

[SpikeGLX](#) / [Markdown](#) / [UserManual.md](#)



billkarsh Update help

 History

 1 contributor

Raw

Blame



1487 lines (1129 sloc) 59.7 KB

SpikeGLX User Manual

Topics:

- [Overview](#)
 - [Screen Saver and Power Settings](#)
 - [Installation and Setup](#)
 - [Calibration Data](#)
 - [Remote Command Servers](#)
 - [Data Directory](#)
 - [Multidrive Run Splitting](#)

- Data Stream
- Supported Streams
 - Stream Length
- Channel Naming and Ordering
- Output File Format and Tools
- Synchronization
 - Procedure to Calibrate Sample Rates
 - Running Without a Generator
 - Running With a Generator
 - Updating the Calibration
- Console Window
- Run Metrics Window
- Configure Acquisition Dialog
 - **Devices** -- Which Streams to Enable
 - Chassis Population
 - **IM Setup** -- Configuring Imec Probes
 - Which Probe?
 - Force Button
 - Copy Settings
 - IMRO Per Channel Settings
 - Acquisition Start Signal
 - **NI Setup** -- Configuring NI-DAQ Devices
 - Sample Clocks -- Synchronizing Hardware
 - Input Channel Strings
 - MN, MA Gain
 - AI Range
 - **Sync** -- Mapping Time Across Streams
 - Square Wave Source
 - Input Channels
 - Imec SMA Connector
 - Calibration Run
 - Measured Samples/s
 - **Gates** -- Carving Runs into Epochs
 - Run -> Gate -> Trigger
 - Gate Modes
 - Gate Manual Override

- **Triggers -- When to Write Output Files**
 - Trigger Modes
 - Continuation Runs
- **Maps**
 - Shank Map
 - Channel Map
- **Save**
 - Save Channel Subsets
- Graphs Window Tools
- Offline File Viewer
- Checksum Tools

Overview

Screen Saver and Power Settings

The following settings guard against interruption during prolonged data acquisition runs (running on batteries is discouraged):

Screen saver settings group:

- Screen saver: (None).

Note 1: The screen saver settings are a control panel and you can get there by typing 'screen saver' into a control panel search box. Screen saver is a program that draws entertaining animation on your otherwise black screen. The running of this class of programs disrupts acquisition. Turn that off.

Note 2: In the power settings you can set the time until the screen turns off. This is a safe option. It shouldn't affect anything except that you may have to log in again after the screen blanks.

Power plan settings:

Keep drilling down until you find the following advanced power plan options:

- Put the computer to sleep: Never.
- Hard disk/Turn off hard disk after: Never.
- Sleep/Sleep after: Never.
- Sleep/Allow wake timers: Disable.
- USB settings/USB selective suspend setting: Disable.

- Intel(R) Graphics Settings/Intel(R) Graphics Power Plan: Maximum Performance.
- PCI Express/Link State Power Management: Off.
- Processor power management/Minimum processor state: 100%.
- Processor power management/System cooling policy: Active.
- Processor power management/Maximum processor state: 100%.

Tip: For some settings, 'Never' might not appear as a choice. Try typing either 'never' or '0' directly into the box.

Installation and Setup

To install SpikeGLX on a new system, just copy a virgin SpikeGLX folder to your C-drive and double click SpikeGLX.exe to begin.

The contents of a virgin (see below) SpikeGLX folder:

```
SpikeGLX/  
  _Calibration/  
  _Help/  
  bearer/  
  iconengines/  
  imageformats/  
  platforms/  
  styles/  
  translations/  
  D3Dcompiler_47.dll  
  FpgaManager.dll  
  FTD3xx.dll  
  libEGL.dll  
  libgcc_s_seh-1.dll  
  libGLESV2.dll  
  libstdc++-6.dll  
  libwinpthread-1.dll  
  NeuropixAPI_version_info.dll  
  opengl32sw.dll  
  Qt5Core.dll  
  Qt5Gui.dll  
  Qt5Network.dll  
  Qt5OpenGL.dll  
  Qt5Svg.dll  
  Qt5Widgets.dll  
  SpikeGLX.exe  
  SpikeGLX_NISIM.exe
```

Virgin: The SpikeGLX folder does not contain a `_Configs` subfolder.

_Configs Folder

There are no hidden Registry settings or other components placed into your system folders. Your personal preferences and settings will be stored in `SpikeGLX/_Configs`.

If you give the software to someone else (please do), delete the `_Configs` folder because several settings in there are machine-dependent.

The `_Configs` folder is automatically created (as needed) when SpikeGLX is launched.

Tip: As you work with SpikeGLX you'll create several of your own custom files to remember preferred settings {channel mappings, Imec readout tables, ...}. **Resist the urge to store these in the SpikeGLX folder.** If you want to upgrade, and, **we will add cool features over time**, the clutter will make it much harder to figure out what you have to replace.

Calibration Data

Each imec probe has associated specific ADC and gain calibration data files. Place each probe's calibration data folder into the SpikeGLX `_Calibration` subfolder.

SpikeGLX reads an EEPROM chip on the probe to obtain its serial and model number. The serial number is used to look up the matching calibration folder name.

By the way, this subfolder also contains supplementary SpikeGLX data:

- The results of imec headstage sample rate calibration.
- The results of NI device sample rate calibration.

Remote Command Servers

Upon first launch SpikeGLX configures its **Remote Command** server and **Gate/Trigger** server with a local-host (loopback) address. Also, the Command server is disabled for security. You only need to think about these settings if you intend to use the remote control features over a network. In that case you need to visit their settings dialogs under the `Options` menu. In most cases, clicking the `My Address` button will set an appropriate interface (IP address) value, and you'll probably never need to alter the default port and timeout values.

Note: If your SpikeGLX address was assigned by a DNS service, it might change if other machines are added or removed on the network. Just click `My Address` again to read the updated value.

Data Directory

On first startup, the software will automatically create a directory called `C:/SGL_DATA` as a default output file storage location. Of course, the `C:/` drive is the worst possible choice, but it's the only drive we know you have. Please use menu item `Options/Choose Data Directory` to select an appropriate folder on your data drive.

You can store your data files anywhere you want. The menu item is a convenient way to "set it and forget it" for those who keep everything in one place. Alternatively, each time you configure a run you can revisit this choice on the `Save` tab of the `Configure Acquisition` dialog.

Multidrive Run Splitting

If you do very long recordings or use many probes you can distribute the data streams across multiple drives like this:

1. Select `Options/Choose Data Directory` .
2. Check the `Multidrive` box.
3. Add additional paths to the directory table.

The result is as follows:

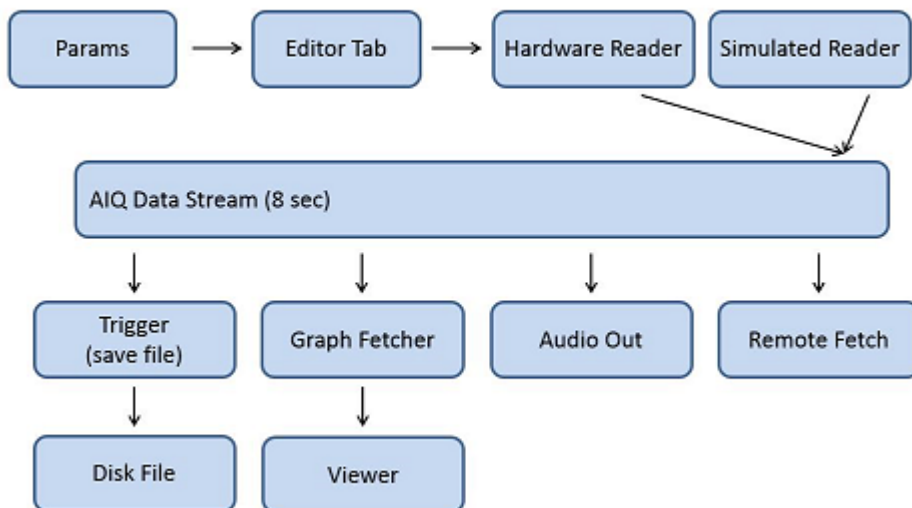
- There are N directories total = main + those in table.
- Logs and NI data are always written to the main directory (`dir-0`).
- Output for logical probe- j is written to `dir-(j mod N)` .

The mod operation is just the remainder when dividing j by N . For example, $7 \bmod 3 = 1$.

Data Stream

The following technical background will help you understand and configure your system, and help explain data storage formats. A key concept is the `data stream` which has several parts:

Stream Components (One stream per probe, one for NI)



On the **input** side, stream-specific hardware acquires data at its own characteristic sample rate and feeds that into a long stream buffer (FIFO queue). This happens in a reader thread .

The enqueued data are then available to other **output** threads:

- The trigger module scans the stream for conditions you've specified and then opens, writes and closes files accordingly.
- The graph fetcher pulls recent data from the stream and pushes it into that stream's custom viewer.
- The Audio module fetches recent data for the sound driver.
- The Command server fetches data from the stream on behalf of remote applications.

Notes:

Each stream gets its own metadata and binary data files.

The file saving **Trigger unit is shared** between streams so that data files are better time synchronized.

Likewise, the **GraphFetcher is shared** to facilitate synchronous data viewing.

More cores allow better load balancing among these activities.

Supported Streams

SpikeGLX supports multiple concurrent data streams that you can enable independently each time you run:

- `imec0` : Imec probe-0 data operating over PXIe.
- `imec1` : Imec probe-1 data operating over PXIe.
- ... : And so on. Each PXIe base station supports up to 4 probes.
- `nidq` : Whisper/NI-DAQ acquisition from USB peripherals or PCI cards.

Imec probes currently read out 384 channels of neural data and have 8 bits of status data (stored as a 16-bit SY word). Bit #0 signals that custom user FPGA code running on the Enclustra has detected an interesting neural event (NOT YET IMPLEMENTED). Bit #6 is the sync waveform, the other bits are error flags. Each probe is its own stream.

An Nidq device (M, X or S-series, a.k.a. 62xx, 63xx, 61xx) can be used to record auxiliary, usually non-neural, experiment signals. These devices offer several analog and digital channels. You can actually use two such devices if needed.

The Whisper system is a 32X multiplexer add-on that plugs into an NI device, giving you 256 input channels. Whisper requires S-series devices (61xx).

Stream Length

To allow fetching of peri-event context data the streams are sized to hold the smaller of {8 seconds of data, 40% of your available RAM}. We always generate a warning message with the length, like this: "*Stream length limited to 8 seconds.*" Making it a warning gives it a highlight color in the logs so you'll take notice of it.

Channel Naming and Ordering

Imec Channels

Each Imec stream acquires up to **three distinct types** of channels:

1. AP = 16-bit action potential channels
2. LF = 16-bit local field potential channels (some probes)
3. SY = The single 16-bit sync input channel

All probes read out 384 AP and a single SY channel. Some probes read out a separate LF band with the same channel count as the AP band.

Throughout the software the channels are maintained in `acquisition order`. That is, each acquired **sample** (or **timepoint**) contains all 384 AP channels, followed by the 384 LF channels (if present), followed by the SY channel.

The channels all have names with two (zero-based) indices, like this:

```
AP0;0 .. AP383;383 | LF0;384 .. LF383;767 | SY0;768
```

For example, LF1;385 tells you:

- This is an LF channel
- It's the second channel in the LF group
- It's the 386th channel overall

The second "overall" index (after the semicolon) is the index you should use for all GUI functions that select channels. For example:

- Which channel to observe in a TTL trigger.
- Which channel to send to audio output.
- Which channels to selectively save.

Imec Data Files Are Split

In memory, the LF channels are upsampled to 30kHz for symmetry with the AP channels. However, for better disk efficiency the AP and LF data are written out separately and the LF data have their natural sampling rate of 2.5kHz.

If you elected to save all channels `YourFile.imec0.ap.bin` would contain:

```
AP0;0 .. AP383;383 | SY0;768
```

and `YourFile.imec0.lf.bin` would contain:

```
LF0;384 .. LF383;767 | SY0;768
```

Note that the sync channel is duplicated into both files for alignment in your offline analyses. Note, too, that each binary file has a partner meta file.

NIDQ Channels

There are four categories of channels {MN, MA, XA, XD} and these are acquired and stored in that order, though they may be acquired from either one or two NI devices (named say, 'dev1 and 'dev2').

1. MN = dev1 multiplexed neural signed 16-bit channels
2. (likewise from dev2)
3. MA = dev1 multiplexed aux analog signed 16-bit channels

4. (likewise from dev2)
5. XA = dev1 non-muxed aux analog signed 16-bit channels
6. (likewise from dev2)
7. XD = dev1 non-muxed aux digital unsigned 16-bit words
8. (likewise from dev2)

Notes:

1. Within a multiplexed subgroup, like MN or MA, all the channels connected to a given multiplexer are grouped together. The names of the channels acquired from neural muxer #2 are "MN2C0"... "MN2C31". Zero-based labeling is used throughout.
2. If a second device is used, each MN, MA, ... category within the central stream is seamlessly expanded as if there were a single higher capacity device.
3. Channel names, e.g. "MA1C2;34" indicate both which channel this is within its own category (here, the 3rd channel in group MA1) and, which it is across all the channels in this stream (here, the 35th channel in the stream). The latter index (34) is how you should refer to this channel in save-strings, in trigger setups and for audio out selection.
4. Up to 32 digital lines can be acquired from your main device (say, dev1) and from a secondary device (say, dev2). The number of bytes needed to hold dev1's lines depends on the highest numbered line. If the highest named line is #31, then 32 bits are required, hence 4 bytes. If #14 is the highest, then 16 bits, hence 2 bytes are used to store the data for that device. Dev1 may need {0,1,2,3 or 4} bytes to hold its XD lines. Dev2 is evaluated the same way, but independently. In the stream, all the bytes for dev1 are together, followed by all those for dev2.
5. Trigger line numbering depends on bytes. Say XD1="0:4,22" and XD2="9." Suppose you want to use line #9 on dev2 as a TTL trigger input. You should specify bit #33, here's why: There are 6 bits used on dev1, but the highest is #22, so three bytes are needed. Therefore the offset to the first bit (bit #0) on dev2 is 24. Add 9 to that to get 33.
6. The streams, hence, graphs and data files, always hold an integral number of 16-bit fields. The bytes of digital data are likewise grouped into 16-bit words. There are anywhere from 0..4 bytes (B1) of dev1 lines followed by 0..4 bytes (B2) for dev2. The count of 16-bit words is $\text{int}((1 + B1 + B2)/2)$. That means, divide by 2 and truncate (round down) to an integer.

7. The Graphs Window depicts digital data words as groups of 16-lines. The lowest line number in a group is at the bottom. In files the data words have the lowest numbered lines in the lowest order bits.

Output File Format and Tools

Output data files are always paired; a `.bin` and a matching `.meta` file.

- The `.bin` file is the binary data. There is no header. The data are packed timepoints. Within each timepoint the 16-bit channels are packed and ordered exactly as described above in the section [Channel Naming and Ordering](#). Note that a timepoint is always a whole number of 16-bit words. There is one 16-bit word per analog channel. At the rear of the timepoint are digital lines, bundled together as the bits of 16-bit words as described in the notes above.
- The `.meta` data are text files in ".ini" file format. That is, every line has the pattern `tag=value`. All of the meta data entries are described in the document `Metadata.html` found at the top level of your release software download.

The SpikeGLX [Downloads Page](#) has simple tools (MATLAB and python) demonstrating how to parse the binary and metadata files.

Synchronization

Each stream has its own asynchronous hardware clock, hence, its own **start time** and **sample rate**. The time at which an event occurs, for example a spike or a TTL trigger, can be accurately mapped from one stream to another if we can accurately measure the stream timing parameters. SpikeGLX has several tools for that purpose:

- SpikeGLX: Clock rate calibration.
- SpikeGLX: Generate/acquire 1 Hz pulser signal.
- CatGT (offline tool): Extract event and pulser edge times.
- TPrime (offline tool): Align/map data using pulser edges.

Procedure to Calibrate Sample Rates

1. A pulse generator is configured to produce a square wave with period of 1 s and 50% duty cycle. You can provide your own source, or SpikeGLX can program the NI-DAQ device to make this signal, or the imec cards can be selected as the source.
2. You connect the output of the generator to one input channel of each stream and name these channels in the `sync` tab in the Configuration dialog.

3. In the `sync` tab you check the box to do a calibration run. This will automatically acquire and analyze data appropriate to measuring the sample rates of each enabled stream. These rates are posted for you in the `sync` tab for use in subsequent runs.

Full detail on the procedure is found in the help for the Configuration dialog's [Sync tab](#).

Running Without a Generator

You really should run the sample rate calibration procedure at least once to have a reasonable idea of the actual sample rates of your specific hardware. In our experience, the actual rate of an imec stream may be 30,000.60 Hz, whereas the advertised rate is 30 kHz. That's a difference of 2160 samples or 72 msec of cumulative error per hour that is correctible by doing this calibration.

The other required datum is the stream start time. SpikeGLX records the wall time that each stream's hardware is commanded to begin acquiring data. However, that doesn't account for the time it takes the command to be transmitted to the device, to be decoded, to be responded to, and for the first data to actually arrive at the device. This estimate of the start time is only good to about 10 ms.

It is an option to do your data taking runs without a connected square wave generator, and you might choose that if you only have one stream, or if the sync hardware is malfunctioning for any reason. Under these conditions runs will start off with time synchronization errors of 5 to 10 ms (owing to T-zero error) and that error will slowly drift depending upon how accurate the rate calibration is and whether the stream has clock drift that isn't captured by a simple rate constant. Thankfully, you can do much better than that...

Running With a Generator

In this mode of operation, you've previously done a calibration run to get good estimators of the rates, and you are dedicating a channel in each stream to the common generator during regular data runs. Two things happen under these conditions:

1. When the run is starting up SpikeGLX uses the pulser to adjust the estimated stream start times so they agree to within a millisecond.
2. During the run, the time coordinate of any event can be referenced to the nearest pulser edge which is no more than one second away, and that allows times to be mapped with sub-millisecond accuracy.

Updating the Calibration

Menu item: `Tools/Sample Rates From Run` lets you open any existing run that was acquired with a connected generator and recalibrate the rates for those streams. You can then elect to update the stated sample rates within this run's metadata, and/or update the global settings for use in the next run.

Console Window

The `console` window contains the application's menu bar. The large text field ("Log") is a running history of informative messages: errors, warnings, current status, names of completed files, and so on. Of special note is the status bar at the bottom edge of the window. During a run this shows the current gate/trigger indices and the current file writing efficiency, which is a key readout of system stability.

Acquisition Performance

The Imec hardware buffers a small amount data per probe. A fast running loop in SpikeGLX requests packets of probe data and marshals them into the central stream. Every few seconds we read how full the hardware buffer is. If it is more than 5% full we make a report in the console log like this:

```
IMEC FIFO queue 5 fill% 6.2
```

If the queue grows a little it's not a problem unless the percentage exceeds 95%, at which point the run is automatically stopped.

Disk Performance

During file writing the status bar displays a message like this:

```
FileQFill%=(0.1,0.0) MB/s=14.5 (14.2 req)
```

The `imec` and `nidq` streams each have an in-memory queue of data waiting to be spooled to disk. The `FileQFill%` is how full each binary file queue is (`imec,nidq`). The queues may fill a little if you run other apps or copy data to/from the disk during a run. That's not a problem as long as the percentage falls again before hitting 95%, at which point the run is automatically stopped.

In addition, we show the overall current write speed and the minimum speed **required** to keep up. The current write speed may fluctuate a little but that's not a problem as long as the average is close to the required value.

You are encouraged to keep this window parked where you can easily see these very useful experiment readouts.

Tools

- Control report verbosity with menu item `Tools/Verbose Log` .
- Enable/disable log annotation with menu item `Tools/Edit Log` .
- Capture recent log entries to a file with menu item `Tools/Save Log File` .

Run Metrics Window

Choose menu item `Window/Run Metrics` to open a window that consolidates the most vital health statistics from the Console log, and adds a few more:

- An overall run health summary LED.
- Each Imec probe's error status flags.
- Each Imec probe's FIFO filling level.
- Each Imec probe's worker thread activity level.
- Imec and NI disk writing performance.
- Each stream's data fetching performance.
- Errors and warnings culled from the Console log.

Click the `Help` button in the window to get a detailed description of the metrics.

Configure Acquisition Dialog

Notes on the dialog as a whole:

- Settings are divided into subgroups on the various tabs. Validation (a.k.a. sanity checking) is always performed on all of the settings on all of the tabs. Validated settings are stored in `SpikeGLX/_Configs/daq.ini` .
- Press `Last Saved` to revert the entire dialog to the values in `daq.ini` .
- Press `Verify | Save` to sanity-check the settings on all tabs, and if valid, save them to `daq.ini` **without** initiating a new run. This is useful when trying to make the Configuration and Audio dialog settings agree before starting a run, as audio settings are checked against `daq.ini` .
- Press `Run` to validate and save the settings to `daq.ini` and then start a new run.

- Press `cancel` to end the dialog session without further altering `daq.ini`.

Devices -- Which Streams to Enable

Each time you visit the Configuration dialog you must go through the `Devices` Tab and tell us which subsystems you want to use (enable). You also have to press the `Detect` button which detects the hardware that's actually connected. This allows the software to apply appropriate sanity checks to your settings choices.

Chassis Population

Each Imec probe plugs into a headstage (HS). Up to four HS plug into the four ports (numbered 1,2,3,4) of a base station (BS). Each BS plugs into a slot in your PXIe chassis. Slot one of a PXI chassis is always the computer interface device, while slots 2-8 can be used for Imec or other devices.

You can place your Imec BS cards in any PXIe compatible slot. Use the `Add` and `Remove` buttons to specify which slots are actually populated.

Each BS/slot accommodates up to 4 probes. Use the `Enable` checkboxes in the table entries to specify which probes to configure and run.

Neuropixels 2.0:

You can plug either one or two probes into the `docks` of a NP 2.0 HS. Dock #1 is on the side of the HS populated by two large capacitors and the Omnetics connector. Dock #2 is on the backside with the HS label.

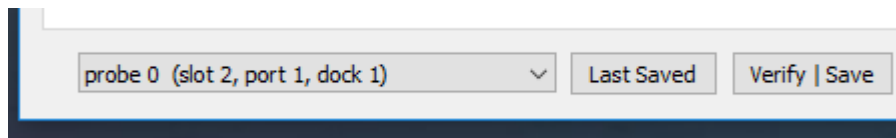
IM Setup -- Configuring Imec Probes

Which Probe?

After you specify which physical slots, ports and docks are enabled and click `Detect` on the `Devices` tab, the table will assign each probe a `logical ID` number in a simple manner, slot by slot, then port by port, then dock by dock.

For example, if you enabled (slot,port,dock): (2,2,2), (2,3,1), (4,1,1), (4,1,2), they would get logical probe #s: (2,2,2)=0, (2,3,1)=1, (4,1,1)=2, (4,1,2)=3.

SpikeGLX primarily uses the zero-based logical probe ID to reference probes.



At the bottom-left of the dialog is a pop-up menu listing the logical probes. All of the tabs refer to the currently selected probe. Whichever tab you are currently working in, you can use the pop-up to change to any enabled probe.

Force Button

Use this button in the `Current probe` box at the top-left of the IM Setup tab to override the serial number or the part number stored in the probe EEPROM chip. Only do this if the EEPROM suffered damage from an electrical discharge event and now reads back incorrect identity data.

Copy Settings

Use the controls in this box to copy all of the configuration settings from one probe to another, or to all probes as a handy shortcut. You can always visit each probe separately using the probe selector pop-up and make choices specific to that probe.

IMRO Per Channel Settings

Click `Edit` in the `Configuration and calibration` item group to open a simple editor that lets you load/save/edit a text file that specifies all the choices you can make for each of the (up to) 384 readout channels of the probe. The text file has extension `(.imro) = Imec readout`.

The big table on the left of the editor shows a row for each readout channel. You can type directly into this table to make choices just for that channel.

To the right of the table are the **set all** controls. As a convenience you can assign a given value to all channels at once with these helpers.

Notes:

A relatively new 'set all' control allows you to position a 384-channel block anywhere along the length of the shank:

1. First decide where the center of interest/activity is on the probe. You can set all banks to zero, one or two, or load a preset checkerboard pattern <https://billkarsh.github.io/SpikeGLX/> under the Interesting Map Files section. Start a run and open the Shank Viewer to see activity. Hover the cursor over the shank to read out a row number at the center of your region of interest. Write it down.
2. Configure a fresh run, and in the IMRO Editor type the row number into the Row box and click Center Here . This will select a continuous block of electrodes centered at that row by assigning the right bank numbers for you.
3. Optionally, to make your graph ordering reflect the electrode assignments, visit the Maps tab . In the Channel mapping box click Edit for the IM probe. Then in the Map dialog, select Shank: Bottom to top in the Auto-arrange menu and click Apply . Now save this channel map. Remember that a channel map is only used to order the graphs in the viewers, and that to make it take effect, you have to click the sort button in the Graphs window panel so it reads Usr Order .

Finally, remember that you can rapidly make changes to the IMRO table and graph ordering (Channel Map) while running by right-clicking in the Graph traces and selecting that editing option. Those changes stick in the settings. The only downside is this on-the-fly editing messes up synchronization. When you like how things are set up, start a fresh run. This will automatically load the last settings and reinitialize sync.

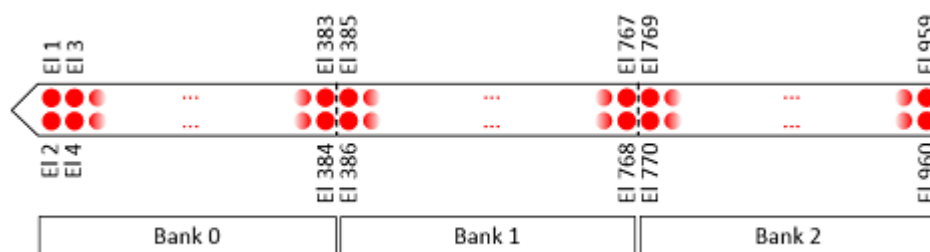
Save the file!

If you choose anything other than default settings, you must save the table as a named file.

Bank

The probes use switches to select the bank that each readout channel is connected to. The relationship is this:

- $\text{electrode} = (\text{channel} + 1) + \text{bank} * 384$; $\text{electrode} \leq 960$



Refid

There are five reference choices [0..4] for each channel.

Refid	Referenced channel
0	external
1	tip electrode
2	electrode 192
3	electrode 576
4	electrode 960

Gain

Each readout channel can be assigned a gain factor for the AP and the LF band. The choices are:

50, 125, 250, 500, 1000, 1500, 2000, 3000

AP Filter

Imec channels are separated into two filtered bands as follows:

- LF: [0.5..1k]Hz (fixed).
- AP: [{0,300}..10k]Hz (selectable high pass).

Acquisition Start Signal

Initially we are only supporting software-based initiation of imec data streaming. That is, the data acquisition starts manually when you click the `Run` button in the configuration dialog.

NI Setup -- Configuring NI-DAQ Devices

Sample Clocks -- Synchronizing Hardware

For all modes:

- A secondary device, if used, *always needs an external clock source*, and that source must always be the same clock that drives the primary. This is the only way to coordinate the two devices. The NI breakout boxes, like `BNC-2110` or `SCB-68A`, make this simple.

- You always need to select a clock source in the `Timing` box at the bottom of the panel. Your selection specifies two key values. (1) It specifies a `Set_sample_rate` (read directly in the menu control) that names the nominal rate for the source, and is used to program your NI device if `Primary clock line` is set to `Internal`. (2) Your selection looks up the `Measured_sample_rate` for this device and enters that on the `Sync` tab of the dialog.
- You will get the best possible alignment of data across your files if you use the calibration features on the `Sync` tab to measure the true sample rates of your devices.
- There is a `Start` checkbox and a selectable digital output line. If enabled, when the run starts, the selected line goes from low to high and stays high until the run is stopped. This is always an option you can use to hardware-trigger other components in your experiment. (*Whisper systems require this signal on line0*).

{Clock, Muxing, Sample Rate} choices depend upon your hardware--

Case A: Internal Clock Source, No Multiplexing

In this simple case, there is no external sample clock. Rather, you can set the NI device to generate its own sample clock waveform. Note that an NI device can achieve a precise value only if it evenly divides the master clock rate. The master rate for a 6133 is 20 MHz which is divisible by 40000 but not by 30000, for example. Although generated internally, the clock source is also routed to an external terminal so you can share it with other hardware:

- On multifunction IO devices, the output is available at `CtrlOutput/pin-2/PFI-12`.
- On digital IO devices like the 6535, the output is at the 1st PFI terminal listed in the primary clock line menu, which is usually `pin-67/PFI-5`.

Settings:

- Set primary clock line = `Internal`.
- *Optionally* connect a wire from the primary clock terminal to a selected `PFI` terminal on the secondary device.
- Leave the `MN` and `MA` channel boxes blank.
- Specify input channels in the `XA` and `XD` boxes.
- `Chans/muxer` is ignored for these channels.

Case B: External Clock Source, No Multiplexing

In this case, the sample clock is being driven by some component in your setup, other than the primary NI device. Follow these steps:

- Set primary clock line = PFI terminal.
- Connect external clock source to that terminal.
- *Optionally* connect same external clock to a secondary device PFI terminal.
- Leave the MN and MA channel boxes blank.
- Specify input channels in the XA and XD boxes.
- `Chans/muxer` is ignored for these channels.

Case C: Whisper Multiplexer

If you specify any MN or MA input channels, the dialog logic assumes you have a Whisper and automatically forces these settings:

- Primary clock line = `PFI2`.
- Start sync signal enabled on digital `line0`.

You must manually set these:

- Set `Chans/muxer` to 16 or 32 according to your Whisper data sheet.

Case D: Whisper with Second Device

Follow instructions for Whisper in Case C. In addition:

1. In the secondary box, select a PFI terminal for the clock.
2. Connect the "Sample Clock" output BNC from the Whisper to the selected PFI terminal on the NI breakout box for the secondary device.

Note that the BNC should be supplying the multiplexed clock rate: $(\text{nominal sample rate}) \times (\text{muxing factor})$.

Input Channel Strings

The `{MN, MA, XA}` input fields work just like the page range field in a print dialog. For example, "0,2:4,6" means you're using NI analog-input (AI) channels `{0,2,3,4,6}`. If you are not using a particular category, like MA, then leave that field empty.

Remember that channel ordering is important. In the central stream we want all the MN channels (if any) to come first, followed by the MA, then XA then XD channels. These groupings allow the software to know how to process the channels: what gain to apply, what type of filter to use, what to name it, and so on. When we acquire AI data from an NI card it returns data from channel AI0 (if any) then from AI1, and so on. **Therefore**, we require you to plug your neural multiplexers (MN's) into the lowest numbered AI channels, then populate the next AI channels with your MA muxers, then plug in any XA lines. For example, this is legal:

```
MN = 2,4 // don't have to start at zero, gaps are okay
MA = 5   // MA comes AFTER MN
XA = 7   // XA strictly AFTER MN and MA
```

Note: The cabling of the Whisper system automatically routes multiplexers to the proper AI channels, but to set up the dialog **you** still have to know that your Whisper box has neural muxers on AI channels 0:5 and aux multiplexers on channels 6,7 (for example).

Digital Strings

As with AI channel strings, the XD field takes a range string like "0,2:4,6:7" but in this case the values are digital line numbers.

We support up to 32 lines from each device, so legal line numbers are [0..31]. Note that Whisper systems reserve line #0 as an output line that commands the Whisper to start.

MN, MA Gain

The multiplexers in your system may have a gain factor. In a typical Whisper box, the MN channels route through Intan chips with a gain of 200 and the MA channels are handled by a unity gain muxer (see the data sheet for your hardware).

The gain values you enter here **do not** affect values recorded in disk files. They are only applied in the Graph window so that familiar, unamplified voltages are plotted and reported in the graph statistics.

Note that some trigger modes ask you to specify a threshold voltage. The value you enter for a threshold should always be what you read directly from the graph of that channel. The software will make any necessary gain adjustments.

AI Range

NI devices let you configure an expected voltage range for an analog channel, say [-2.5..2.5] volts. The purpose of this is to improve your dynamic range, a.k.a. voltage resolution. If you know in advance that none of your voltages will exceed 2.1 volts, then choosing [-2.5..2.5] is better than [-5..5] because you'll get twice the resolution in your measurements. However, the value 3.0V would be pinned (saturated) at 2.5V which is bad. In that case, [-5..5] is a safer choice. Generally, choose the smallest range compatible with your instrument specifications.

Note that other components in the chain may impose their own voltage restrictions. For example, some MA channel banks on some Whisper models saturate at 2.5V. It would be a bad idea to use such channels to read an instrument making output in the range [0..3.3] volts.

Sync -- Mapping Time Across Streams

Note that the imec BS cards have a single SMA connector on the front panel and that the Neuropixels User Manual explains that this connector is multipurpose, available alternatively for SYNC operation or for communication of a hardware start (what they term 'trigger') signal. **However**, in SpikeGLX we employ the SMA exclusively for SYNC duty.

Square Wave Source

Choose `Disable sync waveform` to run without active alignment to edges of a common square wave. The software will still apply the measured sample rates stated in the boxes at the bottom of this tab.

Otherwise, any generator source should be programmed to form a simple square wave with a 1 second period and 50% duty cycle. You have three choices for the generator:

- `Separate high precision pulser` allows you to provide any waveform generator you like for the purpose.
- `Current NI acquisition device` will program your multifunction NI device to produce the required waveform at terminal `Ctr1Out (pin-40/PFI-13)`. (Brian Barbarits will make a simple breakout connector available to allow Whisper users to access this signal.) Digital devices like the 6535 are programmed to make the waveform on `line-0`. In all cases you have to run a wire from this terminal to the appropriate channel input connector.
- `Imec slot N` will program the indicated BS to produce the sync waveform and make it available at its SMA connector.

Whichever pulser source you select, the programmed (set) period is 1 second, but the actual period may differ from that. If you have measured the actual period of the generator's output with a high precision frequency/period analyzer, then enter that value in the `Measured period` box. If you have not measured it, enter 1 in the box.

Input Channels

Imec SMA Connector

If an Imec BS is selected as the source, all Imec BS modules in the chassis will automatically get the signal via the chassis backplane.

For all other menu choices the imec modules are programmed to share the specified input SMA. If you are doing waveform-based sync then you **MUST** connect the waveform source to this input. If you have chosen `Disable sync waveform` then you may optionally connect any TTL signal you want to the selected SMA and that signal will be recorded as bit #6 of the SY channel in all streams. This connection can be used as a TTL trigger input if waveform sync is not being used.

The SMA connector is compatible with 0-5V TTL signals.

Nidq

At this time, Whisper boxes do not allow access to digital inputs, so you must connect the square wave to one of the multiplexed auxiliary analog inputs (MA).

Calibration Run

To do a run that is customized for sample rate calibration, check the box in this item group and select a run duration. We can't tell you how long is optimal. That depends upon how stable the clocks are and you can see that for yourself by repeating this measurement to see how it changes over time. It's probably a good idea to turn on power and let the devices approach a stable operating temperature before calibrating. We can't tell you how long that should be either. Our typical practice is 30 minutes of warm up and 30 minutes of measurement time.

A calibration run will save data files to the current run directory specified on the `Save tab`. The files will automatically be named `CalSRate_date&time_g0_t0...`

Measured Samples/s

When you do a calibration run (and it is successful) the results are stored into your `_Calibration` folder. The next time you configure a run the results will automatically appear in these boxes.

Tabulated device calibration values are stored in the files:

- `_Calibration\calibrated_sample_rates_imec.ini`
- `_Calibration\calibrated_sample_rates_nidq.ini`

You can manually edit these if needed, say, if you've swapped equipment and already know correct rates from previous measurements.

To give you a sense of how much measured values differ from the nominal rates, here's what we typically get:

- IM: 30000.083871
- NI: 25000.127240.

Gates -- Carving Runs into Epochs

Run -> Gate -> Trigger

The hierarchical `run/gate/trigger` scheme provides several options for carving an experiment "run" into labeled epochs with their own data files. The terms "gate" and "trigger" were chosen because they are "Biology neutral". You decide if epochs are really 'windows', 'events', 'trials', 'sessions' or other relevant contexts.

1. You configure experiment parameters, including a `data folder` where all the output files will be stored, a `run name`, a `gate method` and a `trigger method`.
2. You start the run with the `Run` button in the Configure dialog. That means the data acquisition devices begin collecting scans and the Graphs window begins displaying streaming data. *(A run can also be configured and/or started using TCP/IP from a remote application).*
3. Initially, the gate is low (closed, disabled) and no files can be written. When the selected gate criterion is met the gate goes high (opens, enables), the gate index `g` is set to zero and the trigger criteria are then evaluated.

4. Triggers determine when to capture data to files. There are several options discussed more fully in the next section. Triggers act only within a gate-high epoch and are terminated if the gate goes low. **Gates always override triggers.** Each time the gate goes high the gate-index (g) is incremented and the t -index is reset to zero. The trigger program is run again within the new gate window.
5. When the selected trigger condition is met, a new file is created. On creation of the first file with a given g index, a new `run` folder is created in the data folder to hold all the data for that gate. That is, we create folder `data-path/run-name_gN`. Thus, the first file written for the first trigger would be `data-path/run-name_g0/run-name_g0_t0.nidq.bin`. When the trigger goes low the file is finalized/closed. If the selected trigger is a repeating type and if the gate is still high then the next trigger will begin file `data-path/run-name_g0/run-name_g0_t1.nidq.bin`, and so on within gate zero. (For Imec data streams, the same naming rule applies, with `nidq` replaced by `imec0.ap` and/or `imec0.lf`).
6. If the gate is closed and then reopened, triggering resets and the next folder/file will be named `data-path/run-name_g1/run-name_g1_t0.nidq.bin`, and so on.
7. The run itself is always stopped manually, either from the SpikeGLX GUI or from a remote application.

Note that there is an option on the `Save` tab called `Folder per probe`. If this is set, there is still a `run` folder for each g -index `run-name_gN`. However, inside that there is also a subfolder for each probe that contains all the t -indices for that g -index and that probe. A probe subfolder is named like `run-name_gN/run-name_gN_imecM`.

Gate Modes

At present we have provided two simple gate methods:

- `Immediate Start`. As soon as the run starts the gate is immediately set high and it simply stays high ("latches high" in electronics lingo) until the run is stopped.
- `Remote Controlled Start and Stop`. SpikeGLX contains a "Gate/Trigger" server that listens via TCP/IP for connections from remote applications (like StimGL) and accepts simple commands: {SETGATE 1, SETGATE 0}.

Gate Manual Override

You can optionally pause and resume the normal **gate/trigger** processing which is useful if you just want to view the incoming data without writing files or if you want to ability to stop an experiment and restart it quickly with a new run name or changed gate/trigger indices. See the discussion for [continuation runs](#). To enable manual override:

On the Configure Dialog/Gates Tab check Show enable/disable recording button . The button will appear on the Graphs Window main Run Toolbar at run time. *Recently we've made manual override the default setting.*

If this button is shown you also have the option of setting the initial triggering state of a new run to disabled or enabled.

Warning: If you opt to show the button and to disable triggering, file writing will not begin until you press the Enable button.

Warning: If you opt to show the button there is a danger of a run being paused inadvertently. That's why it's an option.

Triggers -- When to Write Output Files

Rule 1: A file is being written when the trigger is high.

Rule 2: Every binary (.bin) file has a matching (.meta) file.

To capture final checksum and size, the metadata are overwritten when the binary file is closed.

Trigger Modes

- **Immediate Start** . As soon as the run starts the trigger is immediately set high and it simply stays high ("latches high") until the run is stopped. Select 'Immediate' mode for the gate and trigger modes to start recording immediately without fussing about experiment contexts.
- **Timed Start and Stop** . First, *optionally* wait L0 seconds, **then**:
 - Latch high until the gate closes, or,
 - Perform a sequence:
 - Write for H seconds.
 - Idle for L seconds.
 - Repeat sequence N times, or until gate closes.
- **TTL Controlled Start and Stop** . Watch a selected (analog or digital) channel for a positive going threshold crossing, **then**:

- Latch high until the gate closes, or,
 - Write for H seconds, or,
 - Write until channel goes low.
 - *Latter 2 cases get flexible repeat options.*
 - *Threshold detection is applied to unfiltered data.*
- Spike Detection Start and Stop . Watch a selected channel for a negative going threshold crossing, **then**:
 - Record a given peri-event window about that to its own file.
 - Repeat as often as desired, with optional refractory period.
 - *Threshold detection is applied post 300Hz high-pass filter.*
 - Remote Controlled Start and Stop . SpikeGLX contains a "Gate/Trigger" server that listens via TCP/IP for connections from remote applications (like StimGL) and accepts simple commands: {SETTRIG 1, SETTRIG 0}.

Normally an NI device is used for TTL inputs, but the [imec SMA connector](#) can be used in special circumstances.

Note that some trigger modes ask you to specify a threshold voltage. The value you enter should be the real-world voltage presented to the sensor. It's the same value you read from our graphs. For example, a threshold for neural spikes might be -100 μ V; that's what you should enter regardless of the gain applied.

Continuation Runs

You can set the name for your run either in the Configure dialog or in the run toolbar at the top of the Graphs window. In the Graphs window you must first pause writing via the optional `Enable/Disable Recording` button as described [here](#).

Both of these inputs accept either an undecorated base-name for a run, or a decorated name of the form `run-name_gN_tM` . The decorated form tells SpikeGLX you wish to continue writing additional files into an existing run, starting at the specified g/t indices. This is very useful if a run had to be interrupted to repair a problem.

Note too, that you can change to a different run name without stopping the run. Use the Graphs window `Disable` button to pause writing, then type in a new undecorated name (no g/t indices). This will be treated as a request for a brand new run name that will start at `_g0_t0`.

Maps

Shank Map

A `shank map` is a table describing where each neural channel is on your physical probe. This information is used for spatial channel averaging, and for activity visualization.

A rudimentary tool is provided to create, edit and save shank maps (and shank map (.smp) files). Click `Edit` in the `Shank mapping` group box for the map of interest.

If you do not supply a map, the `default` layout depends upon the stream. The `imec` default layout is a probe with 1 shank, 2 columns, and a row count determined from your actual probe type and `imro` table choices. The `nidq` default is a probe with 1 shank, 2 columns and a row count equal to $MN/2$ (neural channel count / [2 columns]).

To make and use a custom map you must save it in a file. The file format looks like this:

```
1,2,480 // header: nShanks,nColsPerShank,nRowsPerShank
0 0 0 1 // entry: iShank <space> iCol <space> iRow <space> iUsed
0 1 0 1
0 0 1 1
0 1 1 1
... // one entry per spiking acquisition channel
```

This universal layout scheme has a few simple rules:

- Number of shanks is [1..8].
- Each shank has (nCols x nRows) sites.
- Shank indices (iShank) start at 0 and advance left to right.
- Column indices (iCol) start at 0 and advance left to right.
- Row indices (iRow) start at 0 and advance tip to tail.
- Each site has a `used` index (Boolean 0 or 1) denoting inclusion in spatial averages.

You can mark a site `used=0` if you know it is broken or disconnected. For `Imec` probes, we automatically set `used=0` for reference sites and those you have turned off in the `IM Setup` tab.

Most importantly a shank map is a mapping from an acquisition channel to a probe location. So...while there can be more potential sites (nShanks x nCols x nRows) than channels...

- The number of table entries must equal AP (if `Imec`) or MN (if `Nidq`).

Channel Map

The Graphs window arranges the channels in the standard acquisition order (AP, LF, SY) and (MN, MA, XA, XD) or in a user order that you can specify using a channel Map. Each stream gets its own map file.

A rudimentary tool is provided to create, edit and save channel maps (and channel map (.cmp) files). Click `Edit` in the Channel mapping group box for the map of interest.

If you do not supply a map, the default user order depends upon the stream. The `imec` default order follows the shank map, ordered first by shank, then going upward from tip to base. The `nidq` default is acquisition channel ordered.

To make and use a custom map you must save it in a file. The file format looks like this:

```
6,2,32,0,1 // header (type counts): MN,MA,C,XA,XD
MN0C0;0 256 // entry: channel-name;acq-index <space> sort-index
MN0C1;1 1
MN0C2;2 2
MN0C3;3 3
...
XD0;256 0 // this example makes the digital graph first
```

You can save and reuse channel map files in another run by loading that file from the Channel Map dialog. However, this only makes sense if the loaded map describes the same types and counts of channels as you've configured in the current run, hence, the header values, which are counts of channel types. The `c` value is the number of channels per muxer.

Editing the sort order simply consists of reordering the right-most column of sort-index values which must be in the range $[0..N-1]$, where N is the total channel count. For digital data we don't count individual lines. Rather we count 16-line blocks of channels.

You can edit these files in any text editor if you prefer. You can change channel name strings too (shh).

You can also change the channel map from the Graphs window by right-clicking on the graphs area and selecting `Edit Channel Order...`

Save

Save Channel Subsets

The hardware configuration tabs determine which channels are acquired from the hardware and held in the central data stream. All acquired channels are shown in the Graphs window. However, you don't have to save all of the channels to your disk files.

You can enter a print-page-range style string for the subset of channels that you want to save. This string is composed of index numbers in the range [0..N-1], where N is the total channel count. To save all channels you can use the shorthand string `all` , or just `*` .

Notes:

1. You can also change this list from the Graphs window by right-clicking on the graphs area and selecting `Edit Saved Channels...` .
2. Remember that digital lines are grouped into 16-bit words which are essentially pseudo-channels in your subset string. If you don't save a given word of digital line data, several lines will be affected.

Graphs Window Tools

Second Graphs Window

In the Console window menus choose `Window/More Traces (Ctrl+T)` to open a second Graphs window after a run has started. Only the main Graphs window has run controls and LED indicators for gate and trigger status, but all of the stream viewing and filtering options otherwise work the same in both windows.

Run Toolbar

- `Stop Acquisition` : Stops the current run and returns the software to an idle state. You can do the same thing by clicking the `Graph Window's Close box` or by pressing the `esc` key, or by choosing `Quit (control-Q)` from the File menu (of course the latter also closes SpikeGLX).
- `Acquisition Clock` : The left-hand clock displays time elapsed since the run started and first samples were read from the hardware.
- `Enable/Disable Recording` : This feature is available if you select it on the `Gates tab` . Use this to [pause/resume](#) the saving of data files, to change [which channels](#) are being written to disk files, or to [edit the name](#) of the run and its disk files.
- `Recording Clock` : The right-hand clock displays time elapsed since the current file set was opened for recording.

- **Pause** : The Pause VCR-style button toggles between pause and play of stream data in the graphs so you can inspect an interesting feature. This does not pause any other activity.

Stream Toolbar Controls

- **Channel Name** : The name of the currently selected graph. Single-click on any graph to select it. The current graph is the target of several controls such as the axis scaling boxes. Click on the channel-name string in the toolbar to switch to the page containing the current selection.
- **Expand** : This button toggles between showing just the selected channel, and the standard multi-channel view. You can also double-click a graph to select it and expand/contract it.
- **Sec** : Enter a value in range [0.001..30.0] seconds to set the time span.
- **Ysc1** : Enter a value in range [0.0..9999.0] to set a vertical magnification for the selected graph.
- **C1r** : Click this button to get a Color picking dialog whereby you can define an alternate color for the **data trace** in the selected graph. This only works for analog channels; digital traces are auto-colored.
- **Apply All** : Copies Ysc1 from the selected graph to all other graphs of the same category.

Filters Applied Only to Neural Channels

Notes:

- These filters only affect the appearance of graphs, not saved data.
- These filters are **not** applied to non-neural channels.
- If ever you are suspicious that hardware is not working, turn all the filters off to understand what is coming out of the hardware.

General Filters

- **-<T>** : Time averaging. Samples the data stream per channel to calculate and then subtract the time average value; effectively subtracting the DC component. The value is updated every 5 seconds. This may create artifactual steps during the initial settling phase of Imec preamps.

- `-<S>` : Spatial averaging. At each timepoint a neighborhood of electrodes per channel is averaged; the result is subtracted from that channel. The locations of electrodes are known from your shank map.
 - Notes:
 - a. Certain electrodes are omitted from the average: {Those marked 'use=false' in your map, Imec reference electrodes, Imec electrodes that are turned off}.
 - b. Only AP-band channels are affected.
 - c. Neighborhoods never cross shank boundaries.
 - There are four choices of neighborhood:
 - `Loc 1,2` : Small annulus about the channel's electrode; e.g. inner radius=1, outer=2.
 - `Loc 2,8` : Larger annulus about the channel's electrode; e.g. inner radius=2, outer=8.
 - `G1b A11` : All electrodes on probe.
 - `G1b Dmx` : All electrodes on probe that are sampled concurrently (same multiplexing phase).
- `BinMax` : If checked, we report the extrema in each neural channel downsample bin. This assists spike visualization but exaggerates apparent background noise. Uncheck the box to visualize noise more accurately.

Imec Stream Filters

The `imro` table determines individually for each AP channel if a 300 Hz high pass filter is applied in hardware. The result is the `native` AP signal. In addition, LF band signals are always acquired from the hardware. The filter selector applies software operations to these `native` signals.

- `AP Native` : No software filter is applied.
- `300 - INF` : A software high pass filter is applied to all AP channels.
- `AP = AP+LF` : All AP channels are graphed as the simple sum of the AP and corresponding LF band signal.

Nidq Stream Filters

- `Bandpass` : Applies optional bandpass filtering to neural MN channels.

Page Toolbar Controls

- `Acq/Usr Order` : This button toggles between acquired (standard) channel order and that specified by your custom [channel map](#).

- `ShankView` : Opens the ShankViewer window for this stream.
- `NChan` : Specifies how many graphs to show per page. When the value is changed SpikeGLX selects a page that keeps the middle graph visible.
- `1st` : Shows the index number of the first graph on the current page.
- `Slider` : Change pages.

Right-Click on Graph

For either stream:

- `Select As L/R Audio Channel` : Listen to selected channel (immediately).
- `Edit Channel Order...` : Edit the ChanMap for this stream.
- `Edit Saved Channels...` : Edit the string describing which channels are saved to file. Note that saved channels are marked with an `s` on the right-hand sides of the graphs.
- `Color TTL Events...` : Watch up to 4 auxiliary (TTL) channels for pulses. Apply color stripes to the graphs when pulses occur.

Imec Menu

- `Edit Banks, Refs, Gains...` : Shows the Imro Table editor allowing quick settings changes (available only if recording currently disabled). If you make changes to the bank values the software will automatically generate a new default ShankMap reflecting your current electrode, reference and on/off channel selections.
- `Edit Channel On/Off...` : Shows editor for changing which channels are turned off (available only if recording currently disabled). The current ShankMap is updated to reflect your changes.

*Note: In the Configuration dialog ChanMap Editor you can order the graphs according to the extant ShankMap before the run starts. This is **not** changed even if we dynamically alter the ShankMap in these R-click actions.*

Other Graph Window Features

- Hover the mouse over a graph to view statistics for the data currently shown.

Offline File Viewer

Choose `File::Open File Viewer` and then select a `*.bin` file to open. As of version 20160701 you can open and view data files from any stream, and you can link the files from a run so scrolling is synchronized between multiple viewers.

In a viewer window, choose `Help::File Viewer Help` for more details.

Checksum Tools

SHA1 Checksum

Each `.meta` file stores the SHA1 checksum for the binary file in the field `fileSHA1=`. Use menu item `Tools/Verify SHA1` to recalculate the current value for any `(.bin,.meta)` pair and determine if either file may have been corrupted. The SHA1 checksum, per se, does not provide any pathway to recovery.

PAR2 Redundancy Tool

Of course, you can create a perfect backup of a file by simply copying it whole, and that's the recommended thing to do provided you can afford the storage space.

Alternatively, *Parity ARchive 2* is a Usenet format for detecting and correcting binary file corruption using only a fraction of the original file's size. (That fraction is called the redundancy percentage.) The downside is that the smaller the fraction you use for the backup set, the lower the likelihood of being able to fully recover the original file.

To invoke the tool use menu item `Tools/PAR2 Redundancy Tool` to create a backup set for a given data file. Subsequently, using the same tool, you can use the backup set to verify the file and to attempt recovery in case of corruption.

fin