GaGe

CompuScope Software Development Kit (SDK) for MATLAB
for Windows

User's Guide

CompuScope Driver Version 4.10+

For MATLAB 6.5+

Ninth Edition (July 2008)

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: www.gage-applied.com/support/support_form.php

Table of Contents

Preface

This manual is meant to serve as an aid to engineers using the CompuScope series of high-performance data acquisition cards in the MATLAB 6.5+ for Windows environment.

The CompuScope SDK for MATLAB for Windows supports all GaGe CompuScope cards – PCI and CompactPCI/PXI. Specific hardware features that are available in the SDK sample programs, however, may not be supported by your CompuScope model. Please refer to the CompuScope Hardware Manual for information specific to your CompuScope card in order to determine the capabilities of your CompuScope model.

Throughout this manual, it is assumed that you are familiar with the MATLAB programming environment. If you do not feel comfortable with MATLAB, it is highly recommended that you go through the Getting Started with MATLAB$^®$ manual supplied to you by The MathWorks, Inc. before starting any program development for the CompuScope cards.

It is also assumed that you are familiar with PCs and Microsoft Windows and that you have correctly installed the CompuScope Windows drivers.

This manual will use the terms "CompuScope SDK for MATLAB for Windows" and "CompuScope MATLAB SDK" interchangeably.

Please note that this manual is not intended as a reference for any software other than the CompuScope SDKs for MATLAB for Windows. If you did not receive the correct guide, please contact the factory for a replacement.

Please note that, although 64-bit Gage CompuScope drivers are available for 64-bit Windows Operating systems, the CompuScope MATLAB SDK does not support 64-bit MATLAB.

To maintain the accuracy of the information contained herein, we reserve the right to make changes to this manual from time to time.

Chapter 1: Installation of CompuScope MATLAB SDK

If you purchased the CompuScope MATLAB SDK you will have been shipped a software key that allows installation of the SDK from the GaGe CompuScope CD. Simply select the installation of the CompuScope MATLAB SDK from the CompuScope CD and enter the software key when prompted.

By default, the CompuScope MATLAB SDK will install itself in:
O/S system drive:\Program Files\Gage\CompuScope\CompuScope MATLAB SDK. It is recommended that you use the default installation location. The CompuScope MATLAB SDK will create three sub-folders called *Main*, *CsMl* and *Adv.* These folders respectively contain Main M file programming examples, CsMl M files that are the building block function calls from which the Main M files are constructed and advanced M files for special non-standard CompuScope functionality.

If you require more detailed installation instructions, please refer to the GaGe CompuScope Startup Guide, which was shipped with your order.

Please note, if you had MATLAB installed when you installed the CompuScope MATLAB SDK, then the required path settings will be automatically updated upon installation of the SDK. If, however, if you did not have MATLAB installed when you installed the CompuScope MATLAB SDK, then you must update the MATLAB path settings manually after installation of MATLAB. To do this, append AddPath.m to Startup.m in the MATLAB directory. If Startup.m does not exist, copy AddPath.m into the MATLAB directory and change the filename to Startup.m. AddPath.m is installed within the Main MATLAB folder of the MATLAB SDK

Chapter 2: Overview of CompuScope MATLAB SDK

Structure of CompuScope MATLAB SDK

The overall structure of the CompuScope MATLAB SDK and its relation to the GaGe CompuScope Hardware is best described from the bottom up with reference to the diagram below.

```
┌─────────────────────┐
│   Main MATLAB SDK   │
│       M File        │
└─────────────────────┘
           ↕
┌─────────────────────┐
│        CsMl         │
│       M File        │
└─────────────────────┘
           ↕
┌─────────────────────┐
│      CsMl.DLL       │
└─────────────────────┘
           ↕
┌─────────────────────┐
│      CSSSM.DLL      │
└─────────────────────┘
```

Windows Application Level

Windows Kernel Level

```
           ↕
┌─────────────────────┐
│ GaGe CompuScope drivers │
└─────────────────────┘
           ↕
┌─────────────────────┐
│ GaGe CompuScope Hardware │
└─────────────────────┘
```
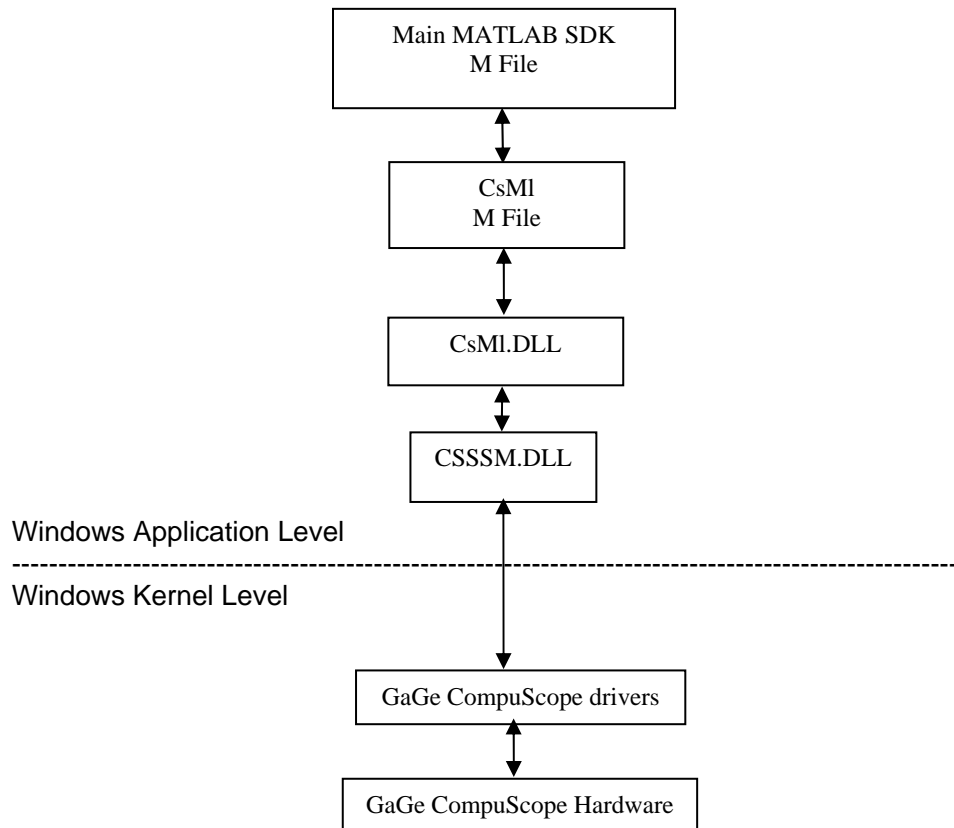
At the lowest level is the CompuScope hardware, which is installed within a slot that is connected to the host PC. The CompuScope hardware is directly controlled by the CompuScope Windows drivers. The drivers reside at the Windows Kernel level, which allows direct low-level access to CompuScope hardware registers and to physical PC RAM.

The Kernel-level drivers communicate with Windows Applications through a Dynamically Linked Library (DLL) called CSSSM.DLL. Communications through CSSSM.DLL use the CompuScope Applications Programming Interface (API), which is the set of C subroutine calls that allows control of all CompuScope functionality.

Above CSSSM.DLL is an intermediate DLL called CsMl.DLL. This DLL contains a single MATLAB MEX function called CsMl() that is used to control GaGe CompuScope hardware from MATLAB. Different index arguments to CsMl() from MATLAB allow different operations to be performed on the CompuScope hardware.

For convenience, all subroutine calls to CsMl() are wrapped within CsMl M files with descriptive names, which makes the MATLAB coding more readable and removes the requirement of keeping track of the required index values. All CsMl M file names have the form CsMl_Xxx.m, where Xxx describes its functionality. Most users should find it unnecessary to modify CsMl M file code so that the CsMl M files become the building blocks from which users may construct any required MATLAB M files for their application.

The Main M files are top-level application M files that can be executed directly under MATLAB. Each Main MATLAB M file illustrates usage of the CompuScope hardware in different operating modes. Main M files are constructed using calls to CsMl M files. Users should not attempt to call CsMl M files directly from the MATLAB command line, since these M files must be called in the correct order in some cases. The Main MATLAB M files are intended as convenient starting points for CompuScope users to develop their own MATLAB applications.

CompuScope Systems

A CompuScope system is defined as a single CompuScope digitizer board or a group of CompuScope boards configured as a Master/Slave CompuScope system. A Master/Slave CompuScope system samples and triggers simultaneously on all channels and is considered to be one multi-channel CompuScope system. For instance, a Master/Slave system composed of four CompuScope two-channel boards will be considered as a single CompuScope system with eight available input channels.

By structuring the drivers to consider CompuScope systems, a single PC can be equipped with almost any imaginable combinations of CompuScope hardware. For instance, a PC could be equipped with two separate Master/Slave systems of four channels each and then an additional single CompuScope system for a total of three CompuScope systems.

CompuScope systems are addressed from MATLAB by first acquiring a *handle* for the system, which is an integer value that uniquely describes the CompuScope system. After usage of the system is complete, the user must release the handle so that it is free for usage by other processes. By obtaining handles for different systems, a single MATLAB M file may simultaneously operate different CompuScope systems. Alternately, separate MATLAB M files may operate independently and simultaneously by calling handles for separate CompuScope systems. Different M files may even access the same hardware as long as one M file frees the system handle before the other M file obtains it, since different applications may not simultaneously access the same CompuScope system.

While most Main example M files access only a single CompuScope system, understanding the CompuScope system structure easily allows users to extend these Main M files to multiple system operation.

Overview of CompuScope Main M files

The Main M files are intended to be convenient starting points around which users can develop customized MATLAB software for their digitizer application. Users can construct more complex M files themselves by using the existing Main M files as a guide to combining their functionalities.

The basic algorithm for all of the Main M files is the following:

1. *Initialize the CompuScope driver and the CompuScope hardware.*
2. *Obtain the handle to the first available CompuScope system in the PC.*
3. *Pass the desired configuration settings to the driver with a call to Setup.m*
4. *Pass the configured settings to the CompuScope hardware using the CsMl_Commit() subroutine.*
5. *Start the CompuScope hardware to digitize data into its on-board memory and await a trigger event.*
6. *Continuously check to see if the CompuScope hardware has finished capturing the current record.*
7. *Download the data of interest from the CompuScope hardware to MATLAB array variables.*
8. *Store the acquired data to a .DAT ASCII data file.*

A list of all CompuScope Main M files is given below with a brief description of the functionalities.

GageSimple.m: A simple M file used to verify correct CompuScope, driver and MATLAB SDK operation. No adjustments are possible and default CompuScope settings are used.

GageAcquire.m: The simplest M file with full configuration control of CompuScope settings that illustrates usage of CompuScope Single Record Mode. An error occurs upon entry of invalid settings.

GageCoerce.m: An M file that is just like GageAcquire.m, except that invalid settings are coerced to the closest valid values.

GageMultipleRecord.m: An M file that acquires and stores captured records when operating a CompuScope system in Multiple Record Mode.

GageDeepAcquisition.m: An M file that acquires and manages large data captures from CompuScope hardware (> 16 MB).

GageComplexTrigger.m: An M file that illustrates usage of complex triggering using multiple on-board trigger engines, if available.

GageMultipleSystems.m: An M file that operates two independent CompuScope systems, each with their own settings and output .DAT files.

DisplayData.m: An M file that plots data from .DAT files acquired in Single Record Mode.

ReadMrDataFiles.m: An M file that plots data from .DAT files acquired in Multiple Record Mode.

All acquired data are stored in .DAT ASCII data files. These files contain a file header that is between two lines of minus signs (-). The acquired data are then listed as a single column of data. Data are stored in volts by default but may alternately be stored as raw ADC data. The DAT file format is described in more detail in the documentation provided with the GaGe File Converter utility, which allows conversion between ASCII DAT files and GageScope SIG binary files.

Not all possible CompuScope operation configurations are covered by the Main M files. For instance, there is an M file that handles deep acquisitions and an M file that handles acquisitions from two CompuScope systems. However, there is not an M file that handles deep acquisitions from two CompuScope systems. For both these functionalities, the user must study GageDeepAcquisition.m and GageMultipleSystems.m and then appropriately combine them to create an M file that meets the requirement.

All Main M files should always be executed in their entirety in order to correctly free CompuScope handles so that other processes will be unable to use them. Locked handles may be freed from MATLAB, however, by invoking the CsMl_FreeSystem(handle) command if the handle value is known or by calling CsMl_FreeAllSystems, which will abort all CompuScope operations and free all CompuScope system handles. Finally, locked handles may be freed from outside of MATLAB by doing a "Refresh" from the CompuScope Manager utility after exiting MATLAB.

Overview of CompuScope CsMl M files

All Main M CompuScope example files are constructed from CsMl M files, which are easy-to-use building blocks that wrap function calls to the intermediate CsMl.dll. A MATLAB user is able to develop a custom MATLAB M file for their requirement using only the CsMl M files, with no modifications to them. All CsMl M files are documented later in this manual.

All CsMl M files have names of the form CsMl_Xxx.m, where xxx describes the functionality. The M files may be called with arguments and return values using the general form:

[return_value1, return_value2,…] = CsMl_Xxx(arg1, arg2,…)

The first argument to the function is usually the *handle* to the CompuScope system being addressed. Generally, not all available return values need to be included in the call but will not be assigned unless they are included. Some return values, however are required. The documentation for each call will tell which return values are required and which can be ignored. Generally speaking, if there is one return value it can be ignored. If there is more than return value, only the rightmost value can be ignored.

A brief description of any CsMl_Xxx M file may be obtained simply by typing "help CsMl_Xxx" at the MATLAB command prompt.

Chapter 3: Detailed Description of CompuScope *Main M files*

Setup.m:

Within all Main M file programming examples, values of configuration parameters are assigned within an M file called Setup.m, which is called by every Main MATLAB M file.

Setup.m is called with an argument of the *handle* of the CompuScope system to be configured. Next, Setup.m calls CsMl_GetSystemInfo.m to obtain information about the CompuScope system.

Setup.m proceeds to assign values for the three main parameter sets: the acquisition parameters, the channel parameters and the trigger parameters. Each parameter set has its own associated variable structure (*acq, chan* and *trig*, respectively) that are used as arguments by the Configuration M files (CsMl_ConfigureAcquisition.m, CsMl_ConfigureChannel.m and CsMl_ConfigureTrigger.m, respectively). The Configuration functions are used to assign parameter values within the CompuScope driver. However, the parameter setting values are not actually sent to the CompuScope hardware until CsMl_Commit is invoked.

Descriptions of all configuration parameters are listed below. Throughout the parameter assignments, the CsMl_Translate M file is used. Using a context field, this function translates descriptive strings into parameter index values for convenience and code readability. For instance, instead of remembering that the CompuScope mode index values for "Single", "Dual", "Quad" and "Octal" modes are 1, 2, 4 and 8, the user can simply translate these strings using CsMl_Translate.

If any of the configuration parameters are not assigned, their values will be assigned to default values by the driver. The *Channel* field within the *chan* structure and the *Trigger* field within the *trig* structure, however, are mandatory.

Acquisition Parameters

acq.SampleRate	The sampling rate in Hz
acq.ExtClock	A flag that activates external clocking when non-zero
acq.Mode	The CompuScope mode, "Single", "Dual", "Quad" or "Octal"
acq.SegmentCount	The number of segments to be acquired
acq.Depth	The post-trigger depth
acq.SegmentSize	The size of memory allocated for the segment
acq.TriggerTimeout	The trigger time-out value in microseconds
acq.TriggerDelay	The trigger delay value in samples
acq.TriggerHoldoff	The trigger holdoff value in samples
acq.TimeStampConfig	A flag that, when non-zero, resets the Time-Stamp counter at the beginning of an acquisition

Channel Parameters

The *chan* variable may be either a structure or an array of structures. MATLAB will handle them either way. If *chan* is a structure, it may be referenced as either *chan(1).Channel* or *chan.Channel*. Either way is correct. If *chan* is an array of structures, each element contains the structure for channel number i, where i begins at 1.

chan(i).Channel	The channel number, beginning with 1
chan(i).Coupling	The input coupling (AC or DC)
chan(i).DiffInput	A flag that, when non-zero, sets the channel to differential coupling, if available.
chan(i).InputRange	The channel full scale input range in millivolts. For instance, set to 2000 for the +/- 1 Volt input range
chan(i).Impedance	The channel terminating input impedance in Ohms
chan(i).DcOffset	The channel input DC offset in millivolts
chan(i).DirectAdc	A flag that, when non-zero, sets the channel to Direct-to-ADC input coupling, if available.

Trigger Parameters

The *trig* variable may be either a structure or an array of structures. MATLAB will handle them either way. If *trig* is a structure, it may be referenced as either *trig(1).Trigger* or *trig.Trigger*. Either way is correct. If *trig* is an array of structures, each element contains the structure for each trigger engine i, where i begins at 1. The use of the *trig* variable as an array of structures is illustrated within GageComplexTrigger.m.

trig.Trigger	The trigger engine number, beginning with 1
trig.Slope	The trigger slope (Positive or Negative)
trig.Level	The trigger level as a percentage (-100 to 100) of the trigger source input range
trig.Source	The trigger source (A Trigger Channel Source (1, 2, …), External or Disable)
trig.ExtCoupling	The input coupling of the external trigger input (AC or DC)
trig.ExtRange	The external trigger full scale input range in millivolts. For instance, set to 10000 for the +/- 5 Volt input range

GageSimple.m:

GageSimple.m is not intended as a starting point for customer applications but is a simple test program to confirm correct operation of the CompuScope hardware, Windows drivers, and MATLAB SDK.

GageSimple.m grabs the *handle* to the first available CompuScope system and does a single acquisition using the driver defaults for all configuration parameters. The driver defaults are a set of default configuration parameters that are tailored to each CompuScope hardware model. After acquisition, acquired data for Channel 1 are displayed in a simple MATLAB plot window.

This M file operates only on a single CompuScope card. In the event of a multi-CompuScope system, the M file only addresses the first card in the first system.

Again, GageSimple.m should only be used to test hardware and software integrity and should not be developed into a custom application M file, since there are better Main M files for this purpose. Simply check that no errors occurred during execution of GageSimple.m and that the MATLAB plot displays the expected waveform data in order to confirm that the MATLAB SDK is operating correctly.

GageAcquire.m

GageAcquire.m is the Main M file for Single Record capture from a single CompuScope system. In the event of a multiple CompuScope system, the M file only operates the first card in the system.

The M file first initializes the drivers using CsMl_Initialize, then obtains the *handle* to the first available CompuScope system using CsMl_GetSystemInfo. Next, Setup.m is called, which assigns values to each configuration parameter and passes these values to the drivers.

After Setup.m, CsMl_Commit is called, which actually passes the parameter values that have been set in the driver to the CompuScope hardware. Invalid settings cause an error and an error message is displayed. Users who prefer the M file to continue operating upon invalid setting entry should use GageCoerce.m, which will coerce invalid entries to valid entries and continue running.

Next, the *transfer* structure is filled with parameters that will determine how the data are transferred to MATLAB from CompuScope memory after acquisition. Parameters within the transfer structure are:

transfer.Segment   The segment number to transfer, starting from 1;

transfer.Start   The point from which to start data transfer. Zero indicates the trigger address. Enter a negative value in order to download pre-trigger data;

transfer.Length   The number of points to transfer, starting from the Start point.

In GageAcquire.m, the transfer length is set equal to the *SegmentSize*, so that all segment data are returned. The transfer start is set equal to the *Trigger Holdoff*, so that all valid pre-trigger data are transferred. These two values (*SegmentSize* and *Holdoff*) were set previously within Setup.m but are obtained using CsMl_QueryAcquisition.

Next, the acquisition is initiated on the CompuScope system using CsMl_Capture. The CompuScope hardware then begins acquiring pre-trigger data and awaiting a trigger event. The acquisition terminates after the trigger event occurs and the CompuScope system has acquired the requested amount of pre-trigger data. The state of the CompuScope system is queried using CsMl_QueryStatus(handle) until the acquisition has completed.

Once the acquisition is complete, data are transferred to MATLAB using CsMl_Transfer. Within GageAcquire.m, CsMl_Transfer is configured to first transfer the raw integer ADC data and then to convert these data to voltage values. These converted voltage values are then returned by CsMl_Transfer. For faster repetitive capture performance, CsMl_Transfer may alternately be configured to transfer raw ADC code data as integers.

Mathematical operations may not be performed on these integer values, however, without first converting them to double values.

After transfer, the data are stored to ASCII DAT files with the appropriate header information. The DAT files have names of the form Acquire_CHx.dat, where x is the channel number. Waveform data are also displayed in a simple MATLAB plot window. While GageAcquire.m simply stores and displays waveform data, the end of the program may be easily modified by the user to instead manage the data according to the requirements of the application. For instance, waveform data may be immediately analyzed in real-time using MATLAB's extensive math libraries.

GageCoerce.m

GageCoerce.m is identical to GageAcquire.m, except that in the event of invalid configuration setting entries, the M file will not stop with an error message. Instead of stopping, if invalid configuration settings are entered, the M file will coerce the settings to values that are available on the current CompuScope system. When coercion occurs, a message is printed which indicates that a parameter was coerced and the current acquisition, channel, and trigger parameters are printed to the console. Current active parameter settings may be queried using the Query CsMl files. Waveform data are stored in DAT files named Coerce_CHx.dat, where x is the channel number.

GageMultipleRecord.m

GageMultipleRecord.m is the Main M file for Multiple Record capture from a single CompuScope system. Multiple Record mode allows multiple waveforms to be rapidly acquired and stacked in on-board CompuScope memory. For instance, a CompuScope card with 32 MegaSamples of on-board memory may acquire 16,000 records of 2,000 samples each in Multiple Record Mode. (Strictly speaking, depending on the CompuScope model, this relation may not exactly apply because of slight inter-record padding requirements.)

Between successive acquisitions, the CompuScope acquisition engine is re-armed by the hardware with no CPU interaction required. Consequently, in Multiple Record Mode, a CompuScope card is capable of capturing bursts of trigger that repeat at rates of 100,000 triggers per second and more.

The number of records or segments to acquire is set using the *SegmentCount* parameter. This is set to 1 within the single Setup.m file that is provided and so must be raised for a large record count. If *SegmentCount* exceeds the maximum possible number of records, which is roughly equal to the available acquisition memory divided by the Depth, then an error will occur. (If coercion had been used in this case, then the actual number of records would have been set equal to its maximum possible value.)

The *SegmentSize* parameter is used to control the size of each Multiple Record segment. For newer CompuScope hardware, this may be set larger than the *Depth* parameter so that pre-trigger data may be acquired. Older CompuScope models do not support pre-trigger data in Multiple Record Mode. Consequently, setting *SegmentSize* to a value different from *Depth* will cause an error. Please refer to your CompuScope Hardware Manual for information specific to your CompuScope model's capabilities. For a more

detailed discussion of the relationship between *SegmentSize* and *Depth*, please see Chapter 5: Special Topics.

When executed, GageMultipleRecord.m first initializes and configures the CompuScope system, as in GageAcquire.m. Next, a single Multiple Record acquisition is performed.

After acquisition, the M file goes into a loop that downloads each Multiple Record from each active channel and stores the record data to ASCII data files.

Newer CompuScope models such as the CS14200, CS14105 and CS12400, support Time-Stamping. This feature registers an on-board counter value for each Multiple Record trigger event that indicates its time of occurrence. Time 0 is the time at which the time-stamping counter was last reset. The Time-Stamp counter is reset to zero at the beginning of GageMultipleRecord.m. If the CompuScope system is equipped with on-board Time-Stamping, GageMultipleRecord.m downloads Time Stamp data and stores each Time Stamp in the corresponding ASCII DAT file. Otherwise, a 0 is stored as the Time Stamp value.

GageDeepAcquisition.m

GageDeepAcquisition.m is the Main M file for large acquisitions from a single CompuScope system. The definition of large varies with system configuration but is roughly about 16 MegaBytes. For small acquisitions, MATLAB is fully capable of downloading the entire acquisition into PC RAM. For larger acquisitions, the host PC will begin to use "virtual RAM", which is a section of the hard drive that Windows treats like PC RAM. Usage of virtual RAM causes the hard drive to spin and slows down execution. For large enough data sets, MATLAB will simply refuse to download the data at all.

In order to avoid trying to keep large data sets in MATLAB at one time, GageDeepAcquisition.m manages deep acquisitions by dividing them up into more manageable data *pages,* whose size is denoted by the *chunksize* variable. The M file does initialization, configuration setting and acquisitions as usual. After the acquisition, however, data are downloaded in pages of size *chunksize.* Each page is stored in its own ASCII DAT file called DeepAcquisition_CHxx-yyyy.dat, where xx is the channel number and yyyy is the page number. One complete data file for the entire acquisition may easily be created by removing the header for each DAT file and combining them in order by page number.

The user may easily replace data storage with analysis and/or display to suit their requirements. Users should always use a similar data paging scheme for downloading large amounts of data and should never attempt to download large amounts of data at one time.

GageComplexTriggerAcquire.m

GageComplexTriggerAcquire.m is a Main M file that illustrates complex triggering. Some CompuScope models are equipped with two on-board trigger engines that can be used for complex triggering. On these models, the two engines can be configured independently and their outputs are Boolean ORed together so that either engine may cause a trigger event. For simple triggering, the second engine is disabled.

The *trig* structure array allows the separate configuration of each trigger engine. For instance, by setting the two engine sources to Channel 1and Channel 2, the user may configure the system to trigger on a pulse that occurs on either channel. Alternately, by setting both engine sources to Channel 1 but selecting different levels and slopes for each, the user may configure the system to do windowed triggering, where the system triggers if the input level leaves a specified voltage range.

GageComplexTriggerAcquire.m uses the *trig* structure as an array, where each element controls a separate trigger engine. Otherwise, the M file is just like GageAcquire.m.

GageMultipleSystems.m:

GageMultipleSystems.m is the Main M file for Multiple Independent CompuScope systems. The M file begins by obtaining the number of CompuScope systems within the host PC. It then obtains a handle to each of these systems. The GageMultipleSystems.m then proceeds to execute the same order of operations as GageAcquire.m, except each operation is within a loop that successively operates on each CompuScope system.

A user can extend the logic illustrated within GageMultipleSystems.m to achieve any acquisition sequence from Multiple Independent CompuScope systems. For instance, the user may choose to separate the code that controls each CompuScope system so that acquisitions are performed sequentially on each system. Alternately, the user may choose to do a deep memory acquisition on one system and a Multiple Record acquisition on another. This is done by appropriately merging GageMultipleRecord.m and GageDeepAcquisition.m using separate CompuScope *handles* for the two code sections.

DisplayData.m

DisplayData.m reads CompuScope data from an ASCII DAT file and plots them in a MATLAB plot window. The M file may be called from within another M file.

ReadMrDataFiles.m

ReadMrDataFiles.m reads CompuScope data from an ASCII DAT file created from a Multiple Record acquisition and plots them in a MATLAB mesh plot. The M file may be called from within another M file.

Digital Input CompuScope cards

All MATLAB SDK programs may be used to perform acquisitions from digital input CompuScope cards, such as the CS3200 and CS3200C. Parameters specific to digital input cards may be set by using the CsMl_ConfigureXxx M files with the appropriate

---

modifiers. By using these special parameter settings for the digital input CompuScope cards, a user may operate any one of the Main M files

The CS3200 and CS3200C allow three digital input sample width modes for optimal memory usage: 8-bit, 16-bit and 32-bit. These modes may be selected *acq.Mode* together with the CsMl_Translate.m within Setup.m and using the following string values:

Single   to activate 8-bit digital input mode
Dual   to activate 16-bit digital input mode
Quad   to activate 32-bit digital input mode

The CS3200 and CS3200C provide different digital input signal level protocols, depending on the model. Different digital input protocols are selected by setting appropriate values of *chan(i).InputRange* within Setup.m and using the following input range values:

  2500   to activate CMOS digital voltage levels
  1500   to activate TTL digital voltage levels
 -2000   to activate ECL digital voltage levels
  3000   to activate PECL digital voltage levels
     0   to activate LVDS digital voltage levels

The CS3200 and CS3200C allow inversion of the sampling clock so that digital samples are acquired on the falling edge of the clock signal rather than on the rising edge. In order to select the falling edge, the desired *acq.Mode* must be ORed with the hexadecimal value 0x800. Consequently, strings with the CsMl_Translate.m file may not be used and *acq.Mode* must be assigned to the following numerical values: .

2049   to activate 8-bit digital input mode with clock inversion
2050   to activate 16-bit digital input mode with clock inversion
2052   to activate 32-bit digital input mode with clock inversion

Advanced Main M files

The MATLAB SDK may contain "Advanced" M files in addition to the documented Main M files. Usage of some of these files may require special CompuScope hardware options, on-board firmware processing images or a special driver version. These M files are located in the adv folder and are provided as-is with limited documentation in the form of an accompanying explanatory text file.

Chapter 4: Overview of CsMl M Files

Please refer to the MATLAB SDK help for a more detailed description of CsMl M file operation. At the MATLAB command prompt, type "help" and the name of the CsMl M file. (e.g. help CsMl_AbortCapture)

CsMl_AbortCapture
CsMl_AbortCapture aborts an acquisition on a CompuScope system.

CsMl_Capture
CsMl_Capture begins an acquisition using the current acquisition parameters on a CompuScope system.

CsMl_Commit
CsMl_Commit sends the acquisition, channel and trigger parameters that are in the driver to a CompuScope system.

CsMl_ConfigureAcquisition
CsMl_ConfigureAcquisition sets the acquisition parameters for a CompuScope system.

CsMl_ConfigureChannel
CsMl_ConfigureChannel sets the channel parameters for a CompuScope system.

CsMl_ConfigureFir
CsMl_ConfigureFir sets the Finite Impulse Response parameters for a CompuScope system.

Note that the CompuScope system must have an optional FIR image available and loaded for this function to work.

CsMl_ConfigureTrigger
CsMl_ConfigureTrigger sets the trigger parameters for a CompuScope system.

CsMl_ErrorHandler
CsMl_ErrorHandler processes errors from other *CsMl_Xxx* calls.

CsMl_ForceCapture
CsMl_ForceCapture forces a trigger event to occur on a CompuScope system.

CsMl_FreeAllSystems

CsMl_FreeAllSystems aborts any acquisitions and frees all CompuScope systems that are currently in use.

CsMl_FreeSystem

CsMl_FreeSystem frees a CompuScope system.

Once a system is freed, it may be used by other applications.

CsMl_GetErrorString

CsMl_GetErrorString translates the error code returned by other *CsMl_Xxx* functions into a descriptive error message.

CsMl_GetExtendedOptions

Extended options are optional programmed firmware images. *CsMl_GetExtendedOptions* returns information about the optional programmed firmware images available on a CompuScope system.

CsMl_GetSystem

CsMl_GetSystem returns a *handle* that uniquely identifies a CompuScope system.

CsMl_GetSystemCaps

CsMl_GetSystemCaps returns information about the capabilities of a CompuScope system. While complicating a MATLAB program, this function may be used to allow the program to support any possible configuration of CompuScope hardware. MATLAB SDK Main M files do not use this function.

CsMl_GetSystemCaps

CsMl_GetSystemInfo returns static information about a CompuScope system.

CsMl_Initialize

CsMl_Initialize initializes a CompuScope system and is typically the first CompuScope call made in a program.

CsMl_QueryAcquisition

CsMl_QueryAcquisition returns information about a CompuScope system.

CsMl_QueryChannel

CsMl_QueryChannel returns information about the specified channel in a CompuScope system.

CsMl_QueryStatus

CsMl_QueryStatus queries the driver for the current acquisition status of a CompuScope system.

CsMl_QueryTrigger

CsMl_QueryTrigger returns information about the specified trigger in a CompuScope system.

CsMl_ReadDataFile

CsMl_ReadDataFile reads data from ASCII DAT files and adjusts the data to account for alignment issues using the *Start address* and *Data length* fields saved in the ASCII file header.

CsMl_ResetTimeStamp

CsMl_ResetTimeStamp resets the time stamp counter associated with a CompuScope system.

CsMl_SaveFile

CsMl_SaveFile saves a buffer of data acquired by a CompuScope system to an ASCII DAT file.

CsMl_Transfer

CsMl_Transfer transfers data for one channel from a CompuScope system

CsMl_Translate

CsMl_Translate is provided for convenience and allows conversion of descriptive string into constant values that are required by the driver, thereby not requiring the user to look up the values in tables.

Chapter 5: Special Topics

Converting from CompuScope ADC Code to Voltages

SDK sample programs are configured to save or display CompuScope waveform data that have been scaled so that the sample values are in Volts. The user may want to bypass the voltage conversion step, however, in order to achieve the best repetitive capture performance. Voltage conversion may then be done at leisure in post-processing.

Raw ADC code waveform data may be converted to voltage data for all CompuScope models using the following equation:

$$Voltage = \frac{Offset - ADC\_Code}{Resolution} \times \frac{(Full\ Scale\ Input\ Voltage)}{2} + DC\_Offset$$

The *Offset* and *Resolution* for the CompuScope system may be obtained programmatically using various SDK tools. The DC offset settings, supported only on newer CompuScope models, must be added.

For instance, for the CompuScope 82G, *Offset*=127 and *Resolution*=128. Let us assume that the user has selected the +/-1 Volt Input Range, for a Full Scale Input Range of 2 Volts. Let us further assume that the user has applied a 200 mV offset. For this example, therefore, the voltage conversion equation becomes:

$$Voltage = \frac{127 - ADC\_Code}{128} \times 1\ \text{Volts} + 0.2\ \text{Volts}$$

Depth and Segment Size

On all CompuScope models, on-board acquisition memory is arranged as a circular buffer for Single Record acquisitions. This means that when the CompuScope is storing acquired data and it reaches the end of the memory, the memory counter rolls over. Data storage wraps around the memory and starts digitizing into the beginning of memory. The end of an acquisition is always initiated by the trigger event, after which the requested number of post-trigger data points is acquired and then the acquisition terminates. This circular memory architecture allows the CompuScope to accumulate an amount of pre-trigger data of up to the amount of CompuScope acquisition memory per channel, less the amount of post-trigger data.

Newer CompuScope models, such as the CS82G, CS12400, CS14200 and CS14105, allow for the accumulation of pre-trigger data in Multiple Record Mode. On these models, memory is divided into multiple circular buffers, the number of which is equal to the number of records specified. The Segment Size control allows the user to specify the size of each circular buffer. This way, the user may select the amount of memory reserved for pre-trigger data accumulation. For instance, if the user selects a post-trigger Depth of 2048 and a Multiple Record Segment Size of 8192, then up to 6144 Samples of pre-trigger data may be acquired. Increasing the Segment Size beyond the Depth will

---

accordingly reduce the maximum possible number of Multiple Records.

Only newer CompuScope models such as the CS82G, CS12400, CS14200 and CS14105, support pre-trigger Multiple Record Mode. On older CompuScope models without pre-trigger data in Multiple Record Mode, the Segment Size setting will have no effect on a Multiple Record acquisition. The value is ignored and is forced to be equal to the Depth by the drivers.

Trigger Holdoff

Trigger Holdoff is a feature that is useful for ensuring the accumulation of a specified amount of pre-trigger data. The Trigger Holdoff setting specifies the amount of time, in Samples during which the CompuScope hardware will ignore trigger events after acquisition has begun and pre-trigger data are being acquired.

Without a non-zero Trigger Holdoff value, there is no guarantee that a given number of pre-trigger samples will be acquired. This is because a trigger event may occur immediately after acquisition, leading to a very small number of pre-trigger points. By ignoring trigger events for a time equal to the specified Trigger Holdoff value, the accumulation of a number of pre-trigger points equal to the Trigger Holdoff setting is guaranteed.

The downside of ensuring pre-trigger data with a non-zero Trigger Holdoff value is that triggers are ignored, so that important trigger events may be missed. For instance, in lightning monitoring applications, researchers usually want to acquire pre-trigger data. These data give information about signal behavior immediately preceding the lightning strike, which triggers the CompuScope hardware. Lightning strikes may occur very close together in time, however, and missing a lightning pulse is much worse than missing pre-trigger data. Consequently, lightning testers should not use Trigger Holdoff but should simply accept acquisitions of only the amount of pre-trigger data that naturally occur between triggers. Generally speaking, therefore, the user must decide whether to use Trigger Holdoff, based on the application.

Trigger Delay

New-generation PCI CompuScope models such as the CS14200, CS14105 and CS12400, support a feature called Trigger Delay. This feature is useful for situations where the signal portion of interest occurs long after the trigger event.

Normally, with a Trigger Delay of zero, the trigger event activates count-down of the post-trigger depth counter, which was preloaded with the post-trigger depth. The counter is decremented by one count for each sample acquired after the trigger event. When the counter value reaches 0, the acquisition is terminated. In this way, the CompuScope acquires a number of samples equal to the depth after the trigger event.

A non-zero Trigger Delay value is used to delay the beginning of the countdown of the post-trigger depth counter. The Trigger Delay value sets the number of samples that the CompuScope hardware will wait after the trigger event occurs before beginning the count-down of the depth counters. The SegmentSize may be set equal to the depth so that the CompuScope hardware need not waste memory by storing data that are not of interest.


As an example, consider a signal where the feature of interest is 20,000 samples long but begins 100,000 samples after the trigger event. In this case, the SegmentSize and Depth should be set to 20,000. The Trigger Delay value should be set to 100,000. Without the Trigger Delay feature, the user would be forced to set the Depth to 120,000 Samples and waste 100,000 Samples of memory, even though only the last 20,000 were downloaded. Using the Trigger Delay feature, however, only the data of interest are retained in CompuScope memory.

CompuScope Acquisition Timing Diagram

The timing diagram below is provided as an aid for understanding the timing of events during the acquisition of a segment or record. The pseudo—signals are indicated as HIGH when the labeled function is active.

A CompuScope system always acquires from the beginning of the Segment Start pulse and continues until the Depth counter's count-down has expired. The memory allotted to the acquisition is equal to the Segment Size and is arranged in a circular fashion. Notice that raw trigger events are ignored until the Trigger Holdoff time has elapsed.

Note also that, although pre-trigger data are acquired throughout the time between Segment Start and the Trigger Event, most of the acquired pre-trigger data have been overwritten in the diagram below and only a small fraction are available for download.
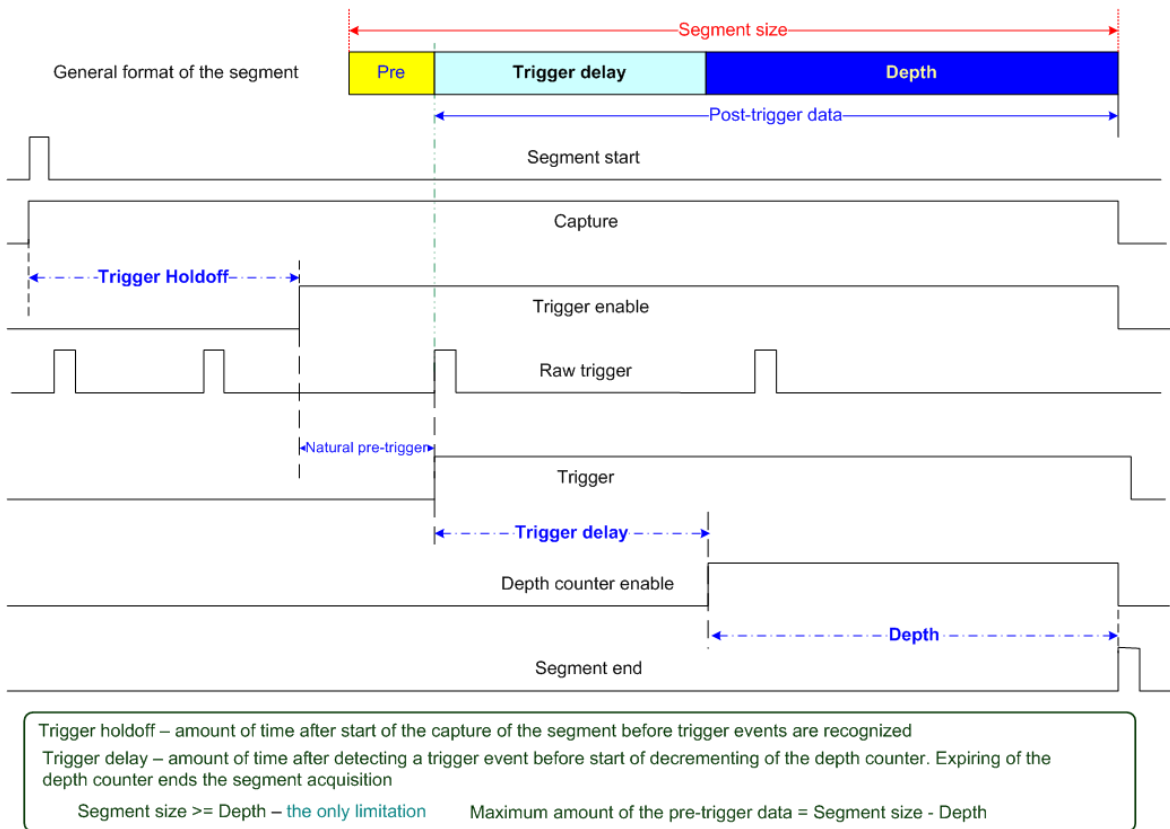
Representative Acquisition Sequences

The diagram below illustrates important representative acquisition sequences that are presented for key acquisition requirements. The first sequence shows a generalized acquisition with data collected during all acquisitions phases illustrated: overwritten pre-trigger data, available pre-trigger data, data acquired during the Trigger Delay and post-trigger data.

The second sequence shows an acquisition using a trigger delay where the Segment Size is made equal to the Depth. This acquisition sequence is useful for applications where the signal region of interest occurs long after the trigger event, as is often the case for Time Domain Reflectometry methods such as ultrasonics, radar and lidar.

The third sequence shows the acquisition in the case of a rapidly occurring trigger. In this case, the Segment size has been set bigger than the Depth. However, the trigger event occurred so rapidly after the start of the acquisitions that there was insufficient time to fill all of the memory allotted for pre-trigger data, which is (Segment Size – Depth) samples. Consequently, there is less preceding valid trigger data than the maximum possible amount of pre-trigger data. If transfer of invalid data were requested, the driver would return an actual start address that is equal to the address of the first valid pre-trigger point. More pre-trigger data could have been acquired using Trigger Holdoff at the expense ignoring (and so possibly missing) triggers.

The final sequence shows acquisition in the case of a trigger that occurs long after Segment Start so that all memory allotted for pre-trigger data has been filled up. In fact, the sequence shows that still more pre-trigger data were acquired but were overwritten by post-trigger data.

General format of the segment

Usually if trigger delay is not zero
Segment size = Depth

Amount of pre-trigger data due to hold off and natural pre trigger is
less then Segment size - Depth

Amount of the pre-trigger data is equal or bigger
then Segment Size - Depth

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:

> 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 (i.e. CD-ROM, joystick, network card, etc.)

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

GaGe Products

For ordering information, see Gage's product catalog, or visit our web site at
http://www.gage-applied.com

CompactPCI Bus Products	CompuScope 82GC	8 bit, 2 GS/s Analog Input Card
	CompuScope 14100C	14 bit, 100 MS/s Analog Input Card
	CompuScope 1610C	16 bit, 10 MS/s Analog Input Card
	CompuScope 3200C	32 bit, 100 MHz Digital Input for CompactPCI Bus
PCI Bus Products	CompuScope 1610	16 bit, 10 MS/s Analog Input Card
	CompuScope 1602	16 bit, 2.5 MS/s Analog Input Card
	CompuScope 14200	14 bit, 200 MS/s Analog Input Card
	CompuScope 14105	14 bit, 105 MS/s Analog Input Card
	Octopus multi-channel digitizer family	Up to 8 channels on a single-slot PCI card, 12 or 14-bit resolution, 10 to 125 MS/s
	CompuScope 12400	12 bit, 400 MS/s Analog Input Card
	CompuScope 1220	12 bit, 20 MS/s Analog Input Card
	Cobra CompuScope family	8-bit, up to 2 GS/s Analog Input Card
	BASE-8 CompuScope	8-bit, up to 500 MS/s Analog Input Card
	CompuScope 3200	32 bit, 100 MHz Digital Input for PCI Bus
CompuGen PCI	CompuGen 4300/4302	12 bit, 4-channel, 300 MHz Analog Output Card
	CompuGen 8150/8152	12 bit, 8-channel, 150 MHz Analog Output Card
	CompuGen 11G/11G2	12 bit, 1 GHz Analog Output Card
CompuGen ISA	CompuGen 1100	12 bit, 80 MS/s Analog Output Card
	CompuGen 3250	32 bit, 50 MHz Digital Output Card
Application Software	GageScope	World's Most Powerful Oscilloscope Software
	GageBit	Digital Input/Output Software
	CompuGen for Windows	Arbitrary Waveform Generator Software for Windows
Software Development Kits	CompuScope SDK for C/C# for Windows	
	CompuScope LabVIEW SDK for Windows	
	CompuScope MATLAB SDK for Windows	
	CompuGen SDK for C/C++	
	CompuGen LabVIEW SDK	
	CompuGen MATLAB SDK	
Instrument Mainframes	LapScope-1	1-slot PCI Expansion Chassis for CompuScope and CompuGen cards
	LapScope-2	2-slot PCI Expansion Chassis for CompuScope and CompuGen cards
	Instrument Mainframe 7500	4-slot Portable Instrument Mainframe for CompuScope and CompuGen cards
	Instrument Mainframe 2020E	18 PCI slot and 1 ISA slot Instrument Mainframe for CompuScope and CompuGen cards
	Instrument Mainframe 4000	5 PCI-X and 1 PCI slot Instrument Mainframe for CompuScope and CompuGen cards
	Instrument Mainframe 8000C	7 cPCI/PXI slot Instrument Mainframe for CompuScope CompactPCI/PXI cards