Intan.jl Documentation

Release 0.1

Paul Thompson

Nov 26, 2018

Installation

1 Installation Overview **3**

2 Installing Julia **5**

3 Installing Intan.jl **7**

4 Getting Started **9**
- 4.1 Overview . . . 9
- 4.2 Connecting Intan . . . 9
- 4.3 Creating GUI . . . 9
- 4.4 Interface Overview . . . 10
- 4.5 Data Acquisition . . . 10
- 4.6 Data Visualization . . . 11
- 4.7 Spike Sorting . . . 12

5 Data Acquisition **15**
- 5.1 Overview . . . 15
- 5.2 Workflow . . . 15
- 5.3 Initialization . . . 16
- 5.4 Calibration . . . 17
- 5.5 Data Collection . . . 17
- 5.6 Saving Neural Data . . . 17

6 Visualization **19**
- 6.1 Overview . . . 19
- 6.2 Interface . . . 20
- 6.3 Channel Gain . . . 22
- 6.4 Channel Threshold . . . 22

7 Spike Sorting **23**
- 7.1 Overview . . . 23
- 7.2 Automatic sorting . . . 23
- 7.3 Manual Sorting . . . 23

8 Experimental Control **25**
- 8.1 Overview . . . 25
- 8.2 Basic Data Structures and Methods . . . 25

8.3 Graphics . . . 26
8.4 Control Flow . . . 26

9 Online Decoding **27**
9.1 Overview . . . 27
9.2 Decoding Methods . . . 27

10 Tasks **29**
10.1 Overview . . . 29
10.2 Center Out . . . 29

11 Types and Methods **31**
11.1 Initialization . . . 31
11.2 Experimental Control . . . 31

12 Performance **33**
12.1 Overview . . . 33
12.2 Variables . . . 33
12.3 Performance . . . 34

Intan.jl is a graphical interface for viewing and processing extracellular electrophysiology data being collected in real time. This system works out of the box with Intan Technologies RHD2000-Series Amplifier Evaluation System, but could be configured to work with other systems that expose their acquisition API like Intan.

CHAPTER 1

Installation Overview

Intan.jl is built entirely using the Julia Programming Language. Additionally, Julia will call functions in several several C and Python libraries. In general, Julia's package manager is pretty good at taking care of these dependencies.

Right now, while Intan.jl is technically cross platform and built entirely from cross platform libraries, the GUI library (Gtk) is much more slugglish on Windows, and consequently would not currently be suitable for real-time processing. Hopefully soon these defects will be resolved.

CHAPTER 2

Installing Julia

Julia can be downloaded from here:

julialang.org/downloads/

Intan.jl is currently compatible with the latest stable version (v0.5.1).

CHAPTER 3

Installing Intan.jl

Launch a julia terminal. Julia packages are installed using functions in the package manager. For instance, to install Gtk, the command would be simply "Pkg.add("Gtk")"

The "add" command will work with all registered packages, and should take care of all package dependencies. On Linux, sometimes the underlying libraries must be installed separately using "sudo apt-get install package-name" in the terminal. These packages are called the following on Ubuntu/Debian:

libgtk-3-dev libhdf5-serial-dev

If you are using the RHD2000 acquisition board from Intan, you will also need to install the drivers necessary to recognize the board, located on the Intan website:

intantech.com/downloads.html

Finally, sometimes on ubuntu, there will be some problems with the udev library being used by Intan and available in the current release. If you experience problems, you may need to simlink the library it wants and the library you have with this command:

sudo ln -sf /lib/x86_64-gnu/libudev.so.1 /lib/x86_64-linux-gnu/libudev.so.0

CHAPTER 4

Getting Started

4.1 Overview

This brief guide describes how to start the GUI, begin collecting extracellular electrophysiological data, and use some of the online spike sorting tools. Additional more comprehensive documentation can be found in later sections.

4.2 Connecting Intan

4.3 Creating GUI

Start the Julia terminal.Then load the Intan package:

```
using Intan
```

The GUI can be started with the following line of code:

```
Intan_GUI()
```

This will create a GUI with the default configuration file. The configuration file specifies how many evaluation boards and headstages are connected, what behavioral task is to be run, how data is to be saved, etc. The default assumes that a 64 channel RHD2164 headstage is connected to the Port A1 of the Intan Evaluation board.

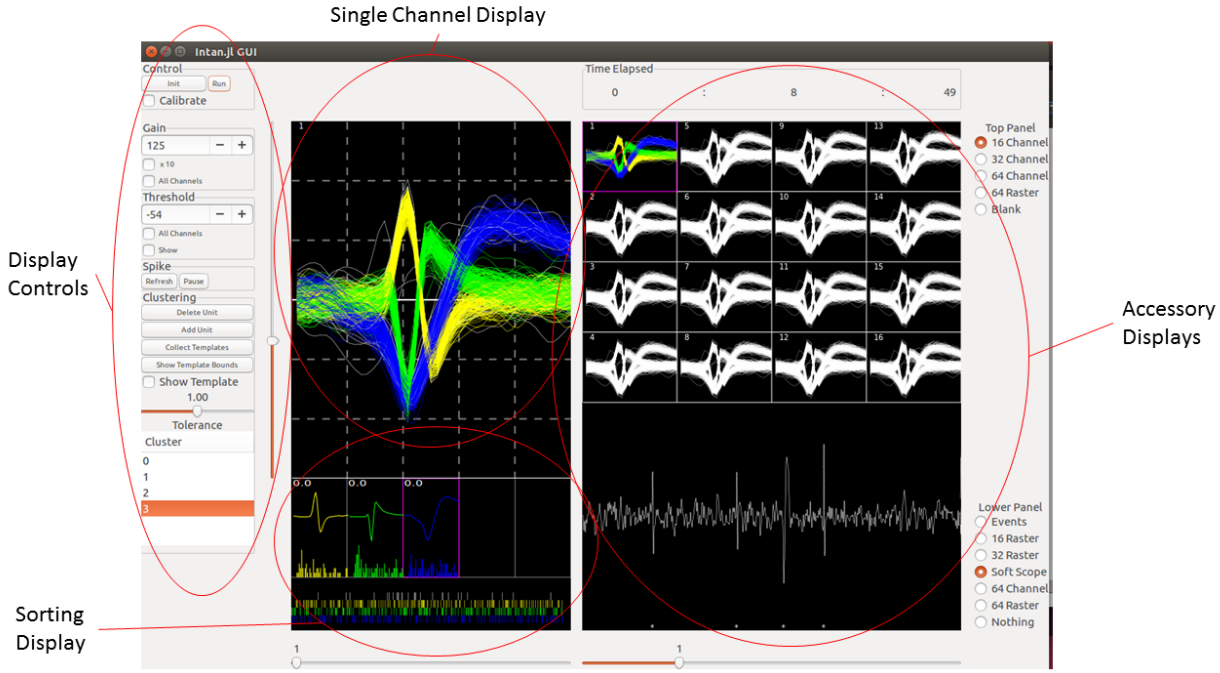
These files are simple and typically only a few lines of code. The user can create their own and then load it passing the path to the file as an argument to Intan_GUI:

```
Intan_GUI("/path/to/your/configuration/file.jl")
```

More documentation on designing the appropriate configuration can be found in its own section of this guide.

4.4 Interface Overview

Now the interface should appear. The GUI is divided into several regions by default:



The far left pane contains the most commonly used buttons for online spike sorting and data visualization. This pane also contains the buttons that start the data acquisition at the top.

The upper left pane shows a magnified view of a single channel. By default this shows the snippet of voltage surrounding a threshold crossing. The slider to set that threshold crossing is located on the left hand side. In this view, waveforms assigned to particular clusters (single units) will be assigned different colors.

The lower left pane shows spike sorting specific visualizations from the single channel highlighted above. These include the templates used for online template matching, and the color coded spike rasters for each individual unit.

The right pane is used for additional visualizations of incoming data. Some of these include 1) multiple channel displays, 2) multiple channel spike rasters, 3) digital oscilloscope and 4) digital events and analog signals from data given to the TTL and ADC inputs to the Intan evaluation board. One the right side of this pane are the buttons used to select which of these are displayed.

Finally, the top menubar has additional options, such as saving and loading sorting parameters, changing sorting options, and defining reference configurations.

4.5 Data Acquisition

The buttons used to start acquisition are located in the top right of the GUI. To begin data acquisition, first communication with the evaluation board and headstages must be established. This is done by clicking on the "Init" button. The output of the Julia terminal should indicate if the board is found and initialized correctly.

The "Run" button will begin acquisition with the evaluation board. After clicking the run button, the experimental timer should start climbing on the top right of the GUI. You will see that the "Calibrate button" is checked at this point, and data is not being displayed. At first, the data is being used by the software to start to find optimum threshold

and sorting values. This calibration period only needs to last a second or so. To start data visualization, uncheck the "calibrate" box.

4.6 Data Visualization

Now waveforms should start to appear in the single channel and multi channel displays. Only waveforms (snippets of the raw voltage signal from a channel) crossing a threshold value are displayed. The user can right-click in this window at any time to refresh the display.

4.6.1 Threshold and Gain

The user can manipulate this signal in two important ways: 1) changing the threshold value necessary for a waveform to be display, and 2) changing the gain of the incoming voltage signal. The current threshold and gain values are displayed to the left of the single channel display. The user can change the gain value (make the signal larger or smaller) by changing the value in the box to the left of the display. The threshold can be adjusted by dragging the threshold slider, located immediately to the left of the display.

The gain can be adjusted in multiples of 10 by selecting the x10 box. Additionally, the gain for every channel, not just the one displayed, can be modified by first selecting the "all channels" box under gain (remember to turn this off when you are done using it!)

The threshold can be displayed as a line across the single channel display by checking the "show" box in the threshold pain. Additionally, the threshold for all channels can be adjusted by selecting the "all channels" box under threshold (remember to turn this off when you are done using it!)

4.6.2 Selecting different channels

You can change the channel that is displayed in the single channel display by 1) left-clicking on the channel in the multi-channel display or 2) dragging the slider under the left pane to the desired channel value. Either way, the single channel that is displayed should change, as indicated by the number displayed in the top left of the single channel display, and by the purple box drawn around the selected channel in the mutli-channel display. Note that the slider under the left pane only allows you to select channels visble on the multi-channel display to the left. For instance, if there are 32 channels total, but you are only using the 16-channel display, you can only select channels 1-16. To see the second 16 channels (17-32), you can move the slider located under the multichannel display to change which channels are visualized.

4.6.3 Other displays

The buttons on the right side of the right display allow the user to change which additional visualizations are available. Depending on their size, multiple can be shown at once.

16, 32, and 64 channel displays

Waveforms from 16, 32, or 64 channels can be displayed simultaneously. These work the same as the 16 channel display in that the selected single channel is highlighted in purple, and you can select a single channel by clicking on it. If a channel is not needed to be visualized, the user can right-click on that channel and selected "disable".

16, 32, and 64 channel rasters

Raster plots are temporal event plots, where each event (a waveform detected on that channel) is displayed as a vertical bar at approximately the time relative to when the screen refreshed. Each raster displays about ~10 seconds worth spikes. Each spike will be color coded based on the unit it is assigned to during spike sorting.

Soft Scope

This is a digital oscilloscope that shows the raw voltage trace for the channel that is displayed in the single channel window. By right-clicking, the user can change the voltage-gain, time divisions, and whether or not the threshold will also be displayed. Asterisks are shown at the bottom of the soft scope to indicate signals that were identified as threshold crossings.

Events

This display can be used to visualize data that is received on either the analog-to-digital convertor on the Intan evaluation board, or as a TTL event on the evaluation board. Up to 6 different channels can be displayed at once. To display an event, right click on the section of the plot wher eyou would like that signal to be displayed and select the channel from the dialogue box.

4.7 Spike Sorting

The general technique used by Intan.jl for spike sorting is 1) selecting waveforms thought to belong to an individual neuron 2) building a template from these waveforms, and 3) performing template matching with this template, and 4) if it is a match, assigning that waveform as belonging to that individual neuron. The user can create as many separate templates as necessary for a single channel.

The "cluster" dialogue box at the bottom left indicates which cluster, or unit, is currently selected. By default, a channel will only have the 0 cluster, which indicates that there are no units and consequently no template matching that will be performed. As you add units, this list will be populated with more units (1, 2, 3 etc). You can click on each unit to make it the "active unit."

4.7.1 Selecting Waveforms

At any time, you can left-click in the single channel display, hold, and drag. You will see a rubberband created while doing this. Any units that cross this rubber band when you let go of the left-click will be assigned to the currently active cluster, and a template will be generated from those waveforms. Right now, selecting units with the 0 cluster selected should do nothing.

4.7.2 Collecting Templates

By default, the single display will be continuously updated as new waveforms stream in, and will automatically refresh about every 10 seconds. This can make unit selection difficult; therefore, you can select the "Collect Templates" button to refresh the display followed by more waveforms continuing to be plotted in real time. The "collect templates" button will now read "stop collection". Once "stop collection" is selected, the single display will freeze. While data will continue to stream in, the user can manipulate these units on the display.

In the "collect units" mode, the right-click now will also generate a rubberband if the user clicks, holds and drags. Any units that cross this rubber band when it is released will be "masked" and disappear from the display. This can be

used to clear obvious noise from the display before moving onto template generation. This mask feature works when the "0" cluster is selected.

4.7.3 Adding Units

Selecting Waveforms

To create a new unit, first click the "add unit" button. In the cluster list, you should see a new unit appear with the number 1, and it will be automatically selected. Now any units that are selected with the left click rubber band will be assigned to that unit. Left-click, drag and release to select a group of units, and you will see the waveforms turn the same color as the new cluster (yellow for cluster 1). Of note, the user should also see each unit turn that color in real time, so that the user has real time feedback about which units will be selected.

Deselecting Waveforms

The user can also use the right-click rubberband to remove units from the display. If unit 0 is selected, these units will be removed from the display. However, if the current unit is selected waveforms not belonging to that unit will be removed from the display, but waveforms that were assigned to that unit will be removed from the display and the template will be recalculated with those units subtracted. In this way, the user can always be generous in their first selection and "prune" away unwanted waveforms to create the best template.

4.7.4 Resuming Display

4.7.5 Deleting Units

4.7.6 Modifying Template Bounds

CHAPTER 5

Data Acquisition

5.1 Overview

This software surrounds hardware made with Intan RHD amplifier and digitization chips for electrophysiology:

http://intantech.com/

Which communicate with Opal Kelly FPGAs

http://intantech.com/products_RHD2000_Rhythm.html

Opal Kelly provides the Rhythm interface as open source C++ and verilog code. This project seeks to be a Julia alternative to allow for easier integration of spike sorting and closed loop control in the Julia language.

5.2 Workflow

The main control loop has the following steps

1. Read neural data from the Intan board.

The Intan board will continuous run throughout the experiment, capture and organize the digital neural signals coming from headstage, and keep them in RAM until the control computer brings them over. How frequently data is brought over from the FPGA board is set by the user, and the "best" frequency depends on the experimental setup (remember that this will not affect the sampling rate of the headstages). By default, Intan.jl pulls from the FPGA board once 600 samples from each channel have accumulated. At a 20kHz sample frequency, this will correspond to a latency of roughly 30 ms.

The control loop must run faster than this minimum latency. Therefore, at the beginning of each loop iteration, the board checks to see if it has accumulated 600 samples. If not, it waits very briefly and then checks again. If it has, it pulls the data over to the PC. This will take a small amount of time due to overhead of the method of transfer, as well as the speed of transfer itself. Once the data is brought over, it must be parsed into a more sensible format, which is a 600 x channel matrix of 16 bit integers.

1. Spike Sorting

The new matrix of voltage data is then put through the spike sorting paradigm of the users choice. Any spikes that are detected and clustered will be stored in RAM for the following steps.

1. Task Logic

Intan.jl will now implement the user's task logic, which corresponds to the "do_task" function for the Task data structure. This may be nothing, such as with "Task_NoTask," or it may involve updating task GUIs (e.g. a center out cursor position), performing some calculation with the newly found spikes (e.g. brain machine interface decoding), controling external task-relevant equpiment (e.g. spike-triggered stimulation), etcetera.

1. Update Intan.jl GUI

Now the main GUI will be updated, with any newly identified spikes plotted.

1. Saving

Finally, the data from this loop iteration will be saved. This includes the neural data and whatever task data is specified in the "save_task" function (this could be nothing). Data is saved as a binary file for speed, which will then be parsed into a more useful HDF5 format immediately after the experiment is complete.

5.3 Initialization

First, the user should specify how the headstage is connected to the Intan Evaluation board. The evaluation board has 4 SPI ports, which each can recieve two headstages via a y connector. These headstages are therefore labeled A1,A2,B1,B2,C1,C2,D1,D2. The user must create an Amp data structure for each amplifying by specifying the port it is connected to and calling the function corresponding to the amplifying type. These are then provided to the FPGA function as an array. For example:

```
#96 channel setup with 64 channel amp and 32 channel amp connected to Port A with a y␣
↪connector

#64 channel amp connected to the A1 port
myamp1=RHD2164("PortA1")

#32 channel amp connected to the A2 port
myamp2=RHD2132("PortA1")

myfpga=FPGA([myamp1,myamp2])
```

```
#128 channel setup with 64 channel amps connected to ports A and B

#64 channel amp connected to the A1 port
myamp1=RHD2164("PortA1")

#64 channel amp connected to the B2 port
myamp2=RHD2164("PortB1")

myfpga=FPGA([myamp1,myamp2])
```

Now add the FPGA to the primary RHD2000 data structure, which will include options for spike sorting, saving, task logic etc.

```
myrhd=RHD2000([myfpga],mytask,sav=mysave)

#Creates the GUI
handles = makegui(myrhd)
```

To connect to the board and perform initialization, click the Init button in the top left hand corner.

5.4 Calibration

The board will need to run for a few seconds before further processing to adequately calculate the thresholds for spike detection on each channel. So when the user clicks "Run", no signals will be displayed at first. This is becuase the "calibration" check box is checked. Once several seconds have passed, the user should uncheck the calibration box and neural signals will start to appear.

5.5 Data Collection

After calibration has finished, the full control loop will run until the "Run" button is unclicked.

5.6 Saving Neural Data

During the experiment, saving the neural data in a binary format is significantly faster than alternative methods. Different parts of the neural data can be saved: the entire voltage trace from each channel, the voltage waveforms, or nothing at all. These options are specified by creating a save type and passing this to the RHD2000 function with the sav keyword like so:

```
#Save all waveforms
mysave=SaveAll()

myrhd=RHD2000([myfpga],mytask,sav=mysave)


#Save just the waveforms
myrhd2=RHD2000([myfpga],mytask,sav=SaveWave())

#Don't save anything
myrhd3=RHD2000([myfpga],mytask,sav=SaveNone())
```

Data is saved in a folder named with the date and time in the working director. Voltage traces will be saved as a file named "v.bin" in the working directory. If all of the analog traces need to be worked with directly, they can be loaded into the workspace with the parse_v function by specifying the channel number:

```
#Assumes v.bin is in working directly and the number of samples per data block is the␣
↪same as when data was collected.
parse_v()
```

Time stamps for each detected spike is also saved in the working directory in binary form as "ts.bin" Binary is not immediately useful for any future analysis, so a parser can be run immediately after recording to save the timestamps in a HDF5 format file such as .mat (MATLAB) or .jld (Julia). This can be done as follows:

```
#Saves timestamps as "spikes.mat" in working directory
save_ts_mat()

#Saves timestamps as "spikes.jld" in working directory
save_ts_jld()
```

Alternatively, data can be exported into MAT, JLD or Plexon file types after the experiment is completed using the "Export" menu in the GUI.

CHAPTER 6

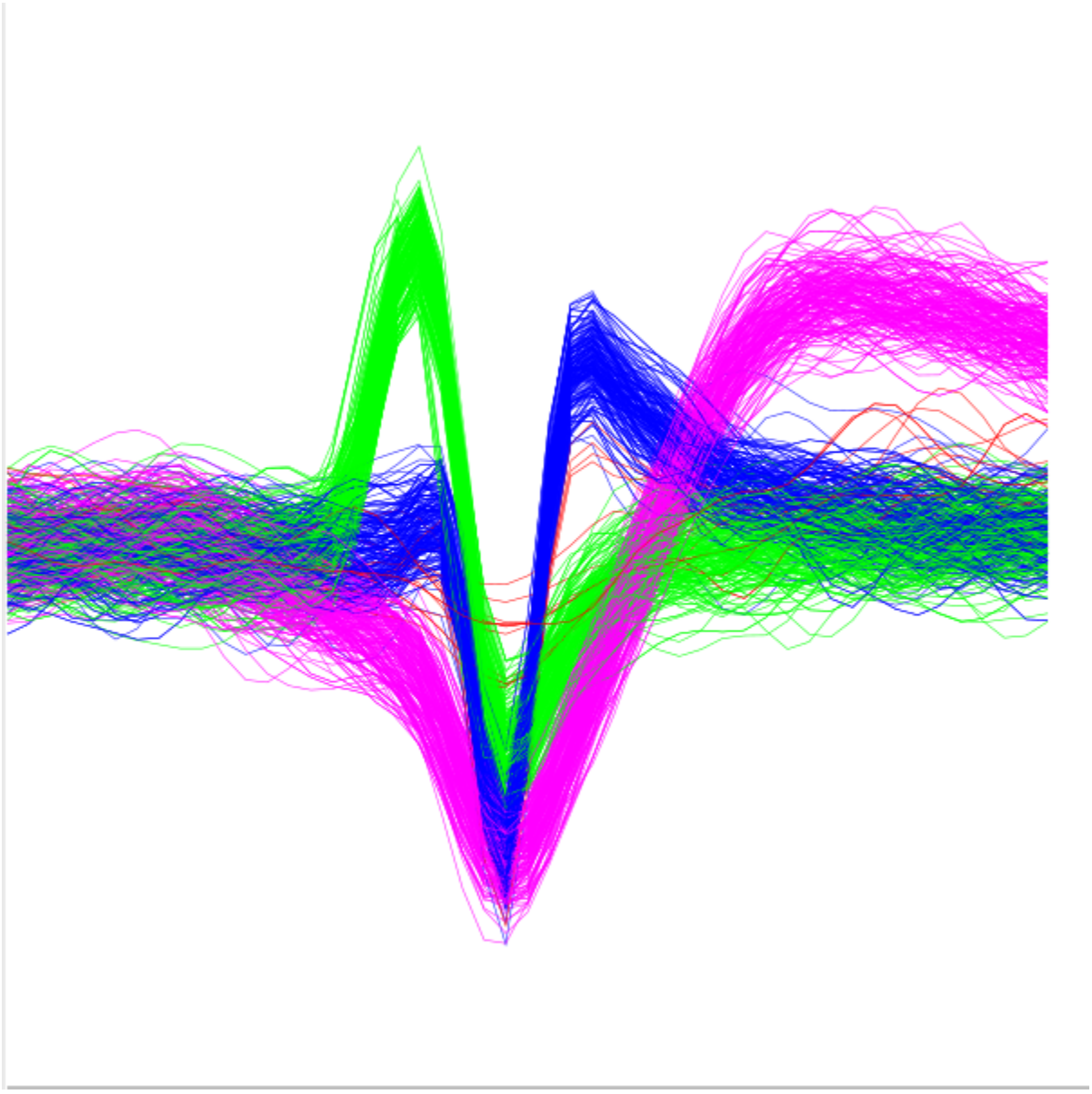
Visualization

6.1 Overview

The current Opal Kelly and Intan recording hardware allow for a few or a few thousand channels of electrophysiological signal to be recorded from simultaneously. The user must be able to quickly assess the quality of the incoming signals and the performance of the automatic spike sorting routines. The main visualization displays of Intan.jl allows the user to view a signal channel at a time and multiple channels at once.

6.2 Interface

6.2.1 Single Channel Display



The left display shows voltage-time plots of detected spike events. The scroll bar at the bottom can be used to select the individual channel from the group display (to the right) to display.

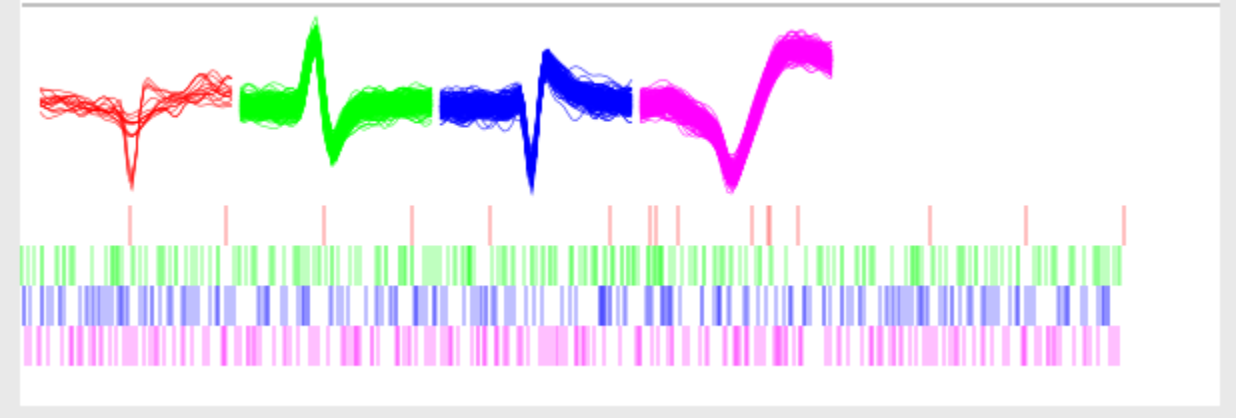
Different clusters are represented with different colors.

6.2.2 Multi-channel Display

The right display shows voltage-time plots of 16 channels simultaneously. The scroll bar at the bottom can be used to select which group of sixteen channels to display. This will reset the left display to the first channel in this group.

Different clusters are represented with different colors.

6.2.3 Waveform Display

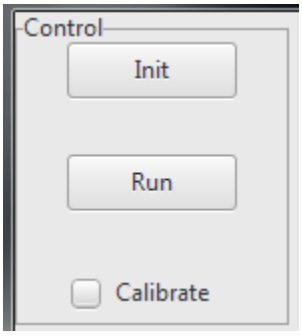


For better visualization, each waveform on the main display channel will be plotted separately, along with the corresponding spike raster.

6.2.4 Event and Analog Display

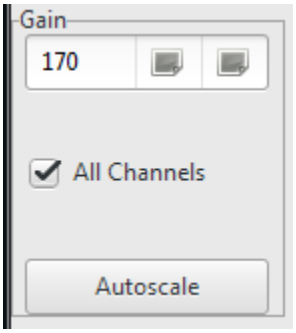
Up to six channels of analog and ttl signals can be displayed at one time. The signal of interest can be selected from the combo boxes to the right. By default, no channels are selected. TTL channels will display as either high or low, while analog signals will show continuous changes.

6.2.5 Control



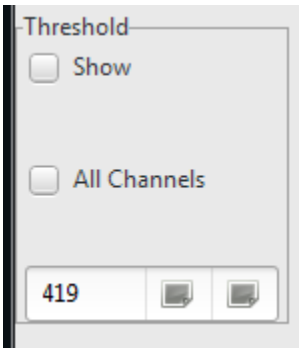
Before starting, click the Init button to connect to the FPGA and calibrate the headstages. The run button will start collecting data. The first data is initially collected in "calibration mode." All the board to run for several seconds (or longer depending on what spike sorting method is selected) before unchecking the calibration box. After this box is unchacked, data will begin to be displayed on the displays.

6.3 Channel Gain



Gain can be adjusted for a single channel or, with the All Channels checkbox selected, for every channel on the headstage. The autoscale button will try to calculate the optimal gain value for the channel which is displayed.

6.4 Channel Threshold



The threshold can be adjusted for a single channel, or with the All Channels checkbox selected, for every channel on the headstage. The show checkbox is used to display the threshold for the single maximized channel.

CHAPTER 7

Spike Sorting

7.1 Overview

Determing which pieces of a voltage vs time signal recorded from an extracellular electrode is attributable to an action potential fired by a neurby neuron is known as spike sorting. Spike sorting can be thought to consistent of several steps including 1) detecting candidate spikes, 2) aligning candidate spikes so they exist in the same relative time window, 3) transforming voltage vs time series of data points into some meaningful feature space, 4) reducing the high dimensional time series into a set of dimensions that can best discriminate electrical activity from one nearby neuron from another, and 5) clustering spikes with similar features to attribute them as arising from the same neuron.

This package makes available all of the spike sorting routines in the julia spike sorting package:

https://github.com/paulmthompson/SpikeSorting.jl

As well as additional manual manipulation of clusters that are automatically determined.

Note that the entire analog trace from each channel can also be saved for later offline sorting.

7.2 Automatic sorting

7.3 Manual Sorting

Spike clusters can be selected manually. For each channel, a cluster can be specified with a series of lines. A spike that crosses those lines will be assigned to that cluster. A new line for the selected cluster can be specified by left clicking, moving and releases. Right click allows the user to cycle through the lines of each cluster. Middle click clicks through clusters.

CHAPTER 8

Experimental Control

8.1 Overview

This package can be used as part of an experimental workflow to record inputs and outputs, trigger events such as the start of a trial, or draw and manipulate what is seen on a computer screen. Every unique version of an experiment can be defined in a task file, which will create the data structures and methods necessary. Here we will outline some of the supported features for task creation.

8.2 Basic Data Structures and Methods

8.2.1 Task Data Structure

Every task will need its own data structure to store whatever variables are necessary for processing during the task. This should also store whatever additional variables besides neural signals that need to be logged.

```
type Task_NoTask <: Task
end
```

8.2.2 Initialization

An "init_task" initialization function will build all necessary elements before anything starts running (external boards, creating GUIs etc).

```
function init_task(myt::Task_NoTask,rhd::RHD2000)
end
```

8.2.3 Experimental Control

The "do_task" function will implement the control logic of the task such as updating GUIs, modifying the data structure, talking to external boards. It is called immediately after spike sorting and before logging.

```
function do_task(myt::Task_NoTask,rhd::RHD2000,myread)
end
```

The "myread" variable is a boolean that indicates if data was read from the Intan board or not. If you only want your experimental control to run every time new data is acquired (for instance every 20 ms when sampling at 30khz), then place your methods inside a conditional myread==true block of code.

8.2.4 Logging Function

The "save_task" function will save the appropriate elements of the data structure, as well as specifying what analog streams from either the Intan or other external DAQs.

```
function save_task(myt::Task_NoTask,rhd::RHD2000)
end
```

8.3 Graphics

8.3.1 Cairo

8.3.2 OpenGL

8.4 Control Flow

CHAPTER 9

Online Decoding

9.1 Overview

Some applications of neurophysiological recordings require real-time processing of neural data to affect some component of a task (e.g. brain computer interfaces). Here we outline some popular methods used in real-time decoding and how they are implemented in this package.

9.2 Decoding Methods

9.2.1 Weiner Filter

9.2.2 Kalman Filter

9.2.3 Unscented Kalman Filter

9.2.4 Population Vector

CHAPTER 10

Tasks

10.1 Overview

Intan.jl could be used to implement a wide variety of experimental protocols. Here we describe the ones that are made available with the package.

10.2 Center Out

CHAPTER 11

Types and Methods

11.1 Initialization

RHD2164 (*portstring*)
Returns data needed by evaluation board needed to recognize 64 channel amp connected to port specified in portstring

RHD2132 (*portstring*)
Returns data needed by evaluation board needed to recognize 32 channel amp connected to port specified in portstring

RHD2000 (*myamps*, *processor*, *mytask*)

makegui (*rhd*)

SaveAll ()

SaveWave ()

SaveNone ()

11.2 Experimental Control

init_task (*mytask*, *rhd*)

do_task (*mytask*, *rhd*)

save_task (*mytask*, *rhd*)

CHAPTER 12

Performance

12.1 Overview

Intan.jl can be used in a variety of experimental setups, such as recording from many electrodes simultaneously, or performing closed loop experiments. A "high performance" setup may be different depending on the experimental paradigm. We will demonstrate some metrics of performance below, and highlight how Intan.jl can be used to achieve different experimental goals.

12.2 Variables

The technical demands necessitated by this data acquisition method is based on two primary factors: how much data we are acquiring per unit time, and how frequenty we need to communicate with the acquisition device (as there is some overhead in each communication). We have control over some

12.2.1 Sample Rate

Our sample rate defines how many samples we are going to collect from each electrode per second. A higher sampling rate will necessarily mean that more data must be transferred from the acquisition board to the computer. However, the sampling rate also sets our temporal resolution for characterizing waveforms: too low of a sample rate will not allow us to sufficiently characterize and separate extracellular potentials from different neurons.

12.2.2 Latency

Latency defines how frequently we communicate with the acquisition board to transfer new signals. There is some overhead in this process (the acquisition board must put the data into a form that can be transferred). This process also sets number of times a closed loop experiment can be implemented per second. For instance, a latency of 10 ms allows for a 100 iterations of closed loop control per second. This also means that whatever additional processing of the neural signals (visualization, decoding, saving etc) should all take place in the time window of 10ms minus whatever time is needed to transfer the data from the acquisition board to the experimental computer.

12.2.3 Channel Count

More channels means more data will need to be transferred per unit time.

12.2.4 Experimental Control Computer

In general, don't cheap out for performance critical equipment.

12.2.5 Data Transfer Protocol

The default method for how data gets from the data acquisition device to the experimental computer uses usb2.0. The transfer rate of usb2.0 is slow when compared to the newer usb3.0 or something like ethernet. The FPGA board used by the default Intan evaluation board (which uses usb2.0) also comes in a usb3.0 and PCI version. These protocols would allow for faster data transfer, but will require some low level tinkering to implement.

12.3 Performance

A general experimental workflow will be this: the acquisition board gathers data from the headstages; the acquisition board transfers data to the experimental control computer; the experimental control computer implements spike sorting, task logic, data visualization and data saving.

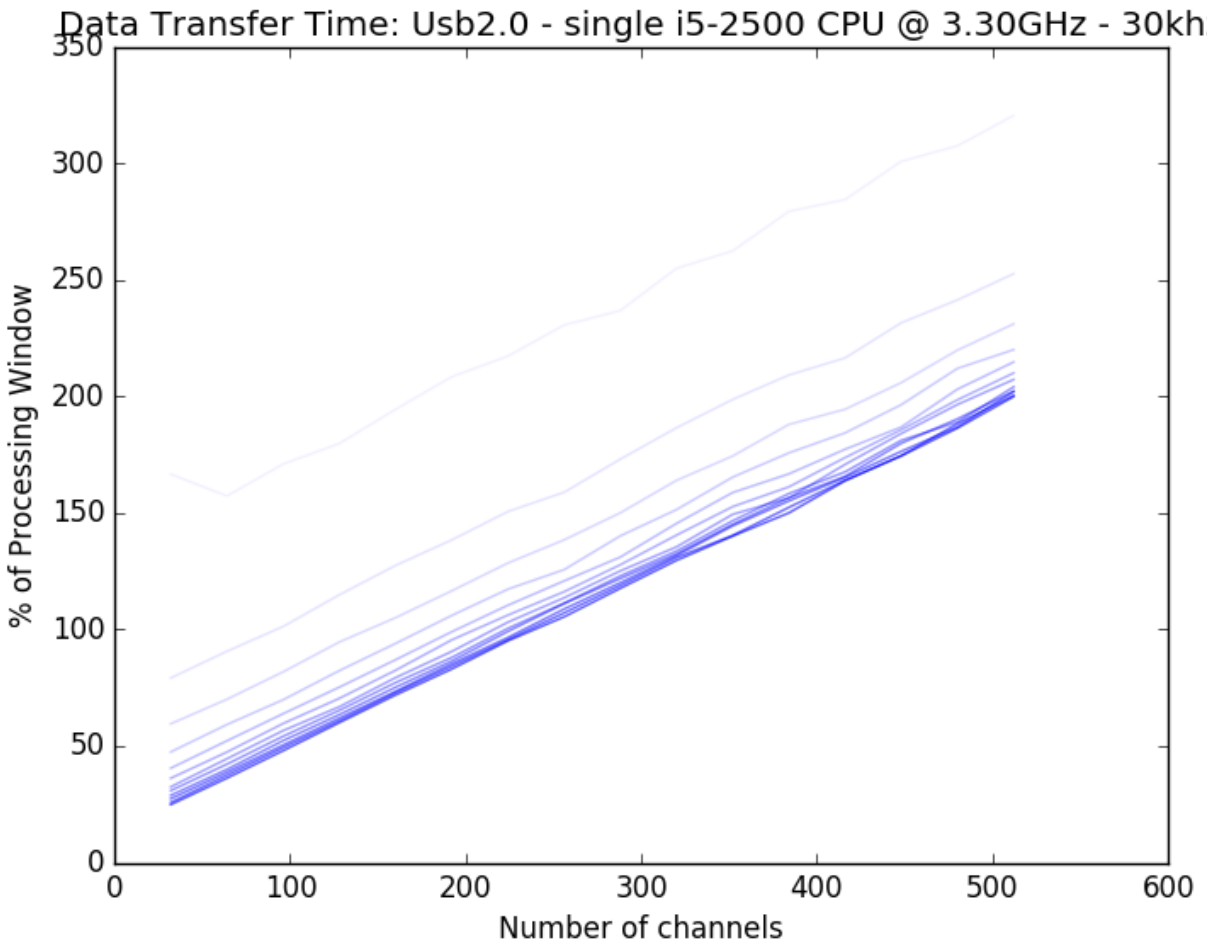
Of the above steps, transferring data from the acquisition board to the experimental computer is the most resource intensive. Additionally, the methods of improving the speed of that step are fundamentally different and require much more "low level" tinkering. For that reason, we will report the "Data Transfer" time separately from the "Data Processing" time.

Rather than report units of processing "time", we are going to normalize values to the sampling latency. For instance, if we are bringing data over from the data acquisition board every 30 ms, and a processing step takes 15 ms, we will report this value as using "50% of processing window". Therefore, values above 100% are strictly impossible to implement, as the entire process (data transfer + data processing) must be less than 100%.

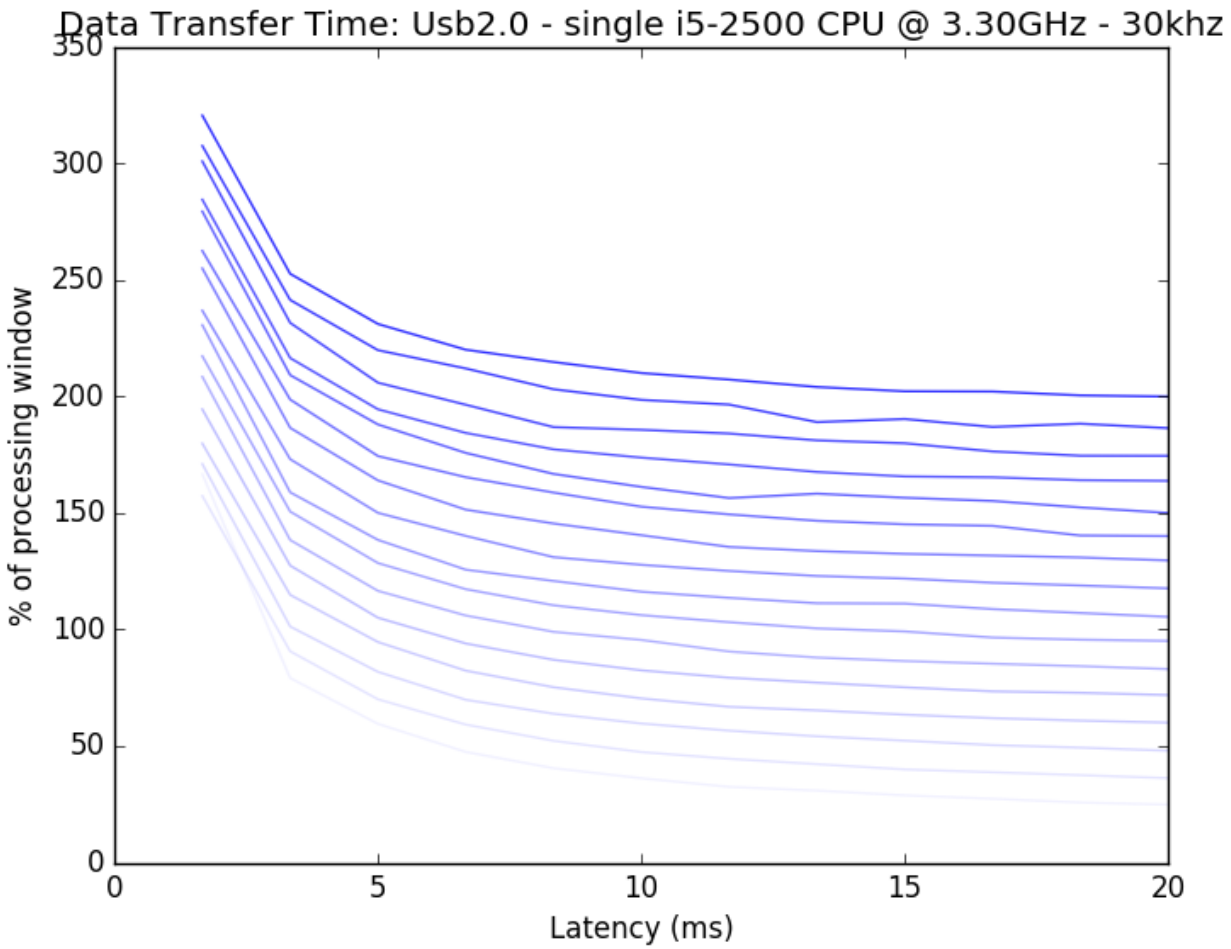
12.3.1 Overall Performance

12.3.2 Data Transfer

First, let us characterize the time to transfer data from an unmodified evaluation board that can be purchased from Intan Technologies. This uses a usb2.0 communication protocol. We can first compare how much time it takes to process data from an increasing number of channels:



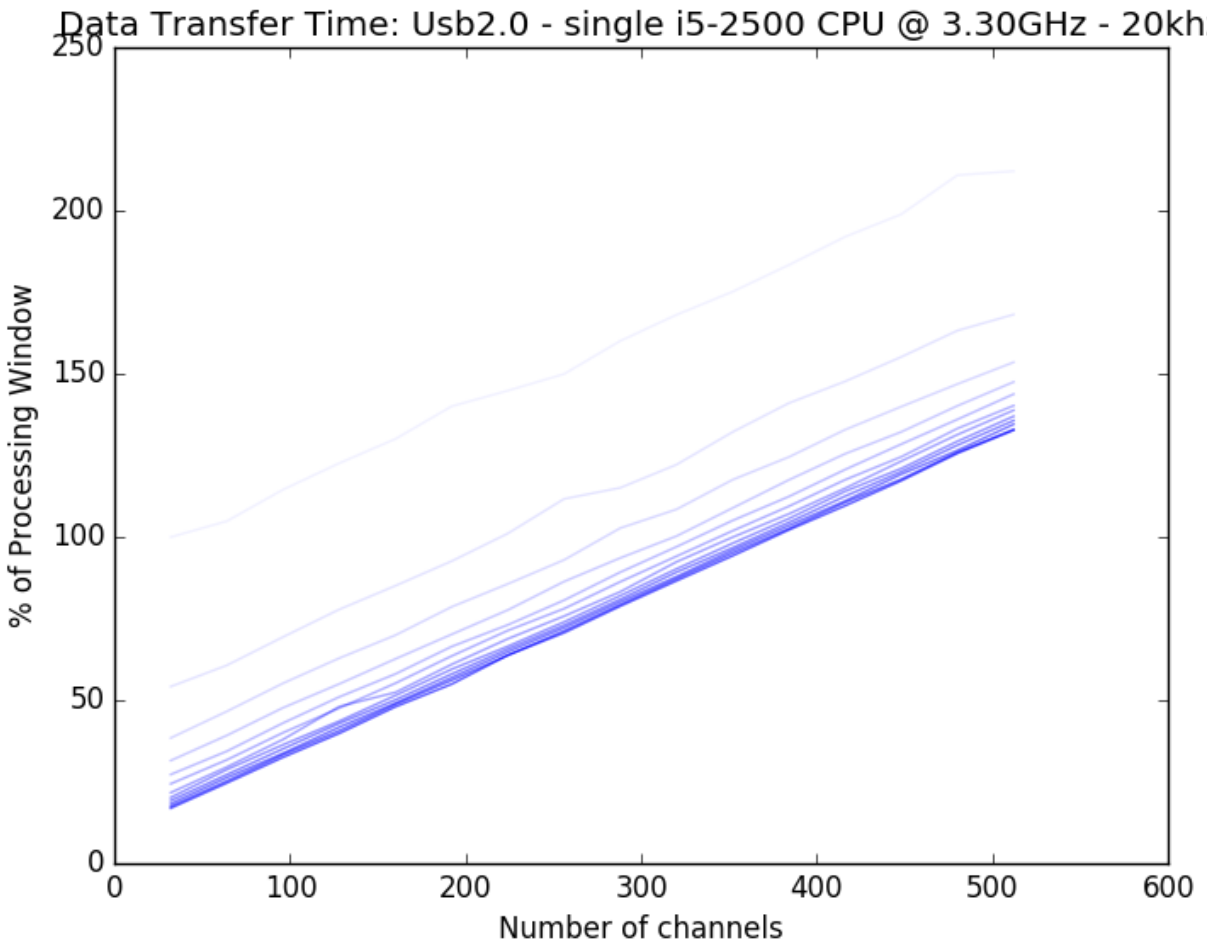
Here we have plotted an increasing latency (sampling less frequency from the acquisition board, with values ranging from 1.7 ms to 20 ms with increasing opacity). First, we note that the data transfer time is roughly linear with increasing channel count. Second, we see that there is a nonlinear relationship with latency, with very small latencies such as 1.7ms being impossible even for small channel counts. This relationship is better visualized here, with transfer time as a function of latency, with increasing channel count lines increasing in opacity:

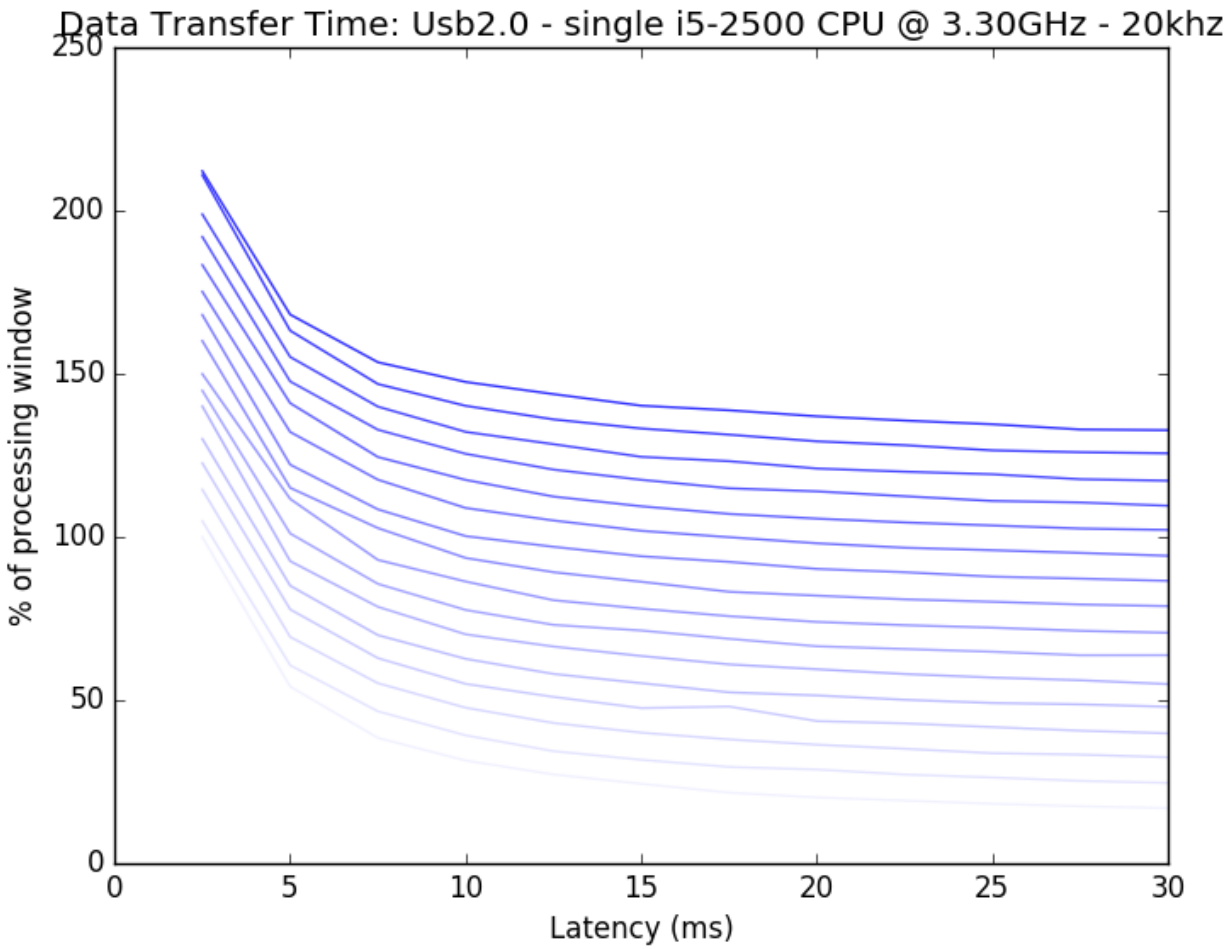


This better displays that very low latencies (<5ms) are asymptotically inefficient, while latency becomes a smaller factor after this value.

We can also see that most points on these plots fall above the 100% impossibility line. For high channel counts, we are simply transferring too much data for a usb2.0 protocl to realistically handle. Partly for this reason, the off the shelf Intan evaluation board is only made to handle 256 channels simultaneously.

We can add the possibility of more channels in a few ways. First we could decrease the amount of data that needs to be transferred. This can be easily done by decreasing the sampling rate for each channel. If we cut our sampling rate from 30khz as above to 20khz (below which would make most spike sorting methods fairly difficult) we can see improvement:


Data Transfer Time: Usb2.0 - single i5-2500 CPU @ 3.30GHz - 20khz
% of Processing Window
Number of channels
0
50
100
150
200
250
0
100
200
300
400
500
600



12.3.3 Filtering

12.3.4 Spike Sorting

12.3.5 Plotting

12.3.6 Saving

Index

D

do_task() (built-in function), 31

I

init_task() (built-in function), 31

M

makegui() (built-in function), 31

R

RHD2000() (built-in function), 31
RHD2132() (built-in function), 31
RHD2164() (built-in function), 31

S

save_task() (built-in function), 31
SaveAll() (built-in function), 31
SaveNone() (built-in function), 31
SaveWave() (built-in function), 31