

# glscopeclient Operator Manual

Andrew Zonenberg  
azonenberg@drawersteak.com

June 22, 2020

## **Abstract**

This document is the user manual for glscopeclient, a user interface and signal analysis tool for oscilloscopes and logic analyzers. As of this writing, glscopeclient is under active development but has not had a formal v0.1 release and should be considered alpha quality.

This is free software: you are free to change and redistribute it. There is NO WARRANTY, to the extent permitted by law.

# Contents

<b>1</b>	<b>Revision History</b>	<b>6</b>
<b>2</b>	<b>Legal Notices</b>	<b>7</b>
2.1	Introduction	7
2.2	License Agreement	7
2.3	Trademarks	7
2.4	Third Party Licenses	7
<b>3</b>	<b>Getting Started</b>	<b>8</b>
3.1	Documentation Conventions	8
3.2	Host System Requirements	8
3.3	Instrument Support	8
3.4	Compilation	8
3.4.1	Linux	8
3.4.2	Windows	9
3.5	Running glscopeclient	10
3.6	Design Philosophy	11
<b>4</b>	<b>Transports</b>	<b>12</b>
4.1	lan	12
4.2	lxi	12
4.3	vicp	12
4.4	usbtmc	12
<b>5</b>	<b>Oscilloscope Drivers</b>	<b>13</b>
5.1	Agilent	13
5.1.1	agilent	13
5.2	Antikernel Labs	13
5.2.1	akila	13
5.2.2	aklabs	13
5.3	Enjoy Digital	13
5.4	Hantek	13
5.5	Keysight	13
5.6	Pico Technologies	13
5.7	Rigol	14
5.7.1	rigol	14
5.8	Rohde & Schwarz	14
5.9	Saleae	14
5.10	Siglent	14
5.11	Teledyne LeCroy / LeCroy	14
5.11.1	lecroy	15
5.12	Tektronix	15
5.13	Xilinx	15
<b>6</b>	<b>Main Window</b>	<b>16</b>
6.1	Menu	16
6.1.1	File	16
6.1.2	Setup	16
6.1.3	View	16
6.1.4	Add	16
6.2	Toolbar	16

6.2.1	Capture buttons . . . . .	17
6.2.2	History button . . . . .	17
6.2.3	Opacity slider . . . . .	17
<b>7</b>	<b>Waveform Groups</b>	<b>19</b>
7.1	Managing Groups . . . . .	19
<b>8</b>	<b>Timeline</b>	<b>21</b>
<b>9</b>	<b>Waveform Views</b>	<b>22</b>
9.1	Plot Area . . . . .	22
9.2	Y Axis Scale . . . . .	22
9.3	Channel Information Box . . . . .	22
9.4	Overlays . . . . .	23
9.5	Statistics . . . . .	23
<b>10</b>	<b>History View</b>	<b>24</b>
10.1	Estimating Waveform Memory Usage . . . . .	24
<b>11</b>	<b>Protocol Analyzer View</b>	<b>25</b>
<b>12</b>	<b>Filters</b>	<b>26</b>
12.1	Introduction . . . . .	26
12.1.1	Key Concepts . . . . .	26
12.1.2	Conventions . . . . .	26
12.2	8B/10B (IBM) . . . . .	27
12.2.1	Parameters . . . . .	27
12.2.2	Output Signal . . . . .	27
12.3	8B/10B (TMD5) . . . . .	28
12.3.1	Inputs . . . . .	28
12.3.2	Parameters . . . . .	28
12.3.3	Output Signal . . . . .	28
12.4	AC Couple . . . . .	29
12.4.1	Inputs . . . . .	29
12.4.2	Parameters . . . . .	29
12.4.3	Output Signal . . . . .	29
12.5	ADL5205 . . . . .	30
12.5.1	Inputs . . . . .	30
12.5.2	Parameters . . . . .	30
12.5.3	Output Signal . . . . .	30
12.6	Base . . . . .	31
12.6.1	Inputs . . . . .	31
12.6.2	Parameters . . . . .	31
12.6.3	Output Signal . . . . .	31
12.7	CAN . . . . .	32
12.8	Clock Jitter (TIE) . . . . .	33
12.9	Clock Recovery (PLL) . . . . .	34
12.10	Clock Recovery (UART) . . . . .	35
12.11	Current Shunt . . . . .	36
12.12	DC Offset . . . . .	37
12.12.1	Inputs . . . . .	37
12.12.2	Parameters . . . . .	37
12.12.3	Output Signal . . . . .	37

12.13 DDR3 Command Bus . . . . .	38
12.14 Deskew . . . . .	39
12.14.1 Inputs . . . . .	39
12.14.2 Parameters . . . . .	39
12.14.3 Output Signal . . . . .	39
12.15 DRAM Trcd . . . . .	40
12.16 DRAM Trfc . . . . .	41
12.17 DVI . . . . .	42
12.18 Ethernet - 10baseT . . . . .	43
12.19 Ethernet - 100baseTX . . . . .	44
12.20 Ethernet - GMII . . . . .	45
12.21 Ethernet - RGMII . . . . .	46
12.22 Ethernet Autonegotiation . . . . .	47
12.23 Eye Bit Rate . . . . .	48
12.24 Eye Height . . . . .	49
12.25 Eye P-P Jitter . . . . .	50
12.26 Eye Pattern . . . . .	51
12.27 Eye Period . . . . .	52
12.28 Eye Width . . . . .	53
12.29 Fall . . . . .	54
12.30 FFT . . . . .	55
12.31 Frequency . . . . .	56
12.32 Horizontal Bathtub . . . . .	57
12.33 HDMI . . . . .	58
12.34 $I^2C$ . . . . .	59
12.35 JTAG . . . . .	60
12.36 MDIO . . . . .	61
12.37 Moving Average . . . . .	62
12.38 Multiply . . . . .	63
12.39 Overshoot . . . . .	64
12.40 Parallel Bus . . . . .	65
12.41 Peak-to-Peak . . . . .	66
12.42 Period . . . . .	67
12.43 Rise . . . . .	68
12.44 SPI . . . . .	69
12.45 Subtract . . . . .	70
12.45.1 Inputs . . . . .	70
12.45.2 Parameters . . . . .	70
12.45.3 Output Signal . . . . .	70
12.46 Threshold . . . . .	71
12.46.1 Inputs . . . . .	71
12.46.2 Parameters . . . . .	71
12.46.3 Output Signal . . . . .	71
12.47 Top . . . . .	72
12.47.1 Inputs . . . . .	72
12.47.2 Parameters . . . . .	72
12.47.3 Output Signal . . . . .	72
12.48 UART . . . . .	73
12.49 USB 1.0 / 2.x Activity . . . . .	74
12.50 USB 1.0 / 2.x Packet . . . . .	75
12.51 USB 1.0 / 2.x PCS . . . . .	76
12.52 USB 1.0 / 2.x PMA . . . . .	77

12.53 Undershoot . . . . . 78

12.54 Upsample . . . . . 79

12.55 Waterfall . . . . . 80

## **1 Revision History**

- June 22, 2020: [in progress] Initial draft

## 2 Legal Notices

### 2.1 Introduction

glscopeclient, libscopehal, and the remainder of the project are all released under the 3-clause BSD license (reproduced below). This is a permissive license, explicitly chosen to encourage integration with third-party open source and commercial projects.

### 2.2 License Agreement

Copyright (c) 2012-2020 Andrew D. Zonenberg. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions, and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions, and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of the author nor the names of any contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE AUTHORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHORS BE HELD LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

### 2.3 Trademarks

This document frequently mentions the names of various test equipment vendors and products in order to discuss glscopeclient's compatibility with said products. The reader should assume that these are all trademarks of their respective owners.

### 2.4 Third Party Licenses

TODO:

- yaml-cpp (shared, MIT license)
- gtkmm (shared, LGPL)
- FFTS (shared, BSD-3)
- liblxi (shared, BSD-3/EPICS)

## 3 Getting Started

### 3.1 Documentation Conventions

Items to be selected from a menu are displayed in monospace font.

Multilevel menu paths are separated by a / character. For example, Attenuation / 1x means to open the Attenuation submenu and select the 1x item.

If there are multiple options for a menu or configuration option, they are displayed in square brackets and separated by a | character. For example, Move waveform to / Waveform Group [1|2] means to select either Waveform Group 1 or Waveform Group 2 from the Move waveform to menu.

This project is under active development and is not anywhere near feature complete! As a result, this document is likely to refer to active bug or feature request tickets on the GitHub issue trackers. Issues are referenced as repository:ticket, for example scopehal-apps:3.

### 3.2 Host System Requirements

All current development is performed on Linux operating systems (primarily Debian and Arch), although experimental (and incomplete) Windows support is provided. glscopeclient uses gtkmm as the UI toolkit. Current development mostly uses 3.24 but any recent 3.x version should work.

Any 64-bit ARM or Intel processor should be able to run glscopeclient. TODO: suggested minimum performance depending on waveform depth etc?

A mouse with scroll wheel, or touchpad with scroll gesture support, is mandatory to enable full use of the UI. We may explore alternative input methods for some UI elements in the future.

Waveform rendering is performed in compute shaders, so OpenGL 4.3 or newer is required. The corresponding minimum hardware requirement is an AMD Radeon HD 5000, NVIDIA GeForce 400 series discrete GPU, or Intel Haswell or newer integrated GPU plus suitably up-to-date drivers. TODO: what AMD integrated/ARM GPUs started supporting GL 4.3?

The minimum RAM requirement to actually launch glscopeclient is relatively small (TODO: do some testing) however history mode and deep captures can easily consume many GB of RAM. We suggest 8GB as a reasonable minimum, with 32 or more encouraged for deep history.

### 3.3 Instrument Support

glscopeclient uses the libscopehal library to communicate with oscilloscopes, so any libscopehal-compatible hardware should work with glscopeclient. See the [Oscilloscope Drivers](#) section for more details on which hardware is supported and how to configure specific drivers.

### 3.4 Compilation

glscopeclient can be compiled on both Linux and Windows, but the specific steps that have to be taken differ quite a lot between these the two.

#### 3.4.1 Linux

1. Install dependencies. On Debian/Ubuntu:

```
1 sudo apt install build-essential cmake pkg-config libglm-dev \  
2     libgtkmm-3.0-dev libsigc++-2.0-dev libyaml-cpp-dev \  
3     liblxi-dev texlive texlive-fonts-extra libglew-dev
```

2. Install FFTS library

```
1 git clone https://github.com/anthonix/ffts.git  
2 cd ffts  
3 mkdir build  
4 cd build
```



```
5 cmake ../
6 make -j
7 sudo make install
```

### 3. Build scopehal and scopehal-apps

```
1 git clone https://github.com/azonenberg/scopehal-cmake.git --recurse-
  submodules
2 cd scopehal-cmake
3 mkdir build
4 cd build
5 cmake ../
6 make -j
```

4. Install scopehal and scopehal-apps: right now, you don't. As of now, glscopeclient is intended to be run from the glscopeclient source directory (src/glscopeclient). Anybody want to contribute and set up a proper install process?

### 3.4.2 Windows

On Windows, we make use of the MSYS2 development environment, which gives us access to the MingGW-w64 toolchain. Since this toolchain allows glscopeclient to be compiled as a native Windows application, MSYS2 is only needed to actually compile the project, not to run it.

1. Download and install MSYS2. You can download it here: <https://www.msys2.org/>  
It is of utmost importance that **all** steps outlined on the website are followed precisely, even if they might seem unnecessary. This will avoid lots of hard to diagnose problems later on in the build.

All following steps are to be done in a MinGW64 shell (**not** in a MSYS shell, which also gets installed by the MSYS2 installer).

#### 2. Install build tools

```
1 pacman -S mingw-w64-x86_64-gcc mingw-w64-x86_64-cmake \
2   mingw-w64-x86_64-make
```

#### 3. Install dependencies

```
1 pacman -S mingw-w64-x86_64-glm mingw-w64-x86_64-libsigc++ \
2   mingw-w64-x86_64-gtkmm3 mingw-w64-x86_64-yaml-cpp \
3   mingw-w64-x86_64-glew
```

#### 4. Build FFTS library

```
1 git clone https://github.com/anthonix/ffts.git
2 cd ffts
3 mkdir build
4 cd build
5 cmake -G"MinGW Makefiles" ../
6 mingw32-make -j
7 cp ./libffts_static.a /mingw64/lib/
8 cp ../include/ffts.h /mingw64/include/
```

#### 5. Build scopehal and scopehal-apps

```
1 git clone https://github.com/azonenberg/scopehal-cmake.git --recurse-
  submodules
2 cd scopehal-cmake
3 mkdir build
```

```

4  cd build
5  cmake -G"MinGW Makefiles" -DLIBFFTS_INCLUDE_DIR=/mingw64/include/ \
6      -DLIBFFTS_LIBRARIES=/mingw64/lib/libffts_static.a ../
7  mingw32-make -j

```

## 6. Copying binaries and running glscopeclient

Since glscopeclient is built using the MinGW toolchain, it depends on a rather large number of dynamic libraries. Since the application cannot find these libraries when being run from the build directory, there are two ways to make glscopeclient runnable: “packaging” the application by collecting all the dependencies, and installing it into the MinGW64 virtual file system.

### (a) Installing glscopeclient into the virtual filesystem

```

1  cp ./src/glscopeclient/glscopeclient.exe /mingw64/bin/
2  cp -r ./src/glscopeclient/shaders /mingw64/bin/
3  cp -r ./src/glscopeclient/styles /mingw64/bin/
4  cp -r ./src/glscopeclient/gradients /mingw64/bin/
5  cp ./lib/graphwidget/libgraphwidget.dll /mingw64/bin/
6  cp ./lib/log/liblog.dll /mingw64/bin/
7  cp ./lib/scopehal/libscopehal.dll /mingw64/bin/
8  cp ./lib/scopeprotocols/libscopeprotocols.dll /mingw64/bin/

```

The application can now be started from any MinGW64 terminal session:

```

1  glscopeclient.exe

```

### (b) Packaging glscopeclient and its dependencies This approach is not officially supported yet.

## 3.5 Running glscopeclient

When running glscopeclient with no arguments, a dialog box (Fig. 1) is displayed allowing you to connect to an instrument. As of this writing, there is no support for connecting to multiple instruments via the dialog.

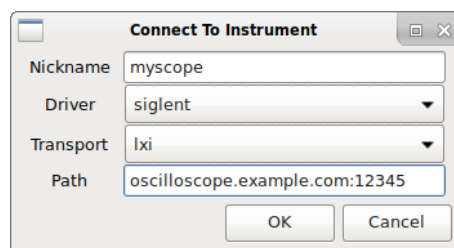


Figure 1: Connection dialog

It is also possible to connect to arbitrarily many instruments by passing the “connection string” for each instrument as a command-line argument.

```

1  ./glscopeclient --debug \
2      mylecroy:lecroy:vicp:myscope.example.com:1234 \
3      myrigol:rigol:lan:rigol.example.com

```

The `-debug` argument may be omitted or replaced with any other liblogtools argument for controlling console debug verbosity (`-quiet`, `-verbose`, `-debug`, `-trace`, etc). If you’re using glscopeclient at its current level of maturity you’re probably a developer, so we suggest `-debug` by default.

Each instrument is described by a “connection string” containing four colon-separated fields.

- Nickname. This can be any text string not containing spaces or colons. If you have only one instrument it's largely ignored, but when multiple instruments are present channel names in the UI are prefixed with the nickname to avoid ambiguity.
- Driver name. This is a string identifying the command protocol the scope uses. Note that not all scopes from the same vendor will use the same command set or driver!
- Transport. This is a string describing how the driver connects to the scope (e.g. RS232 or Ethernet)
- Arguments for the driver identifying the device to connect to, separated by colons. This varies by driver but is typically a hostname:port combination, TTY device path, or similar.

### **3.6 Design Philosophy**

glscopeclient's UI is heavily mouse driven and context based. Space used by always-visible buttons, sliders, etc is kept to a minimum in order to keep as much screen real estate as possible usable for waveform display. Additional controls are displayed in menus or pop-up dialogs, then hidden as soon as they are not needed.

Most UI elements can be interacted with by left clicking (select), left dragging (move), using the scroll wheel (zoom), double clicking (open properties dialog), or right clicking (context menu).

## 4 Transports

### 4.1 lan

SCPI over TCP with no further encapsulation.

This transport takes two arguments: hostname/IP and port number.

If port number is not specified, uses TCP port 5025 (IANA assigned) by default. Note that Rigol oscilloscopes use the non-standard port 5555, not 5025, so the port number must always be specified when using a Rigol instrument.

### 4.2 lxi

SCPI over LXI VXI-11.

Note that due to the remote procedure call paradigm used by LXI, it is not possible to batch multiple outstanding requests to an instrument when using this transport. Some instruments may experience reduced performance when using LXI as the transport.

TODO: document this more completely.

### 4.3 vicp

SCPI over Teledyne LeCroy Virtual Instrument Control Protocol.

This transport takes two arguments: hostname/IP and port number.

If port number is not specified, uses TCP port 1861 (IANA assigned) by default.

### 4.4 usbtmc

SCPI over USB Test & Measurement Class protocol.

This transport takes one argument: path to the usbtmc kernel driver

Example:

```
1 ./glscopeclient myscope:siglent:usbtmc:/dev/usbtmc0
```

## 5 Oscilloscope Drivers

### 5.1 Agilent

Agilent devices support a similar similar SCPI command set across most device families.

Please see the table below for details of current hardware support:

Device Family	Driver	Notes
DSO5000 series	agilent	Not recently tested, but should work.
DSO6000 & MSO6000 series	agilent	Working. No support for digital channels yet.
DSO7000 & MSO7000 series	agilent	Untested, but should work. No support for digital channels yet.

#### 5.1.1 agilent

Example:

```
1 ./glscopeclient --debug myscope:agilent:lan:192.168.1.1:5025
```

This driver has been tested on an MSO6034A.

### 5.2 Antikernel Labs

Device Family	Driver	Notes
Internal Logic Analyzer IP	akila	
BLONDEL Oscilloscope Prototype	aklabs	

#### 5.2.1 akila

This driver uses a raw binary protocol, not SCPI.

Under-development internal logic analyzer analyzer core for FPGA design debug. The ILA uses a UART interface to a host system. Since there's no UART support in scopehal yet, socat must be used to bridge the UART to a TCP socket using the "lan" transport.

#### 5.2.2 aklabs

This driver uses two TCP sockets. Port 5025 is used for SCPI control plane traffic, and port 50101 is used for waveform data using a raw binary protocol.

### 5.3 Enjoy Digital

TODO (scopehal:79)

### 5.4 Hantek

TODO (scopehal:26)

### 5.5 Keysight

TODO

### 5.6 Pico Technologies

TODO (scopehal:15)

## 5.7 Rigol

Rigol oscilloscopes have subtle differences in SCPI command set, but this is implemented with quirks handling in the driver rather than needing different drivers for each scope family.

Device Family	Driver	Notes
DS1000Z	rigol	
MSO5000	rigol	

### 5.7.1 rigol

This driver has been primarily tested on a MSO5000 series scope.

## 5.8 Rohde & Schwarz

TODO (scopehal:59)

## 5.9 Saleae

TODO (scopehal:16)

### 5.10 Siglent

Many recent Siglent oscilloscopes are developed in partnership with Teledyne LeCroy (Siglent-designed hardware running Teledyne LeCroy firmware) and are sold under both brands. As a result, there is some crossover in driver support.

Device Family	Driver	Transport	Notes
SDS5000X series	siglent	lxi, lan, usbtmc	Untested
SDS2000X Plus series	siglent	lxi, lan, usbtmc	Untested
SDS2000X-E series	siglent	lxi, lan, usbtmc	Untested
SDS2000X series	siglent	lxi, usbtmc	Base functionality present, lxi and usbtmc tested on SDS2304X.
SDS1000X series	siglent	lxi, usbtmc	Untested
SDS1000X+ series	siglent	lxi, usbtmc	Untested
SDS1000X-E series	siglent	lxi, lan, usbtmc	Base functionality present, lan transport tested on SDS1204X-E.
SDS1000CFL series	siglent	usbtmc, ?	Untested. Documentation contradiction about support over Ethernet.
SDS1000DL+ series	siglent	lxi, usbtmc	Untested
SDS1000CML+ series	siglent	lxi, usbtmc	Untested
SDS3000X series	lecroy	vicp	Should be same as WaveSurfer 3000

Unlike TCP/IP sockets ("lan"), VXI-11 ("lxi") is a synchronous protocol that does not support queueing multiple transactions. When an oscilloscope supports both the lan and the lxi transport, chances are that the lan transport will have a better performance than the lxi transport.

### 5.11 Teledyne LeCroy / LeCroy

Teledyne LeCroy (and older LeCroy) devices use the same driver, but two different transports for LAN connections.

While all Teledyne LeCroy / LeCroy devices use almost identical SCPI command sets, Windows based devices running XStream or MAUI use a custom framing protocol ("vicp") around the SCPI data while the lower end RTOS based devices use raw SCPI over TCP ("lan"). Some of these devices also require use of

the Siglent driver as they are Siglent OEM designs rebranded by Teledyne LeCroy and have some quirks in the firmware which require workarounds.

Please see the table below for details on which configuration to use with your hardware.

Device Family	Driver	Transport	Notes
DDA	lecroy	vicp	
HDO	lecroy	vicp	
LabMaster	lecroy	vicp	Untested, but should work
MDA	lecroy	vicp	Untested, but should work
SDA	lecroy	vicp	Untested, but should work
T3DSO	siglent	lan	Untested, but should work
WaveAce	siglent	lan	Untested, but should work
WaveJet	siglent	lan	Untested, but should work
WaveMaster	lecroy	vicp	Untested, but should work
WaveRunner	lecroy	vicp	
WaveSurfer	lecroy	vicp	

### 5.11.1 lecroy

This is the primary driver for MAUI based Teledyne LeCroy / LeCroy devices.

Example:

```
1 ./glscopeclient --debug myscope:lecroy:vicp:192.168.1.1:1861
```

This driver has been tested on a wide range of Teledyne LeCroy / LeCroy hardware including DDA 5005, DDA 5005A, WaveSurfer 3034, WaveRunner 8104, and HDO9204. It should be compatible with any Teledyne LeCroy or LeCroy oscilloscope running Windows XP or newer and the MAUI or XStream software.

### 5.12 Tektronix

TODO (scopehal:73, scopehal:13)

### 5.13 Xilinx

TODO (scopehal:40)

## 6 Main Window

The main window of glscopeclient consists of the menu bar and tool bar at top and a status bar at the bottom. All remaining space is occupied by one or more waveform groups.

### 6.1 Menu

#### 6.1.1 File

This menu contains commands for manipulating glscopeclient session files.

A session consists of a YAML file called filename.scopesession containing instrument and UI configuration, as well as a directory called filename\_data which contains waveform metadata and sample values for all enabled instrument channels.

**Save:** The UI layout may be saved to a session file so that a common instrument setup can be returned to in the future. Waveform data may optionally be saved in the session as well.

**Open:** Loads a .scopesession file. Checkboxes at the bottom of the file browser dialog allow the UI configuration, instrument settings, and waveform data to be individually loaded or ignored.







**Quit:** Exits the application

#### 6.1.2 Setup

This menu allows instrument settings to be configured. Currently there's not much here.

#### 6.1.3 View

This menu allows display settings to be configured. As of now, the only option is selection of the color palette for eye patterns.

Name	Colors	Notes
CRT		Similar color scheme to a major scope vendor.
Grayscale		Common monochrome palette.
Ironbow		Common "hot metal" palette.
KRain		Similar color scheme to a major scope vendor.
Rainbow		Common HSV rainbow palette.
Viridis		Perceptually uniform palette from matplotlib.

#### 6.1.4 Add

This menu allows a new waveform view to be created for any channel on a connected instrument.

### 6.2 Toolbar

The toolbar contains buttons and controls for the most frequently used actions.

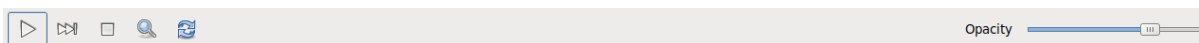


Figure 2: glscopeclient toolbar



### 6.2.1 Capture buttons

The capture button group (Fig. 3) contains three buttons. From left to right these are “arm normal trigger”, “arm one-shot trigger” and “stop trigger”.

Note that the “normal” trigger mode still uses one-shot capture internally so that all waveform data can be downloaded before the next trigger event.



Figure 3: Capture control buttons

### 6.2.2 History button

The history button (Fig. 4) toggles display of the [waveform history view](#).



Figure 4: History button

### 6.2.3 Opacity slider

The opacity slider (Fig. 5) controls the alpha/opacity used to display intensity-graded waveforms. Higher opacity values lead to better display of sparse waveforms (compare the crisp lines of Fig. 6 to the barely visible trace in Fig. 7) but can lead to a washed-out appearance if too many sample points are shoved into a small area.

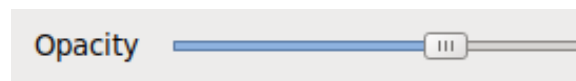


Figure 5: Trace opacity slider

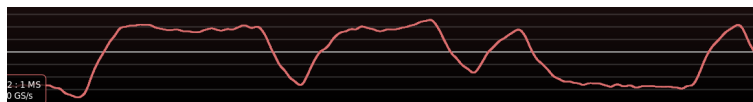


Figure 6: Sparse waveform at a high zoom level

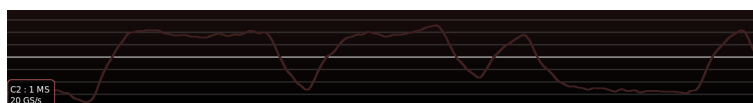


Figure 7: Dim waveform showing difficulty of seeing waveform at low opacity

For example, the DVI waveform in Fig. 8 looks like a solid white blob with a vaguely visible outline. No fine detail can be observed other than the increased over/undershoot and random-looking edges on the scanlines, compared to the flat appearance of the blanking period between scanlines and at the end of the frame.

When the opacity is reduced in this example, many more nuances of the signal become apparent. The high/low voltage levels of the signal compared to the transitions between them are obvious, and the H/V sync pulses within the blanking period show up as a slightly darker region.

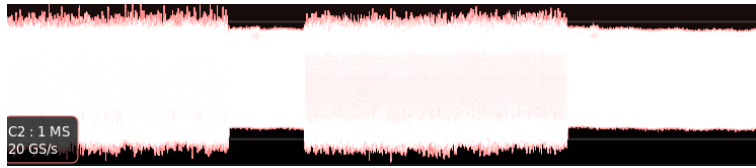


Figure 8: Intensity-graded waveform showing washed-out appearance at high opacity

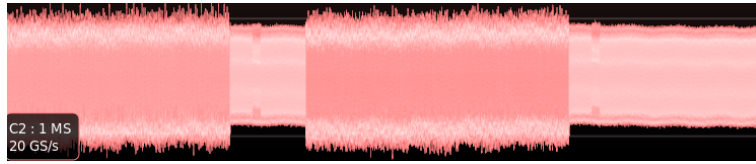


Figure 9: Intensity-graded waveform at lower opacity level

As of this writing, the opacity setting is global for the entire application. Should this be changed to per waveform group? If so, how should the group be selected and should there still be an option to make changes globally?

## 7 Waveform Groups

A waveform group is a collection of one or more waveforms stacked vertically under a common timeline. All waveforms within a group share the same timeline and vertical cursor(s).

When glscopeclient starts up, by default all channels on the attached instruments are displayed in a single waveform group (Figure 10).

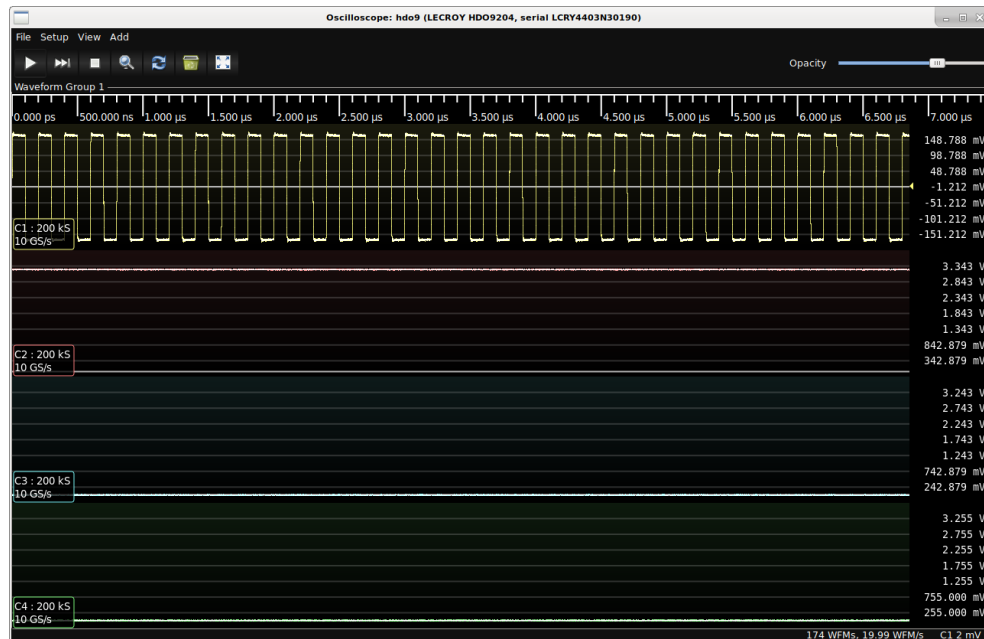


Figure 10: Top level glscopeclient window with a single waveform group

As you add protocol decodes or look at different parts of a waveform, it may be necessary to create additional waveform groups. Typical reasons for creating additional groups include:

- Zooming into one set of signals to see detail on short time scales while maintaining a high level overview of others
- Viewing signals with incompatible horizontal units. For example, a FFT has horizontal units of frequency while an analog waveform has horizontal units of time. Eye patterns also have horizontal units of time, but are always displayed as two UIs wide and cannot be zoomed. <sup>1</sup>

### 7.1 Managing Groups

Additional groups may be created by right clicking a waveform and selecting [Move|Copy] waveform to / Insert new group at [right|bottom] from the context menu. This will split the current group's area in half horizontally or vertically, with the selected waveform moved or copied to the newly added group and all other waveforms in the original group.

Waveforms may be also be moved within, or between, groups by clicking the channel information box and dragging it. A yellow insertion bar will appear when dragging, showing the location the waveform will be inserted in.

If a waveform is dragged to the very bottom or right side of a waveform group, the destination group will be split vertically or horizontally and the new waveform will be inserted below or to the right of the destination group. The insertion bar turns orange when dragging near the edge of a group, to indicate that a split will take place.

<sup>1</sup>It is currently possible to place signals with incompatible horizontal units in the same group. This may lead to confusion; a future software release will likely force creation of a new group if a protocol decode is incompatible with the parent trace's time scale.

Dividers between waveform groups may be dragged with the left mouse button. Any group may be subdivided again, to create arbitrarily complex tiles of waveforms. Figure 11 shows a two-level hierarchy created by moving channel 2 to a new group at right, then moving channel 4 to a new group below that one.

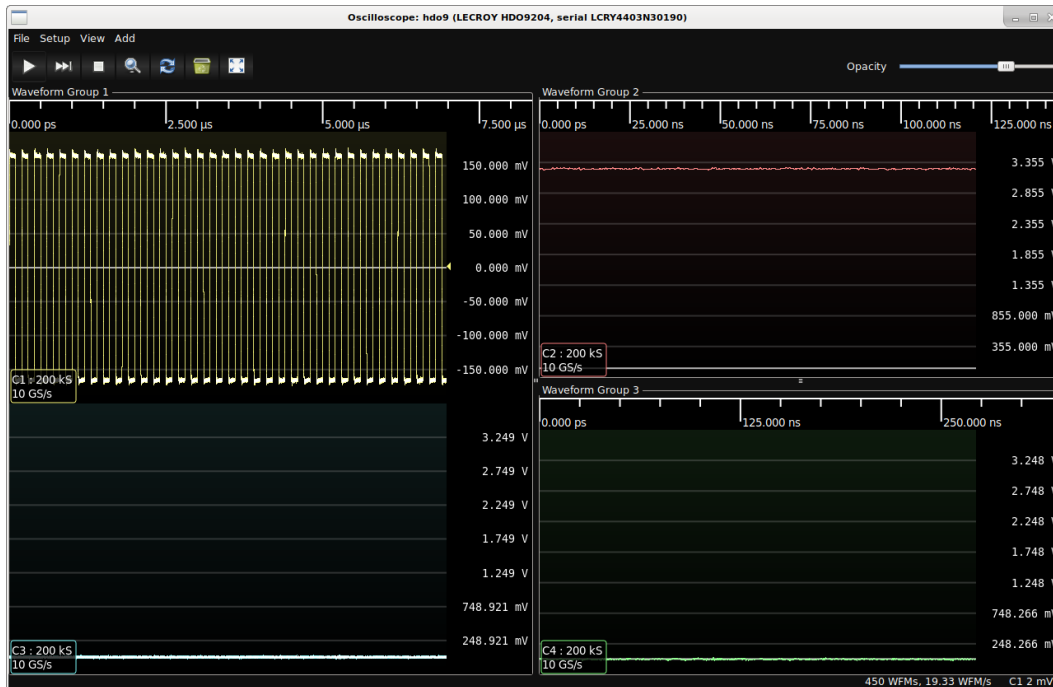


Figure 11: Top level glscopeclient window with several waveform groups separated by splitters

Protocol decode overlays may be reordered by dragging the channel information box with the mouse, however they cannot currently be moved to another waveform or group.

New waveform groups are given an automatically generated name when created, for example "Waveform Group 2". This name will be editable in a future software release (scopehal-apps:53).

## 8 Timeline

The timeline is displayed at the top of each waveform group and shows the X axis scale for the group. The timeline (and all accompanying waveform views in the group) may be zoomed by scrolling with the mouse wheel, or panned by dragging with the left mouse button.

Unlike classical oscilloscope user interfaces, there is *no relationship* between the timeline scale/position and the duration of the acquisition. It is possible to zoom or scroll beyond the end of the acquisition (displaying empty background with no signal) or have a deep capture in which nearly all acquired data is offscreen.

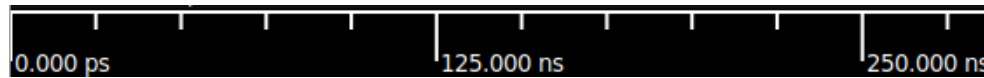


Figure 12: The timeline

TODO: talk about how to set trigger offset within acquisition, once implemented (scopehal:21, scopehal-apps:12)

Double-clicking on the timeline brings up the timebase properties dialog (Fig. 13), which allows the sample rate and memory depth to be configured. If multiple instruments are connected, a separate tab appears in the dialog for each instrument.

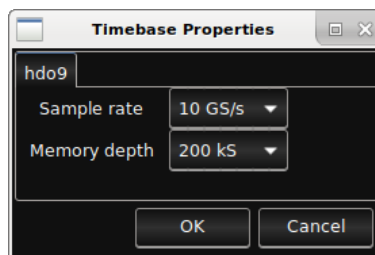


Figure 13: Timebase properties dialog

Note that the timeline may occasionally show units other than time. For example, an “eye width” measurement has X axis units of voltage and Y axis units of time.

## 9 Waveform Views

A waveform view is a 2D graph of a signal or protocol decode within a waveform group.

### 9.1 Plot Area

The plot area shows the waveform being displayed. The background has a subtle gradient from light at top to dark at bottom, in order to visually separate adjacent waveform view within the same group.

The horizontal grid lines line up with the voltage scale markings on the Y axis. If the plot area includes  $Y=0$ , the grid line for zero is slightly brighter.

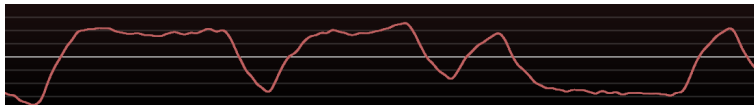


Figure 14: Waveform plot area

The waveform is drawn as a semi-transparent line so that when zoomed out, the density of voltage at various points in the graph may be seen as lighter or darker areas. This is referred to as “intensity grading”.

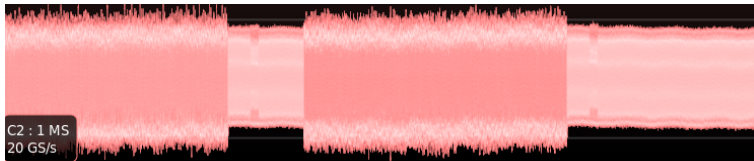


Figure 15: Intensity-graded waveform

### 9.2 Y Axis Scale

Each waveform view has its own Y axis scale, which is locked to the ADC range of the instrument.

If the waveform view is connected to a physical channel of an instrument, the gain may be configured by scrolling with the mouse wheel, and the offset may be adjusted by dragging with the left mouse button. If the view is displaying the output of a filter block, gain and offset are set by the filter and not adjustable.

If a left-pointing arrow (as seen in Fig. 16) is visible, the current channel is selected as a trigger source. Click on the arrow and drag up or down to select the trigger level.

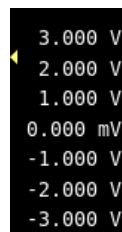


Figure 16: Y axis of a waveform view showing trigger arrow

### 9.3 Channel Information Box

The channel information box is displayed in the lower left corner of each waveform view. It contains summary information about the channel. Currently this is the display name of the channel, the sample rate, and the record length of the acquisition. Other information, such as probe coupling, may be displayed there in the future.

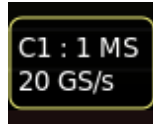


Figure 17: Channel information box

The information box may be dragged with the left mouse button to move the entire waveform view to a new location.

Double-clicking the information box opens the channel properties dialog (Fig. 18). This dialog allows changing of the channel's nickname or color. The "hardware name" of the channel is also displayed, so that a renamed channel can be easily traced back to a physical instrument input.

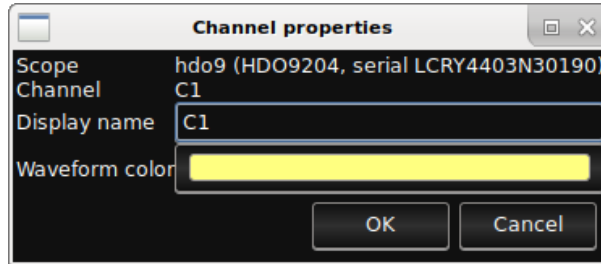


Figure 18: Channel properties dialog

## 9.4 Overlays

Waveforms may have additional information overlaid on top of them, such as protocol decodes. Each overlay has its own information box, which may be double-clicked to open the properties dialog and configure it just like any other channel.

Fig. 19 shows an example of an analog waveform with three overlays: thresholding it to NRZ digital, recovering a sampling clock with a CDR PLL, and finally decoding the serial NRZ data stream to TMDs protocol data and control events.

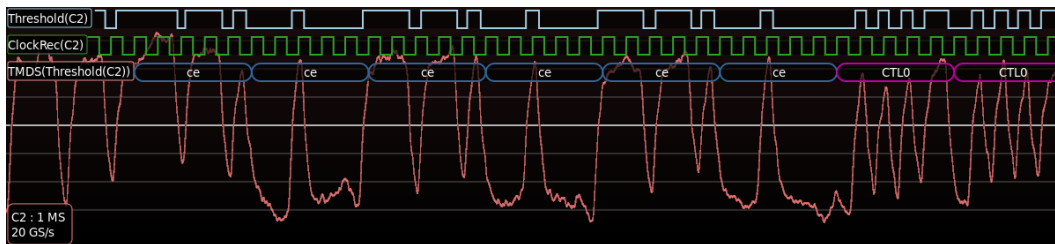


Figure 19: Waveform showing two digital overlays and a data decode overlay

Overlays can be deleted by means of the right-click context menu. Dragging the information box with the left mouse button allows overlays to be reordered, however they cannot currently be moved to another waveform view.

## 9.5 Statistics

Statistics may be shown for any waveform by checking the "statistics" box in the context menu. The default statistics are minimum, average, and maximum although more may be added in the future.

## 10 History View

glscopeclient has the ability to save every waveform during a session in memory, allowing you to go back in time and see previous state of the system being debugged. Clicking on a timestamp in the history view pauses acquisition and loads the historical waveform data for analysis.

By default, the history view (Fig. 20) is not displayed and no history is captured. If the history view is closed, history continues to be captured up to the configured maximum history depth.

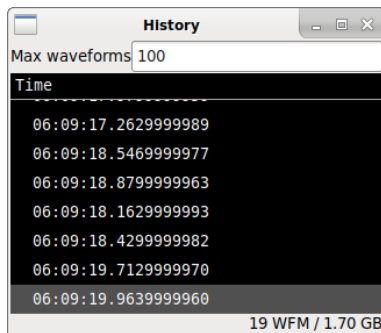


Figure 20: Waveform history view

The “max waveforms” box allows the depth of the history to be configured. It defaults to 10 but can be set to any positive integer value. Older waveforms beyond the history limit are deleted as new waveforms are acquired.

The status bar at the bottom of the history view displays the total number of waveforms in the history, as well as an estimate of the amount of RAM used by the history.

### 10.1 Estimating Waveform Memory Usage

When selecting a maximum depth for the history, it is important to pick a reasonable limit to avoid running out of RAM! glscopeclient will happily fill tens or hundreds of gigabytes of memory with deep waveforms if given a chance. Memory usage of waveform data can be roughly estimated as  $16 + \text{sizeof}(\text{sample type})$  bytes per point, since each sample contains a 64-bit timestamp and duration plus the sample data.

For example, an analog sample takes 20 bytes of RAM (16 of time plus a 32-bit floating point voltage measurement) per sample. Thus, a 1M point analog waveform takes approximately 20 MB of RAM per channel, or 80 MB per capture on a four-channel oscilloscope with all channels enabled.

On the larger side, a 10M point four channel capture would use 800 MB and a 64M point deep-memory capture would use 5 GB. A deep history setting, such as 100 waveforms, is thus wildly inappropriate for such deep captures! A future software release may support spilling waveform data to a temporary directory on disk, permitting effectively unlimited history depth given sufficient disk space.

Digital waveforms use one byte per sample for the actual measurement, so 17 MB per channel for a 1M point waveform.

Filter memory usage varies depending on the specific filter in question, however it is typically not a large contributor to the overall glscopeclient RAM footprint when using history mode because filters are evaluated dynamically each time a waveform is pulled from history rather than having output cached for every historical waveform. Thus, at most one copy of each filter’s output is present in memory regardless of history depth.

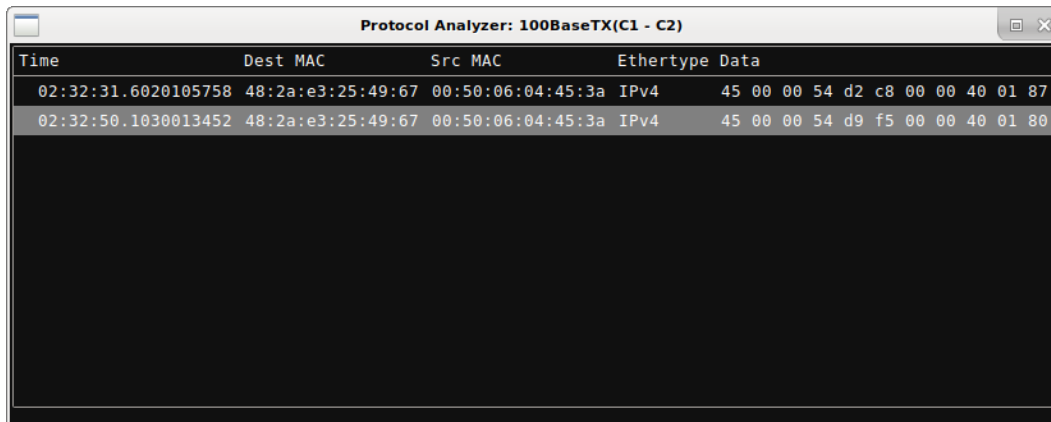


## 11 Protocol Analyzer View

Some filters for decoding packet-oriented data provide an alternate means of visualizing the decoded traffic.

The protocol analyzer view (Fig. 21) displays each packet in the history as a row in a list view. The first column is always the timestamp of the packet; remaining columns vary depending on the particular filter in question.

Clicking on a packet pauses acquisition, loads the relevant waveform from history if the packet is not in the current waveform, and scrolls the waveform view containing the protocol decode to the start of the packet. This allows packet-level data to be easily correlated to physical layer waveforms.



The screenshot shows a window titled "Protocol Analyzer: 100BaseTX(C1 - C2)". It contains a table with the following data:

Time	Dest MAC	Src MAC	Ethertype	Data
02:32:31.6020105758	48:2a:e3:25:49:67	00:50:06:04:45:3a	IPv4	45 00 00 54 d2 c8 00 00 40 01 87
02:32:50.1030013452	48:2a:e3:25:49:67	00:50:06:04:45:3a	IPv4	45 00 00 54 d9 f5 00 00 40 01 80

Figure 21: Protocol analyzer view

Once closed, the protocol analyzer view is gone for good, although it will eventually be possible to bring it back via some sort of menu (scopehal-apps:44).

## 12 Filters

### 12.1 Introduction

#### 12.1.1 Key Concepts

glscopeclient and libscopehal are based on a “filter graph” architecture internally. The filter graph is a directed acyclic graph with a set of source nodes (waveforms captured from hardware or loaded from a saved session) and sink nodes (waveform views, protocol analyzer views, and statistics) connected by edges representing data flow.

A filter is simply an intermediate node in the graph, which takes input from one or more waveform nodes and outputs a waveform which may be displayed, used as input to other filters, or both. A waveform is a series of data points which may represent voltages, digital samples, or arbitrarily complex protocol data structures.

As a result, there is no internal distinction between math functions, measurements, and protocol decodes, and it is possible to chain them arbitrarily. Consider the following example:

- Two analog waveforms representing serial data and clock are acquired
- Each analog waveform is thresholded, producing a digital waveform
- The two digital waveforms are decoded as  $I^2C$ , producing a series of packets
- The  $I^2C$  packets are decoded as writes to a serial DAC, producing an analog waveform
- A moving average filter is applied to the analog waveform
- A measurement filter finds the instantaneous frequency of each cycle of the DAC output

In this document we use the term “filter” consistently to avoid ambiguity.

#### 12.1.2 Conventions

Each filter takes one or more inputs (vector inputs), zero or more parameters (scalar inputs), and outputs a signal (vector output).

If the output signal is a complex-valued type (as opposed to a single scalar, e.g. voltage, at each sample) the “Output Signal” section will include a table describing how various types of output data are displayed. Printf-style format codes maybe used for clarity. For example, “%02x” means data is formatted as hexadecimal bytes with leading zeroes.

All filters with complex output use a standardized set of colors to display various types of data fields in a consistent manner. These colors are currently hard coded in a table but will be made editable in the future (scopehal-apps:43)

Suggestions on changes to the default colors, or new categories for color coding, are welcome

Color name	Use case	Default Color
Address	Memory addresses	#ffff00
Checksum Bad	Incorrect CRC/checksum	#ff0000
Checksum OK	Valid CRC/checksum	#00ff00
Control	Miscellaneous control data	#c000a0
Data	User data	#336699
Error	Malformed/unreadable data	#ff0000
Idle	Inter-frame gaps	#404040
Preamble	Preamble/sync words	#808080

## 12.2 8B/10B (IBM)

Decodes the standard 8b/10b line code used by SGMII, 1000base-X, DisplayPort, JESD204B, PCIe gen 1/2, SATA, USB 3.0, and many other common serial protocols.

Signal name	Type	Description
data	1-bit digital	Serial 8b/10b data line
clk	1-bit digital	DDR bit clock, typically generated by use of the <a href="#">Clock Recovery (PLL)</a> decode on the input data.

### 12.2.1 Parameters

This filter takes no parameters.

### 12.2.2 Output Signal

The 8B/10B filter outputs a time series of 8B/10B sample objects. These consist of a control/data flag and a byte of data.

Type	Description	Color	Format
Control	Control codes	Control	K%d.%d
Data	Pixel/island data	Data	D%d.%d
Error	Malformed data	Error	ERROR

### 12.3 8B/10B (TMDS)

Decodes the 8-to-10 Transition Minimized Differential Signalling line code used in DVI and HDMI.

#### 12.3.1 Inputs

Signal name	Type	Description
data	1-bit digital	Serial TMDS data line
clk	1-bit digital	DDR <i>bit</i> clock, typically generated by use of the <a href="#">Clock Recovery (PLL)</a> decode on the input data. Note that this is 5x the rate of the HDMI pixel clock signal.

#### 12.3.2 Parameters

This filter takes no parameters.

#### 12.3.3 Output Signal

The TMDS filter outputs a time series of TMDS sample objects. These consist of a type field and a byte of data.

The output of the TMDS decode is commonly fed to the [DVI](#) or [HDMI](#) protocol decoders.

Type	Description	Color	Format
Control	Control codes (H/V sync)	Control	CTL%d
Data	Pixel/island data	Data	%02x
Error	Malformed data	Error	ERROR
Guard band	HDMI data/video guard band	Preamble	GB

## 12.4 AC Couple

Automatically removes a DC offset from an analog waveform by subtracting the average of all samples from each sample. Dynamic range is lost compared to doing true AC coupling in the instrument front end, but there are certain situations where offset removal is necessary in postprocessing.

### 12.4.1 Inputs

Signal name	Type	Description
din	Analog	Input waveform

### 12.4.2 Parameters

This filter takes no parameters.

### 12.4.3 Output Signal

This filter outputs an analog waveform with identical sample rate to the input, vertically shifted to center the signal about zero volts.

## 12.5 ADL5205

Decodes SPI data traffic to one half of an ADL5205 variable gain amplifier.

### 12.5.1 Inputs

Signal name	Type	Description
spi	SPI bus	The SPI data bus

### 12.5.2 Parameters

This filter takes no parameters.

### 12.5.3 Output Signal

This filter outputs one ADL5205 sample object for each write transaction, formatted as “write: FA=2 dB, gain=8 dB”.

## 12.6 Base

Calculates the base (logical zero level) of each cycle in a digital waveform. It is most commonly used as an input to statistics, to view the average base of the entire waveform.

### 12.6.1 Inputs

Signal name	Type	Description
din	Analog	Input waveform

### 12.6.2 Parameters

This filter takes no parameters.

### 12.6.3 Output Signal

This filter outputs an analog waveform with one sample for each group of logical zeroes in the input signal, containing the average value of the zero level.

## 12.7 CAN



## 12.8 Clock Jitter (TIE)

## 12.9 Clock Recovery (PLL)

## **12.10 Clock Recovery (UART)**

### **12.11 Current Shunt**

Converts a voltage waveform acquired across a known resistance into a current waveform.

## 12.12 DC Offset

Adds a constant value to each sample in an analog waveform.

### 12.12.1 Inputs

Signal name	Type	Description
din	Analog	Input waveform

### 12.12.2 Parameters

Parameter name	Type	Description
Offset	Float	The offset to apply

### 12.12.3 Output Signal

This filter outputs an analog waveform with one sample for each sample in the input, shifted by the requested offset.

## 12.13 DDR3 Command Bus

## 12.14 Deskew

Moves an analog waveform earlier or later in time to compensate for trigger offsets, probe length mismatch, etc. It is generally preferable to deskew using the skew adjustment on the channel during acquisition; this filter is provided for correction in postprocessing.

### 12.14.1 Inputs

Signal name	Type	Description
din	Analog	Input waveform

### 12.14.2 Parameters

Parameter name	Type	Description
Skew	Float	Time offset to shift the waveform

### 12.14.3 Output Signal

This filter outputs an analog waveform with one sample for each sample in the input, phase shifted by the requested offset.

## 12.15 DRAM Trecd



## 12.16 DRAM Trfc

## 12.17 DVI

## **12.18 Ethernet - 10baseT**

## **12.19 Ethernet - 10obaseTX**

## **12.20 Ethernet - GMII**

## **12.21 Ethernet - RGMII**

## **12.22 Ethernet Autonegotiation**

## 12.23 Eye Bit Rate



## **12.24 Eye Height**

## 12.25 Eye P-P Jitter

## 12.26 Eye Pattern

## **12.27 Eye Period**

## 12.28 Eye Width

**12.29 Fall**

## 12.30 FFT

### 12.31 Frequency



## 12.32 Horizontal Bathtub

## 12.33 HDMI

**12.34**  $I^2C$

## 12.35 JTAG

## 12.36 MDIO

## 12.37 Moving Average

### **12.38 Multiply**

Multiplies one waveform by another. No resampling is performed; both inputs must have identical sample rates.

Unit conversions are performed, for example the product of a voltage and current waveform is a power waveform.

## 12.39 Overshoot



## 12.40 Parallel Bus

## 12.41 Peak-to-Peak

## 12.42 Period

## 12.43 Rise

## 12.44 SPI

## 12.45 Subtract

Subtracts one waveform from another. No resampling is performed; both inputs must have identical sample rates.

### 12.45.1 Inputs

Signal name	Type	Description
IN+	Analog	Positive input waveform
IN-	Analog	Negative input waveform

### 12.45.2 Parameters

This filter takes no parameters.

### 12.45.3 Output Signal

This filter outputs an analog waveform with one sample for each sample in the input, containing the difference of the two input waveforms.

## 12.46 Threshold

Converts an analog waveform to digital by thresholding at a constant level (no hysteresis).

### 12.46.1 Inputs

Signal name	Type	Description
din	Analog	Input waveform

### 12.46.2 Parameters

Parameter name	Type	Description
Threshold	Float	Decision threshold

### 12.46.3 Output Signal

This filter outputs an digital waveform with one sample for each sample in the input, which is true if the corresponding input sample is above the threshold and false if less than or equal.

## 12.47 Top

Calculates the top (logical one level) of each cycle in a digital waveform. It is most commonly used as an input to statistics, to view the average top of the entire waveform.

### 12.47.1 Inputs

Signal name	Type	Description
din	Analog	Input waveform

### 12.47.2 Parameters

This filter takes no parameters.

### 12.47.3 Output Signal

This filter outputs an analog waveform with one sample for each group of logical ones in the input signal, containing the average value of the one level.



## 12.48 UART

## **12.49 USB 1.0 / 2.x Activity**

**12.50    USB 1.0 / 2.x Packet**

**12.51 USB 1.0 / 2.x PCS**

**12.52 USB 1.0 / 2.x PMA**

## 12.53 Undershoot

## 12.54 Upsample

## 12.55 Waterfall