	Turns on Windowing. Window coefficients must be loaded with SetFftWindowConfig()
IFft	1 means perform an inverse FFT
FftMr	1 means one FFT block per acquisition segment, 0 means all FFT blocks are processed with the same acquisition segment. Currently not implemented.

If the Windowing flag is set to 1, windowing is turned on and it is expected that the user will supply a set of 16-bit integer window coefficients by calling SetFftWindowConfig().

If successful, the function returns CS_SUCCESS (1). Otherwise, a negative integer representing a CompuScope error code is returned.

SetFftWindowConfig (handle, windowSize, list)

Sets the fft windowing coefficients for the CompuScope system associated with handle. Can only be used with the expert FFT firmware. FFT windowing mode must be turned on by calling SetFftMode(). The parameter windowSize is the number of coefficients (which should be half of the FftSize set in SetFftMode()). The list parameter is a list of 16-bit integers which are the window coefficients. If successful, the function returns CS_SUCCESS (1). Otherwise, a negative integer representing a CompuScope error code is returned.

SetMulRecAverageCount (handle, count)

Configures the number of averages when using signal averaging for the CompuScope system associated with handle. Can only be used with the expert Signal Averaging firmware. If successful, the function returns CS_SUCCESS (1). Otherwise, a negative integer representing a CompuScope error code is returned.

SetIdentityLed (handle)

Flashes the identity LED on the CompuScope system identified by handle, if available. LED will flash for about 5 seconds. Not available on all CompuScope models. If successful, the function returns CS_SUCCESS (1). Otherwise, a negative integer representing a CompuScope error code is returned.

SetOneSampleResolution (handle)

Enables one sample depth resolution on the CompuScope system identified by handle, if available. Not available on all CompuScope models. If successful, the function returns CS_SUCCESS (1). Otherwise, a negative integer representing a CompuScope error code is returned.

GetSystemCaps (handle, capsID)

This function returns information and availability about various system capabilities in the CompuScope system identified by handle. The capsID parameter is the specific capability of interest. If the function succeeds, the return value will give information about the capability requested. Otherwise, the return value will be a negative integer which represents a CompuScope error code. The capsID constants are defined in GageConstants.py.

The value(s) returned depend on which capsID is requested and will be described below:

CAPS_SAMPLE_RATES

Returns a list of tuples, each tuple consisting of a string and a value, one for each sample rate available for the current acquisition mode. An example tuple would be ('1 MHz', 1000000). Note that for some CompuScope systems, the available sample rates are dependent on the acquisition mode.

CAPS_INPUT_RANGES

This capsID must be OR'd with a channel index (starting from 1, use 0 for external trigger). Returns a list of tuples, each tuple consisting of a string and a value, one for each input range available for the requested channel. An example tuple would be ('±1 V', 2000). The value field is in millivolts peak-to-peak. Note that for some CompuScope systems, the available input ranges are dependent on the current impedance.

CAPS_IMPEDANCES

This capsID must be OR'd with a channel index (starting from 1, use 0 for external trigger). Returns a list of tuples, each tuple consisting of a string and a value, one for each impedance available. An example tuple would be ('50 Ohm', 50). The value field is in ohms.

CAPS_COUPLINGS

This capsID must be OR'd with a channel index (starting from 1, use 0 for external trigger). Returns a list of tuples, each tuple consisting of a string and a value, one for each coupling available for the current input range. An example tuple would be ('DC', 1). The value field is the value of the CompuScope constant, CS_COUPLING_DC or CS_COUPLING_DC, available in GageConstants.py.

CAPS_ACQ_MODES

Returns a list of tuples, each tuple consisting of a string and a value, one for each available acquisition mode in the CompuScope system. An example tuple would be ('Dual', 2). The value field is the value of the CompuScope constant for the acquisition mode, which is available in GageConstants.py.

CAPS_TERMINATIONS

This capsID must be OR'd with a channel index (starting from 1, use 0 for external trigger). Returns a list of tuples, each tuple consisting of a string and a value, one for each termination available. An example tuple would be ('Direct ADC', 8). The value field is the value of the CompuScope constant, CS_DIFFERENTIAL_INPUT or CS_DIRECT_ADC_INPUT, available in GageConstants.py.

CAPS_TRIGGER_SOURCES

Returns a list of tuples, each tuple consisting of a string and a value, one for each trigger source available in the CompuScope system. An example tuple would be ('Channel 1', 1). The value field is the value of the CompuScope constant, CS_TRIG_SOURCE_DISABLE, CS_TRIG_SOURCE_EXT or a channel number. The constants are available in GageConstants.py. This define can be OR'd with an acquisition mode, i.e. CS_MODE_DUAL, etc. to retrieve trigger sources available only for that mode.

CAPS_FILTERS

Returns a list of tuples, each tuple consisting of a string and a value, one for each built-in filter available in the CompuScope system. An example tuple would be ('Low Cutoff', 1). The value field indicates the frequency of the filter. If the function returns an error, it means that filter capability is not available. Not all CompuScope models have filter capability.

The following defines return an unsigned integer with the requested information.

CAPS_FLEXIBLE_TRIGGER

Query for availability of flexible triggering (triggering from any card in master / slave system). Return value is an unsigned integer.

0 – no support for flexible trigger
1 – full support for flexible trigger
2 – supports flexible trigger on input channels, but only 1 external trigger

CAPS_MAX_SEGMENT_PADDING

Query for max padding for segment or depth.

CAPS_BOARD_TRIGGER_ENGINES

Query for number of trigger engines per board.

CAPS_FWCHANGE_REBOOT

Query if the system needs to be powered down after switching expert images. Return value is 0 (No), 1 (Yes).

CAPS_SKIP_COUNT

Query for external clock skip count.

CAPS_MAX_PRE_TRIGGER

Query for maximum pre-trigger depth.

CAPS_DEPTH_INCREMENT

Query for the amount that the depth can be increased or decreased.

CAPS_TRIGGER_DELAY_INCREMENT
Query for the amount that the trigger delay can be increased or decreased.

CAPS_TRIGGER_RES
Query for the trigger resolution.

CAPS_TRIG_ENGINES_PER_CHAN
Queries for the number of trigger engines per channel.

CAPS_STM_TRANSFER_SIZE_BOUNDARY
Queries for the transfer size boundary in expert Streaming mode.

The following defines return a signed 32-bit integer with the requested information. If the capability exists, a 1 (CS_SUCCESS) is returned. Otherwise, an error code CS_FUNCTION_NOT_SUPPORTED is returned.

CAPS_DC_OFFSET_ADJUST
Queries if DC offset adjust is available.

CAPS_CLK_IN
Queries for external synchronization clock inputs.

CAPS_BOOTIMAGE0
Queries for FPGA Boot image capability.

CAPS_EXT_TRIGGER_UNIPOLAR
Queries for external trigger unipolar capability.

CAPS_MULREC
Queries for multiple record capabilities.

CAPS_SELF_IDENTIFY
Queries for the ability to flash an LED upon request.

CAPS_SINGLE_CHANNEL_2
Queries for the capability of capturing data from channel 2 in single channel mode.

CAPS_TRANSFER_EX
Query for CsTransferEx() support.

The following defines return a 64-bit signed integer with the value requested. If the function fails, an error code is returned.

CAPS_MIN_EXT_RATE
Returns the minimum allowable external clock rate.

CAPS_MAX_EXT_RATE

Returns the maximum allowable external clock rate.

The following defines return a list of tuples if successful. If not, the return value is a negative integer denoting a CompuScope error code. If the system does not support the capability, an error will also be returned.

CAPS_AUX_CONFIG

Query for Aux input/output capabilities. If successful, the return value is a list of tuples with the following values:

ClockOut
TriggerOut
TrigOut_In
AuxInput
AuxOutput

A 1 signifies the capability is there, a 0 means it is not. For example, (ClockOut, 1) would mean that the system has the clock out capability.

CAPS_CLOCK_OUT

Queries for clock out capabilities. If successful, the return value is a list of tuples with the values being a string with the capability name and the CompuScope constant value that defines it. For example:

('Disable', CS_OUT_NONE)
('Sample Clock', CS_CLKOUT_SAMPLE)
('Reference Clock', CS_CLKOUT_REF)

CAPS_TRIG_OUT

Queries for trigger out capabilities. If successful, the return value is a list of tuples with the values being a string with the capability name and the CompuScope constant value that defines it. For example:

('Disable', CS_OUT_NONE)
('Trigger Out', CS_TRIGOUT)

CAPS_AUX_OUT

Queries for aux out capabilites. If successful, the return value is a list of tuples with the values being a string with the capability name and the CompuScope constant value that defines it. For example:

('Disable', CS_OUT_NONE)
('Busy Out', CS_PULSEOUT_DATAVALID)

CAPS_AUX_IN_TIMESTAMP

Queries for aux in timestamp capabilities. If successful, the return value is a list of tuples with the values being a string with the capability name and the CompuScope constant value that defines it. For example:

('Disable', CS_OUT_NONE)
('Enabled (Live Pos. Level)', CS_EXTTRIGGEN_LIVE)

CAPS_TRIG_ENABLE

Queries for trig enable. If successful, the return value is a list of tuples with the values being a string with the capability name and the CompuScope constant value that defines it. For example:

('Always Enable', 0)
('Gate (Live Pos. Level', CS_EXTTRIGGEN_LIVE)
('Pos. Edge Latch', CS_EXTTRIGEN_POS_LATCH)
('Neg. Edge Latch', CS_EXTTRIGEN_NEG_LATCH)
('Pos. Edge Latch Once', CS_EXTTRIGEN_POS_LATCH_ONCE)
('Pos. Edge Latch Once', CS_EXTTRIGEN_NEG_LATCH_ONCE)

Commit (handle)

This function sends configuration parameters that are in the driver to the CompuScope system associated with handle. The parameters are sent to the driver via SetAcquisitionConfig. SetChannelConfig and SetTriggerConfig. The call to Commit sends these values to the hardware. If successful, the function returns CS_SUCCESS (1). Otherwise, a negative integer representing a CompuScope error code is returned.

StartCapture (handle)

Starts an acquisition on the CompuScope system associated with handle. If successful, the function returns CS_SUCCESS (1). Otherwise, a negative integer representing a CompuScope error code is returned.

ForceCapture (handle)

Emulates a trigger event on the CompuScope system associated with handle. If successful, the function returns CS_SUCCESS (1). Otherwise, a negative integer representing a CompuScope error code is returned.

AbortCapture (handle)

Aborts an acquisition or transfer on the CompuScope system associated with handle. If successful, the function returns CS_SUCCESS (1). Otherwise, a negative integer representing a CompuScope error code is returned.

GetStatus (handle)

Returns the status the current acquisition on the CompuScope system associated with handle. If the function fails, the return value will be a negative integer which represents a CompuScope error code. Otherwise, the return value will be one of the following values:

0: ACQ_STATUS_READY	ready for acquisition or transfer
1: ACQ_STATUS_WAIT_TRIGGER	waiting for trigger event
2: ACQ_STATUS_TRIGGERED	trigger has occurred, still busy acquiring
3: ACQ_STATUS_BUSY_TX	data transfer is in progress
4: ACQ_STATUS_BUSY_CALIB	auto-calibration is in progress

TransferData (handle, channel, mode, segment, start, length)

Transfers data from the memory of the CompuScope system identified by handle to computer memory. The other parameters are:

channel: The channel to transfer data from. Channel numbers begin from 1.
mode: The transfer mode. Can be:

0: regular memory transfer (TxMODE_DEFAULT). Regular transfer of unsigned 8-bit or signed 16-bit data.
2: time stamp (TxMODE_TIMESTAMP). Signed 64-bit transfer of timestamp data.
16: 32-bit data (TxMODE_DATA_32). Signed 32-bit transfer. Used with expert Signal Averaging firmware.
46: FFT data transfer (TxMODE_DATA_FFT). Unsigned 32-bit transfer used with expert FFT firmware.
Any other values will result in an invalid transfer mode error.

segment: the segment to be transferred. Segment numbers start at 1.
start: start address for transfer relative to the trigger address (trigger = 0)
length: size of requested data transfer in samples

If the function fails, the return value is a negative integer which represents a CompuScope error code. Otherwise, a tuple consisting of the following is returned:

tuple(0): a buffer of the requested data (the buffer is a numpy array)
tuple(1): the actual start address
tuple(2): the actual length

The actual start and actual length values may differ from the requested start and length if the data had to be adjusted for alignment purposes. The type of the values in the buffer depends on the CompuScope model and the transfer mode. For example, 8-bit CompuScopes will return unsigned 8-bit values, 12, 14 and 16-bit CompuScopes return signed 16-bit values.

GetStreamingBuffer (handle, cardIndex, size)

This function requests a contiguous buffer suitable for streaming from the driver for the CompuScope system identified with handle. The cardIndex parameter indicates the index of the board in a master / slave system. Board indices begin at 1. Use 1 for a single card system. The size parameter is the requested size of the buffer in bytes. If the function succeeds, the return value is a suitable buffer (from

numpy) for streaming. If the function fails, the return value is a negative error which represents a CompuScope error code.

FreeStreamingBuffer (handle, cardIndex, buffer)

This function frees a numpy buffer for CompuScope system identified by handle. The buffer must have been previously obtained by calling GetStreamingBuffer. The cardIndex parameter identifies for which board in a master / slave system the buffer was obtained. Use 1 for a single card system. Failure to free a buffer will result in memory leaks. If successful, the function returns CS_SUCCESS (1). Otherwise, a negative integer representing a CompuScope error code is returned.

TransferStreamingData (handle, cardIndex, buffer, transferSizeInSample)

Transfers streaming data from the CompuScope system associated with handle. The cardIndex parameter identifies which board in the system the transfer is for. For a single card system use 1. The buffer must be suitable for streaming and have been previously obtained by calling GetStreamingBuffer. The transferSizeInSamples is the requested transfer size in samples. If the function succeeds, the return value is CS_SUCCESS (1) and the data is returned in the buffer parameter. If the function fails, the return value is a negative integer which represents a CompuScope error code.

GetStreamingTransferStatus (handle, cardIndex, waitTimeout)

Returns the current DMA status of a streaming transfer for the board indentified by cardIndex in the CompuScope system indentified by handle. For a single card system, cardIndex should be set to 1. The waitTimeout (in milliseconds) parameter controls how to long to wait before returning. If it is 0, the function returns immediately with the status. If it is not 0, the function will wait until the current DMA transfer is completed or the waitTimeout value has expired before returning. If the function fails, a negative integer representing a CompuScope error code is returned. Otherwise, a tuple is returned containing the following values:

tuple(0): errorFlag – returns one or many error flags that may occur during streaming. Currently, STM_TRANSFER_ERROR_FIFOFULL is defined. This error indicates the application is not fast enough to transfer the data from on-board memory to PC RAM and that the FIFO is full, which results in data loss.

tuple(1): actualLength – holds the number of valid samples in the buffer once the DMA has completed.

tuple(2): endOfData – if this value is 1, all data from the current acquisition has been transferred. If it is 0, there is more to transfer. In infinite streaming mode this value is always 0 and actualLength is the requested size of the transfer.

ConvertToSigHeader (dict, comment, name)

This function converts a dict, along with an optional comment and name to a SIG file header suitable for loading into GageScope. The dict should contain the following values:

SampleRate	Sample rate at which the data was acquired
RecordStart	Starting address (in samples) of each segment in file relative to the trigger address
RecordLength	Length (in samples) of each segment in the file
RecordCount	Number of records saved in the file
SampleBits	Actual vertical resolution in bits of the stored signal
SampleSize	Actual sample size, in bytes, of the stored signal

SampleOffset	Actual sample offset for the stored signal. Used in conversion of raw ADC values to voltage values
SampleRes	Actual sample resolution of the stored signal. Used in conversion of raw ADC values to voltage values
Channel –	Channel number that is in the file. 1 corresponds to the first channel
InputRange	Channel full scale input range, in mV peak-to-peak
DcOffset	Channel DC offset, in mV
TimeStamp	Time stamp structure, elements are hour, minute, second, 100ths of a second. This is implemented as a dictionary with the following values: Hour, Minute, Second, Point1Second

If any fields are missing from the dictionary, the library will try to fill them in with appropriate values.

The comment and name parameters are optional.

If the function succeeds, the return value is a 512byte numpy array which can be used as a SIG file header. If the function fails, a negative integer is returned which represents a CompuScope error code.

GetErrorString (errorCode)

This function takes a CompuScope error code and returns a corresponding string explaining the error. If the function fails an error is returned, otherwise the return value is a string corresponding to the error code.

Support Functions in GageSupport.py

CalculateChannelIndexIncrement (mode, channel_count, board_count)

This routine calculates the channel increment when looping through the channels of a CompuScope system. For example, on an 8-channel board in dual channel mode, the channels will be numbered 1 and 5. The channel increment is 4. This routine works or both single and master / slave systems. The return value is the channel index increment to use.

LoadAcquisitionConfiguration (handle, iniFile)

This routine reads values from the INI file in the string iniFile and sets the CompuScope system identified by handle with the acquisition parameters found in the file. Any missing parameters will be set to the default of the system. If a complete path is not specified, the INI file should be in the same folder as the calling application. The format of the INI file can be found in the document INI_FILE_DESCRIPTION that comes with the CompuScope SDK. The python library configparser (ConfigParser for Python versions > 3) must be present in the system. The return value will be one of CS_SUCCESS (1), INI_FILE_MISSING (-1) or PARAMETERS_MISSING (-2).

LoadChannelConfiguration (handle, channel, iniFile)

This routine reads values from the INI file in the string iniFile and sets the CompuScope system identified by handle with the channel parameters found in the file. Each channel can have its own section in the INI file. Any missing parameters, or channels, will be set to the default of the system. If a complete path is not specified, the INI file should be in the same folder as the calling application. The format of the INI file can be found in the document INI_FILE_DESCRIPTION that comes with the CompuScope SDK. The python library configparser (ConfigParser for Python versions > 3) must be present in the system. The return value will be one of CS_SUCCESS (1), INI_FILE_MISSING (-1) or PARAMETERS_MISSING (-2).

LoadTriggerConfiguration (handle, trigger, iniFile)

This routine reads values from the INI file in the string iniFile and sets the CompuScope system identified by handle with the trigger parameters found in the file. Each trigger engine can have its own section in the INI file. Any missing parameters, or trigger engines, will be set to the default of the system. If a complete path is not specified, the INI file should be in the same folder as the calling application. The format of the INI file can be found in the document INI_FILE_DESCRIPTION that comes with the CompuScope SDK. The python library configparser (ConfigParser for Python versions > 3) must be present in the system. The return value will be one of CS_SUCCESS (1), INI_FILE_MISSING (-1) or PARAMETERS_MISSING (-2).

LoadApplicationConfiguration (iniFile)

This routine reads values from the INI file in the string iniFile and sets common application transfer parameters for the application. The values that can be set are StartPosition, TransferLength, SegmentStart, SegmentCount, PageSize, SaveFileName and SaveFileFormat. Any missing parameters will be set to default values. If a complete path is not specified, the INI file should be in the same folder as the calling application. The format of the INI file can be found in the document INI_FILE_DESCRIPTION that comes with the CompuScope SDK. The python library configparser (ConfigParser for Python versions > 3) must be present in the system.

SaveFile (filename, channel, buffer, format, stHeader)

This routine will save the data in buffer into a file named filename on the hard disk. If the filaneme does not include a path, the file will be saved to the curret directory. The channel should be the channel number of the CompuScope system that the data was captured from. It is used only for the SIG file header, but must be included for all formats. The file type is determined by format. The available formats are:

TYPE_DEC (0): ascii decimal values
TYPE_HEX (1): ascii hexadecimal values
TYPE_FLOAT (2): ascii voltages
TYPE_SIG (3): GageScope SIG file format
TYPE_BIN (4) raw binary data (no header)

The stHeader parameter is a dictionary that contains information about the capture. It is used to add a header to the file, to convert to volages and to convert various parameters to their GageScope SIG file equivalents. The dictionary fields are:

SampleRate Sample rate of the acquisition in Hz.

Start	Start address of the transfer in samples.
Length	Length of the transfer in samples.
SampleSize	Actual size of each sample in bytes.
SampleOffset	Actual sample offset of the signal. Used for conversion to voltages.
SampleRes	Actual sample resolution of data. Used for conversion to voltages.
SegmentCount	Number of records in the capture.
SegmentNumber	Number of records.
SampleBits	Actual vertical resolution of data in bits.
InputRange	Channel full scale input range, in mV peak-to-peak
DcOffset	Channel DC offset, in mV

The appropriate values for these fields can be obtained by calling GetAcquisitionConfig() and GetChannelconfig().

If the function is successful, a 1 is returned. If the function fails, the name of the file is returned for use in an error message.

Technical Support

We offer technical support for all our Software Development Kits.

In order to serve you better, we have created a web-based technical support system that is available to you 24 hours a day.

By utilizing the internet to the fullest, we are able to provide you better than ever technical support without increasing our costs, thereby allowing us to provide you the best possible product at the lowest possible price.

To obtain technical support, simply visit:

http://www.gage-applied.com/support/support-form.php

Please complete this form and submit it. Our form processing system will intelligently route your request to the Technical Support Specialist (TSS) most familiar with the intricacies of your product. This TSS will be in contact with you within 24 hours of form submittal.

In the odd case that you have problems submitting the form on our web site, please e-mail us at

tech-support@gage-applied.com

As opposed to automatic routing of technical support requests originating from the GaGe web site, support requests received via e-mail or telephone calls are routed manually by our staff. Providing you with high quality support may take an average of 2 to 3 days if you do not use the web-based technical support system.

Please note that Technical Support Requests received

via e-mail or by telephone will take an average of 2 to 3 days to process.

It is faster to use the web site!

When calling for support we ask that you have the following information available:

1. Version and type of your CompuScope SDK and drivers.
   (The version numbers are indicated in the About CD screen of the CompuScope CD. Version numbers can also be obtained by looking in the appropriate README.TXT files)

2. Type, version and memory depth of your CompuScope card.

3. Type and version of your operating system.

4. Type and speed of your computer and bus.

5. If possible, the file saved from the Information tab of the CompuScope Manager utility.

6. Any extra hardware peripherals (example: RAID Controller, DSP board, etc.)

7. Were you able to reproduce the problem with standalone Gage Software (e.g. GageScope, CsTest)?