Online Control and Data Analysis

Use Matlab or Python during recording

Updated 2025-02-27

Tucker-Davis Technologies
11930 Research Circle
Alachua, FL 32615 USA
Phone: +1.386.462.9622
Fax: +1.386.462.5365

Notices

The information contained in this document is provided "as is," and is subject to being changed, without notice. TDT shall not be liable for errors or damages in connection with the furnishing, use, or performance of this document or of any information contained herein.

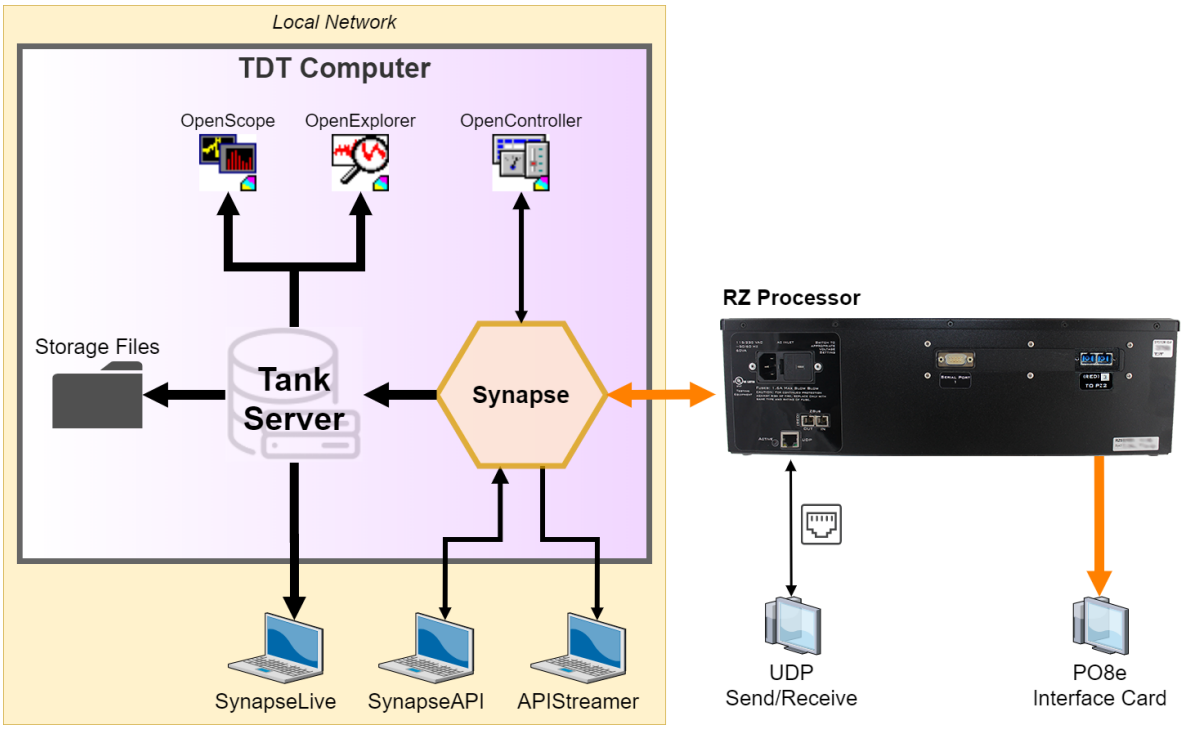
The latest versions of TDT documents are always online at https://www.tdt.com/docs/

Table of Contents

Online Control and Data Analysis

TDT Online Control and Data Analysis

TDT provides several API tools for reading data as it is recorded and controlling Synapse and TDT hardware programmatically. This page provides an overview and comparison of these tools.



Synapse runs on Windows, but many of these tools can run across network or fiber optic connections to other computers with different operating systems.

Library	Direction	Connection	Bandwidth	Speed	MATLAB	Python	Cross-Platform
SynapseAPI	Read-Write	PC/Network	++	+++			
RZUDP	Read-Write	Network	+	+++++			
SynapseLive	Read-Only	N/A	++++	+			
APIStreamer	Read-Only	PC/Network	++	+++			
PO8eStreaming	Read-Only	Fiber Optic	+++++	+++++			

Library	Use Case
SynapseAPI	Interface with Synapse while it is running. Mimic anything a user can do, and read/write parameters on hardware.
RZUDP	Event-driven packets, typically spike counts, sent across network interface
SynapseLive	Read all saved data directly into MATLAB through the tank server as it is recorded (6 second delay typical). Can be combined with SynapseAPI for closed-loop control.
APIStreamer	Read streaming data directly from the hardware while Synapse is recording. Can be combined with SynapseAPI for closed-loop control.
PO8eStreaming	Fast access to large amounts of data. Often used with RZUDP or SynapseAPI for closed-loop control.

SynapseLive

The MATLAB SDK includes the SynapseLive class for reading data from the tank as it is acquired, with a several second delay. This is a Windows-only solution.

You can download the latest MATLAB SDK files here. The SynapseLive class installs into:

```
C:\TDT\TDTMatlabSDK\TDTSDK\SynapseLive
```

Example experiments and scripts can be found here:

```
C:\TDT\TDTMatlabSDK\Examples\OnlineAnalysis
```

All block data is available in MATLAB during the recording. It is in the same data structure as if you read the block offline using TDTbin2mat.

The amount of delay between when the data occurs and when it is available in MATLAB is controlled by the Tank Engine Cache Delay parameter. This is the amount of time Synapse has to save the data to disk before it's made accessible to other applications, like SynapseLive. You can change this setting in Synapse Menu > Preferences > Advanced.

The lower the delay, the sooner you can read the data into MATLAB. However, if the data rate is high or variable (for example there is a brief flood of spike activity caused by a sudden noise burst) then there is a chance that Synapse and the tank server could get too far behind. Synapse will give you data errors and stop recording if this is the case.

Consider also increasing the Data Read Limit in Synapse Menu > Preferences > Run-time Ops. This prioritizes data storage over visualization in Synapse.

Examples

Online Signal Averaging

Import strobe store gizmo data into MATLAB using SynapseLive during the experiment

Plot the average waveform

Good for Evoked Potential visualization

Online Raster PSTH

Import snippet and epoc data into MATLAB using SynapseLive while the experiment is running

Generate peri-event raster and histogram plots over all trials

Good for stim-response experiments, such as optogenetic or electrical stimulation, where you need immediate visual feedback

Online Streaming Data

Read a streaming data store into MATLAB from Tank server during a recording

APIStreamer

APIStreamer is a class that reads directly from buffers on the hardware without waiting for it to go to the tank. Because it is essentially competing with Synapse for access to the hardware, it has lower bandwidth than SynapseLive, but it has much faster access.

You can download the latest MATLAB SDK files here. The MATLAB APIStreamer class installs into:

```
C:\TDT\TDTMatlabSDK\TDTSDK\APIStreamer
```

Examples install into:

```
C:\TDT\TDTMatlabSDK\Examples\OnlineAnalysis
```

The Python APIStreamer installs with pypi:

```
pip install tdt
```

Release v96 of Synapse and above includes the custom gizmos that APIStreamer can read from, in the Gizmo List under Custom > TDT > API.

1. Attach one of the APIStreamer gizmos (e.g. APIStreamMC for multi-channel signals) to your data source.
2. Use one of the examples to read from the buffer inside the APIStream gizmo during the recording.

The data rate limit for APIStreamer is ~8 channels of 32-bit floats at ~25 kHz or the equivalent data rate (e.g. 32 channels at ~6 kHz). Beyond that we recommend a PO8e Streaming Interface card, which is a direct connection from the RZ to a separate PCI card, bypassing Synapse altogether.

Matlab Examples

Online Streaming Data

Read a stream of multi-channel data directly into Matlab using APIStreamer

Online Spectrum Analyzer

Read a single channel and plot the power spectrum over time

Good for visualization and online analysis

Python Examples

Input Parameter	Description
host	IP Address of Synapse computer (default 'localhost')
gizmo	Gizmo name in Synapse (default 'APIStreamerMC1')
history_seconds	How much data to keep in Python memory (default 1 second)
callback	Function handle to run every time new data is collected
verbose	Print debugging information to the console (default True)

Online Data Processing

APIStreamer is collecting data in the background. Reads the data buffer every 1 second.

```
import time
import tdt

s = tdt.APIStreamer(gizmo='APIStreamerMC1', history_seconds=10)

while 1:
    print(s.get_data())
    time.sleep(1)
```

Online Data Processing Callback

Runs the `do_something` function every time new data is read from the hardware

```
import tdt

def do_something(result):
    print('process data', result.data.shape)

s = tdt.APIStreamer(gizmo='APIStreamerMC1', history_seconds=10,
callback=do_something)
```

Online Streaming Data

Plots the data from the APIStreamer

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.animation as animation

import tdt

s = tdt.APIStreamer(gizmo='APIStreamerMC1', history_seconds=10)

fig = plt.figure(figsize=(10,5))
ax1 = fig.add_subplot(111)

def animate(i):

    result = s.get_data()
    if result is None:
        return

    xs, ys = result
    if len(xs) == len(ys) == 1:
        return

    ax1.clear()
    ax1.plot(xs, ys.T + np.arange(ys.shape[0]).T)
    ax1.set_xlabel('time, s')
    ax1.set_ylabel('V')

ani = animation.FuncAnimation(fig, animate, interval=100,
cache_frame_data=False)

plt.show()
```

Plotting Multiple Streams

Reads from two APIStreamer gizmos and plots them

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.animation as animation

import tdt

s = tdt.APIStreamer(gizmo='APIStreamer1Ch1', history_seconds=10)
s2 = tdt.APIStreamer(gizmo='APIStreamer2Ch1', history_seconds=10)

fig = plt.figure(figsize=(10,5))
ax1 = fig.add_subplot(211)
ax2 = fig.add_subplot(212)

def animate(i):

    result = s.get_data()
    if result is None:
        return

    result2 = s2.get_data()
    if result2 is None:
        return

    xs, ys = result
    ax1.clear()
    ax1.plot(xs, ys.T + 1*np.arange(ys.shape[0]).T)
    ax1.set_xlabel('time, s')
    ax1.set_ylabel('V')

    xs2, ys2 = result2
    ax2.clear()
    ax2.plot(xs2, ys2.T + 1*np.arange(ys2.shape[0]).T)
    ax2.set_xlabel('time, s')
    ax2.set_ylabel('V')

ani = animation.FuncAnimation(fig, animate, interval=100,
cache_frame_data=False)
plt.show()
```

PO8eStreaming

PO8eStreaming is a library of methods for accessing data on one or several PO8e interfaces through a custom Windows or Linux application. See PO8e Streaming Interface for more hardware-related information.

The PO8eStreaming libraries and examples for C++, Python, and MATLAB install with the TDT drivers into:

```
C:\TDT\RPvdsEx\Examples\PO8e\
```

Users should be mindful of using good 'closed loop' access when working with PO8eStreaming. This means always releasing any open connections to PO8e cards.

A typical PO8e access session for a client consists of five main steps:

1. Call `connectToCard` to get a pointer to an available PO8e card.
2. Call `startCollecting` to begin reading from PO8e card.
3. Run the circuit on the RZ device that streams to the PO8e card.
4. Perform any number of buffer operations.
5. Call `releaseCard` to release the card object from memory.

Setup and Control

The methods in this group are used to setup access to any PO8e card(s) in the system.

cardCount

Returns the number of PO8e cards detected in the system. Call this first to determine the possible values for the `cardIndex` passed to the constructor.

C++ prototype

```
static int cardCount();
```

C prototype

```
int cardCount();
```

Returns	Type	Values
totalCards	int	0-4, total number of cards in the system

Example

C++

```
int totalCards = PO8e::cardCount();
```

C

```
int totalCards = cardCount();
```

connectToCard

Returns a pointer to the specified card index. Note that the index will be consistent across system boots and is dependent on the PCIe bus layout, so if you move the cards between slots their respective indices can change.

C++ prototype

```
static PO8e* connectToCard(unsigned int cardIndex = 0);
```

C prototype

```
void* connectToCard(unsigned int cardIndex = 0);
```

Inputs	Type	Values
cardIndex	unsigned int	0-3, target card index
Returns		
pointer to PO8e instance		

Example

This code sample creates a PO8e object pointing to the first card identified in the system.

C++

```
PO8e *card = PO8e::connectToCard(0);
```

C

```
void *card = connectToCard(0);
```

releaseCard

Free the PO8e card objects through this interface. It is done this way to ensure that in Windows the objects are freed from the correct heap context.

C++ prototype

```
static void releaseCard(PO8e *card);
```

C prototype

```
void releaseCard(void* card);
```

Inputs	Type
card	pointer to PO8e object

Example

This code sample releases the card object memory.

C++

```
PO8e::releaseCard(card);
```

C

```
releaseCard(card);
```

Hardware Data Access

The methods in this group are used to read data from PO8e card(s).

startCollecting

Call this to start collecting a data stream from the PO8e card. Collected data will be buffered as needed.

C++ prototype

```
bool startCollecting(bool detectStops = true);
```

C prototype

```
bool startCollecting(void* card, bool detectStops = true);
```

Inputs	Type	Description
detectStops	bool	Tell the PO8e to detect when the stream from the RZ is stopped
Returns		
pointer to PO8e instance		

Example

This code sample tells an existing PO8e object to begin collecting data.

C++

```
card->startCollecting(true);
```

C

```
startCollecting(card, true);
```

stopCollecting

Call this to stop collecting a data stream from the PO8e card.

C++ prototype

```
void stopCollecting();
```

C prototype

```
void stopCollecting(void* card);
```

Example

This code sample stops data collection on a PO8e object.

C++

```
card->stopCollecting(true);
```

C

```
stopCollecting(card, true);
```

waitForDataReady

This function provides a means to efficiently wait for data to arrive from the RZ unit.

C++ prototype

```
size_t waitForDataReady(int timeout = 0xFFFFFFFF);
```

C prototype

```
int waitForDataReady(void* card, int timeout = 0xFFFFFFFF);
```

Inputs	Type	Description
timeout	int	Maximum duration (in ms) to wait for streaming to begin.

Example

This code sample blocks execution until buffered data is ready on the card.

C++

```
card->waitForDataReady();
```

C

```
waitForDataReady(card);
```

samplesReady

Returns the number of samples (per channel) that are currently buffered.

C++ prototype

```
size_t samplesReady(bool *stopped = 0);
```

C prototype

```
int samplesReady(void* card, bool *stopped = 0);
```

Inputs	Type	Description
stopped	bool pointer	The value pointed to will be set to true if the underlying mechanisms detect that data has stopped flowing.
Returns		
numSamples	size_t or int	Number of samples (per channel) that are currently buffered

Example

This code returns the number of samples (per channel) currently buffered on the card and detects if streaming has stopped.

C++

```
bool stopped;
size_t numSamples = card->samplesReady(&stopped);
if (stopped)
    PO8e::releaseCard(card);
```

C

```
bool stopped;
int numSamples = samplesReady(card, &stopped);
if (stopped)
    releaseCard(card);
```

readChannel

Copy the data buffered for an individual channel. Note that this call does *not* advance the data pointer. Use calls to `flushBufferedData` to discard the data copied using this function.

The user is responsible for ensuring that the buffer is large enough to hold `nSamples * dataSampleSize()` bytes.

The optional offsets array should be `nSamples` long and will be populated with the data offset of each block. This allows a user to detect if the buffer on the RZ unit has overflowed.

C++ prototype

```
int readChannel(int chanIndex, void *buffer, int nSamples, int64_t *offsets =
NULL);
```

C prototype

```
int readChannel(void* card, int chanIndex, void *buffer, int nSamples,
int64_t *offsets);
```

Inputs	Type	Description
chanIndex	int	The channel to read data from (1-based)
buffer	void pointer	The location to write buffered data to
nSamples	int	The number of samples to read
offsets	int64_t pointer	The location to write the buffer indices to
Returns		
numSamples	int	Number of samples that were read

Example

This code sample reads 1 sample from channel 2 and stores it in buff.

C++

```
short buff[8192];
card->readChannel(2, buff, 1);
```

C

```
short buff[8192];
readChannel(card, 2, buff, 1);
```

readBlock

Copy the data buffered for all channels. Note that this call does *not* advance the data pointer. Use calls to `flushBufferedData` to discard the data copied using this function.

The data will be grouped by channel and the number of samples returned applies to all channels. The user is responsible for ensuring that the buffer is large enough to hold `nSamples * numChannels() * dataSampleSize()` bytes.

The optional offsets array should be nSamples long and will be populated with the data offset of each block. This allows a user to detect if the buffer on the RZ unit has overflowed.

C++ prototype

```
int readBlock(void *buffer, int nSamples, int64_t *offsets = NULL);
```

C prototype

```
int readBlock(void* card, void *buffer, int nSamples, int64_t *offsets);
```

Inputs	Type	Description
buffer	void pointer	The location to write buffered data to
nSamples	int	The number of samples to read
offsets	int64_t pointer	The location to write the buffer indices to
Returns		
numSamples	int	Number of samples that were read

Example

This code sample reads 1 sample from all channels, stores it in a buffer, and flushes that data from the card.

C++

```
short buff[1024];
card->readBlock(buff, 1);
card->flushBufferedData(1);
```

C

```
short buff[1024];
readBlock(card, buff, 1);
flushBufferedData(card, 1);
```

flushBufferedData

Releases samples from each buffered channel.

C++ prototype

```
void flushBufferedData(int numSamples = -1, bool freeBuffers = false);
```

C prototype

```
void flushBufferedData(void* card, int numSamples = -1, bool freeBuffers =
false);
```

Inputs	Type	Description
numSamples	int	Number of samples to release. Passing -1 releases all buffered samples
freeBuffers	bool	Controls the optional freeing of the underlying data buffers

Example

This code sample flushes one sample from all channels.

C++

```
card->flushBufferedData(1);
```

C

```
flushBufferedData(card, 1);
```

Hardware Information Retrieval

The methods in this group access information pertaining to the current data stream, including number of channels and sample size in bytes.

numChannels

Counts the number of channels in the current stream. This value is set in the PO8e HAL in Synapse (or Stream_Remote_MC macro in RPvdsEx). Changing the number of channels mid-stream triggers an error condition.

C++ prototype

```
int numChannels();
```

C prototype

```
int numChannels(void* card);
```

Returns	Type	Description
nChannels	int	Number of channels in the current data stream

Example

This code determines how many channels are in the current stream.

C++

```
int nChannels = card->numChannels();
```

C

```
int nChannels = numChannels(card);
```

numBlocks

Counts the number of blocks that the current stream is divided into. This value will always be 1 when using Synapse, or can be set in the Stream_Remote_MC macro in RPvdsEx.

Each block will contain the same number of channels, so dividing the value from `numChannels()` by this value will leave no remainder. Changing the number of blocks mid-stream triggers an error condition.

C++ prototype

```
int numBlocks();
```

C prototype

```
int numBlocks(void* card);
```

Returns	Type	Description
nBlocks	int	Number of blocks the current data stream is divided into

Example

This code determines how many blocks are in the current stream.

C++

```
int nBlocks = card->numBlocks();
```

C

```
int nBlocks = numBlocks(card);
```

dataSampleSize

Returns the size in bytes of each data sample (per channel). This value is set in the PO8e HAL in Synapse (or Stream_Remote_MC macro in RPvdsEx). Changing the data type mid-stream triggers an error condition.

C++ prototype

```
int dataSampleSize();
```

C prototype

```
int dataSampleSize(void* card);
```

Returns	Type	Description
dataSize	int	Size of each data sample in bytes

Example

This code determines how many bytes are in each sample.

C++

```
int dataSize = card->dataSampleSize();
```

C

```
int dataSize = dataSampleSize(card);
```

getLastError

This returns the most recent error.

C++ prototype

```
int getLastError();
```

C prototype

```
int getLastError(void* card);
```

Returns	Type	Description
errCode	int	The most recent error code

Example

This code returns the most recent error code.

C++

```
int errCode = card->getLastError();
```

C

```
int errCode = getLastError(card);
```

Examples

The example files below are installed with the TDT drivers package.

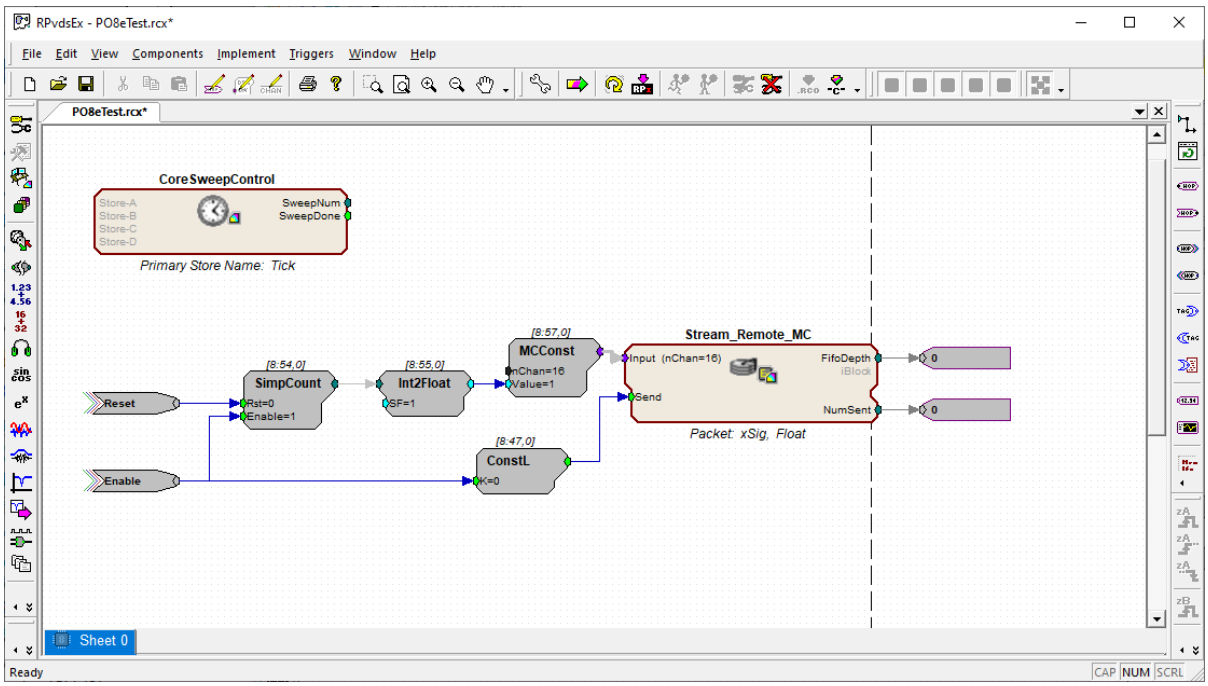
PO8eTest

```
C:\TDT\RPvdsEx\Examples\PO8e\PO8eTest.rcx
C:\TDT\RPvdsEx\Examples\PO8e\PO8eTest.exe
```

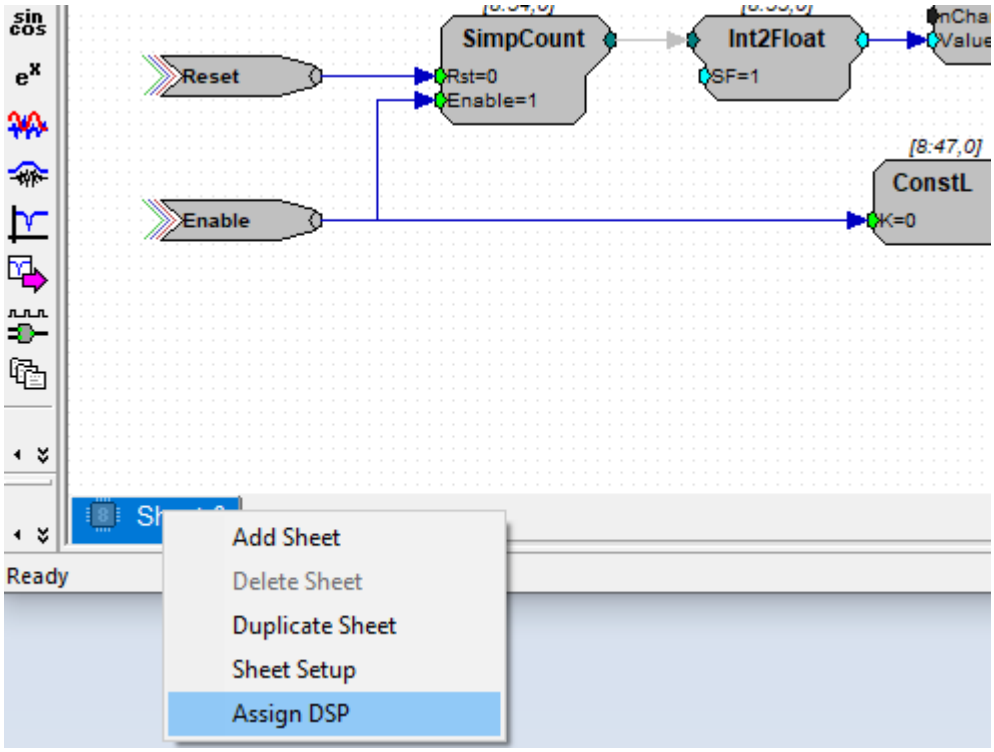
PO8eTest.exe connects to any PO8e card(s) in the PC, waits for a stream then displays the data rate that each PO8e card is receiving.

PO8eTest.rcx streams 256 channels of floats to the PO8e card at 6.1 kHz.

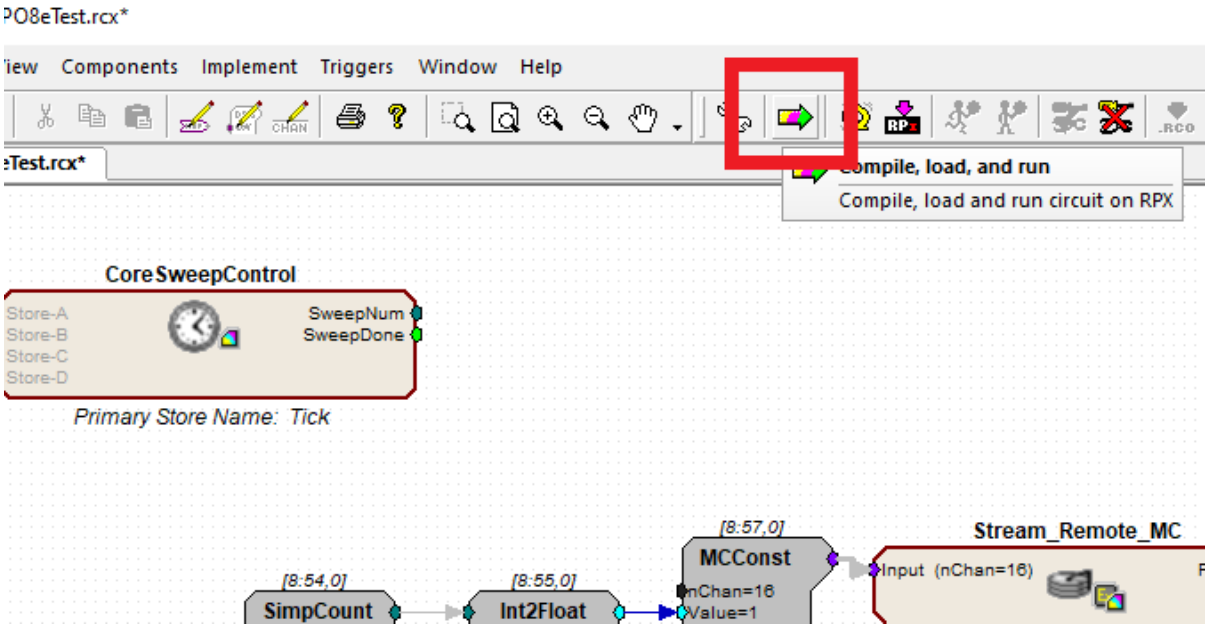
1. Run PO8eTest.exe test application first.
2. Verify that it connected to the PO8e card.
3. Open PO8eTest.rcx in RPvdsEx.



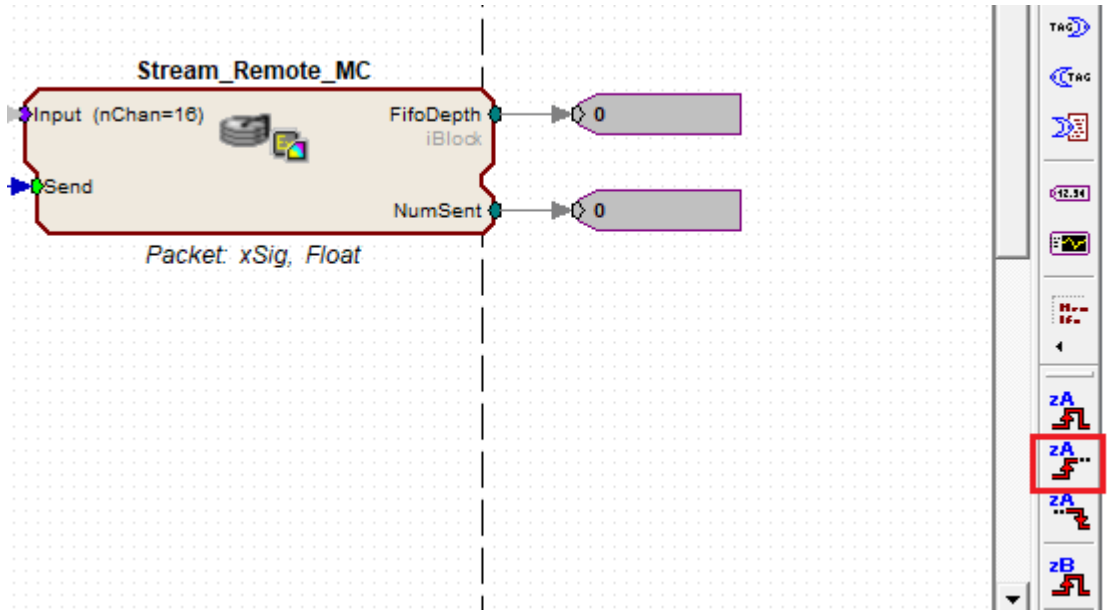
4. Assign the circuit to the DSPU or DSPS in your RZ.



5. Run the circuit.

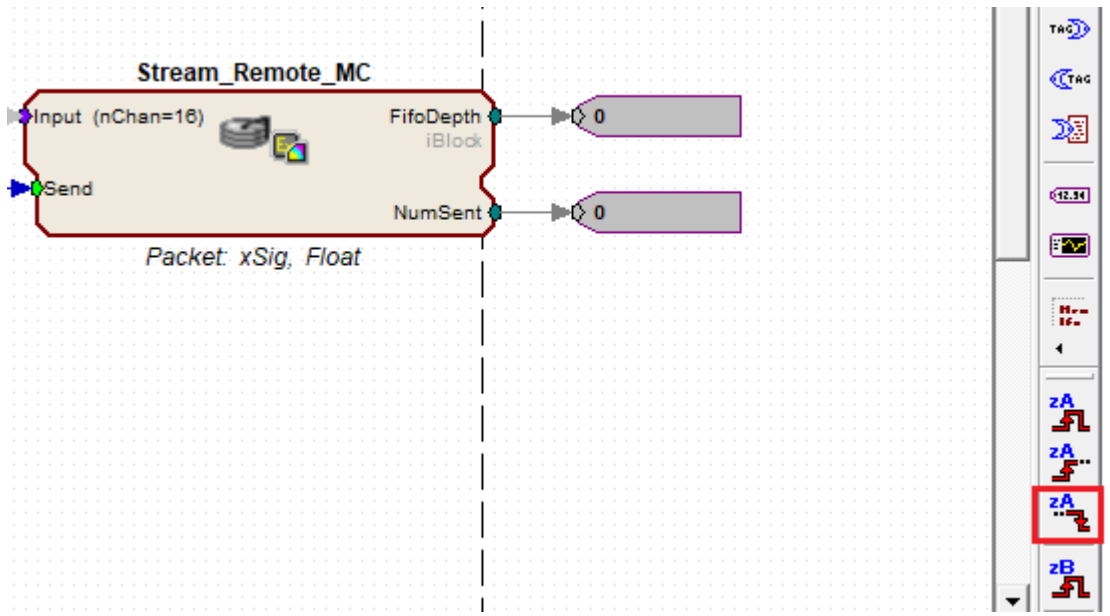


6. Set the zBusA trigger high to begin streaming.



7. Verify that PO8eTest.exe sees data.

8. Set the zBusA trigger low to stop streaming.



Source Files:

```
C:\TDT\RPvdsEx\Examples\PO8e\Example\
C:\TDT\RPvdsEx\Examples\PO8e\Include\
```

MATLAB

```
C:\TDT\RPvdsEx\Examples\PO8e\Matlab\
```

The PO8e_4card.m class wraps the PO8e Streaming Library. The PO8e_4cardTest.m example demonstrates reading from PO8e card(s) in MATLAB.

Python

```
C:\TDT\RPvdsEx\Examples\PO8e\Python64
```

The PO8e.py class wraps the PO8e Streaming Library and contains demo code for reading from PO8e card(s) in Python.

Linux Support

Some Linux library files and example build instructions are installed to:

```
C:\TDT\RPvdsEx\Examples\PO8e\Linux
```

Please contact TDT if you need assistance.

RZUDP

Custom software applications are fully supported for any computer language that supports IP network protocols.

Getting Started

TDT provides wrapper classes and demo files for communicating with the RZUDP interface from MATLAB or Python. You'll also need the IP address of the RZ UDP interface. See RZ-UDP Interface.

> **Note**
>
> If you are communicating with the emulated UDP interface in Corpus, use 'localhost' for the RZ IP address if your script is running on the same computer as Corpus, or use the computer's IP address if your script is running on a different computer on the network.

MATLAB

You can download the latest MATLAB SDK files here.

The TDTUDP class installs to:

```
C:\TDT\TDTMatlabSDK\TDTSDK\UDP
```

Example scripts install to:

```
C:\TDT\TDTMatlabSDK\Examples\UDP
```

Python

The Python TDTUDP class interfaces with the RZ UDP hardware. It is available in the tdt pypi package (pip install tdt).

Other Languages

C++ example files are available on request. A C# implementation can be found on github.

MATLAB Examples

Reading from RZ UDP

```
RZ_IP = '10.1.0.100'; % find remote IP address of RZ device using zBusMon

% read demo
u = TDTUDP(RZ_IP, 'TYPE', 'single', 'VERBOSE', 1);

while 1
    u = u.read();
    if isempty(u.data)
        continue
    end
    u.data
    % do something with the data here
end
```

Writing to RZ UDP

```
RZ_IP = '10.1.0.100'; % find remote IP address of RZ device using zBusMon

% write demo
u = TDTUDP(RZ_IP);

for i = 1:10
    cmd = [i zeros(1,15)];
    u = u.write(cmd);
end
```

Python Examples

Reading from RZ UDP

```
import tdt

RZ_IP = '10.1.0.100'

# this example has UDPSend hal connected to SortBinner gizmo for spike counts
udp = tdt.TDTUDP(host=RZ_IP, sort_codes=4, bits_per_bin=4)

while 1:
    data = udp.recv()

    # if looking at binner packets, extract sort codes
    channel = 4
    sort_code = 2
    print('CHANNEL:', channel, 'SORT:', sort_code, '\t', data[sort_code-1]
[channel-1], end='\t\t\t\r')
```

Writing to RZ UDP

```
import time
import tdt

RZ_IP = '10.1.0.100'

udp = tdt.TDTUDP(host=RZ_IP, send_type=np.float32)

SEND_PACKETS = 1
ct = 0
while 1:
    ct += 1
    fakedata = range(ct % 10, SEND_PACKETS + ct % 10)
    if udp.send_type == float:
        fakedata = [x * 2. for x in fakedata]
    udp.send(fakedata)
    time.sleep(.1) # slow it down a bit
```

Reading and Writing

```
import tdt

RZ_IP = '10.1.0.100'

udp = tdt.TDTUDP(host=RZ_IP, send_type=np.float32, recv_type=np.float32)

SEND_PACKETS = 8
ct = 0
while 1:
    ct += 1
    fakedata = range(ct % 10, SEND_PACKETS + ct % 10)
    if udp.send_type == float:
        fakedata = [x * 2. for x in fakedata]

    data = udp.recv()
    print(data)

    udp.send(fakedata)
```

> **Note**
>
> The listening port on the UDP Ethernet interface is 22022 and cannot be changed.

UDP Interface Performance

The UDP interface is a 10 Mb Ethernet interface, but the usable bandwidth is significantly lower due to overhead and serialization of the data. All data is transferred as single channel or multi-channel packets from Synapse. Sort code data can be highly compressed to increase speed using the SortBinner gizmo.

The table below displays the expected throughput for different channel counts.

Channels (32-bits)	Rate (Hz)
1	600
8	500
16	400
32	300
64	150
128	100
192	50

SynapseLive Examples

Online Signal Averaging Example

Import strobe store gizmo data into MATLAB using SynapseLive during the experiment
Plot the average waveform
Good for Evoked Potential visualization

Download M File

Download Experiment File

Housekeeping

Clear workspace and close existing figures. Add SDK directories to Matlab path.

```
close all; clc;
[MAINEXAMPLEPATH,name,ext] = fileparts(cd); % \TDTMatlabSDK\Examples
[SDKPATH,name,ext] = fileparts(MAINEXAMPLEPATH); % \TDTMatlabSDK
addpath(genpath(SDKPATH));
```

Variable Setup

Set up the variables for the data you want to extract. We will extract a single channel from a fixed duration strobed storage gizmo.

```
EVENT = 'StS1';
CHANNEL = 1;
```

show the last N waveforms in the plot.

```
N = 5;
```

Set KEEPALL to 0 to only show the running average of the last N waveforms. Otherwise, all waveforms in the block are included in the average.

```
KEEPALL = 0;
```

Setup SynapseLive

```
t = SynapseLive('MODE', 'Preview', 'EXPERIMENT', 'OnlineAveragingDemo'); % we
will default to 'Preview' mode
t.TYPE = {'snips'}; % we only care about these types of events
t.VERBOSE = false;
first_pass = true;
```

```
Connected to TANK: F:\Tanks\OnlineAveragingDemo-220119-151526, BLOCK: TempBlk
Waiting for initial data.............................done
```

The Main Loop

```
prevWaves = cell(1,N);
nsweeps = 0;
while 1

    % slow it down a little
    pause(1)

    % get the most recent data, exit loop if the block has stopped
    if isempty(t.update)
        break
    end

    % read the snippet events
    r = t.get_data(EVENT);
    if isstruct(r)
        if ~isnan(r.data)
            % get our channel of data
            chan_data = r.data(r.chan == CHANNEL,:);
            nsize = size(chan_data,1);

            % cache the waveforms in our circular buffer
            prevWaves = circshift(prevWaves, -nsize);
            for i = 1:(min(nsize, N))
                prevWaves{i} = chan_data(end-(i-1),:);
            end

            % find average signal
            cache_ind = ~cellfun('isempty', prevWaves);
            if KEEPALL == 0
                % if we are only keeping the previous N, do average on just
those
                avg_data = mean(cell2mat(prevWaves(cache_ind)'), 1);
            else
                if first_pass
                    first_pass = false;
                    nsweeps = nsize;
                    avg_data = new_mean;
                else
                    new_mean = mean(chan_data, 1);
                    % add new average into the old average
                    avg_data = (avg_data .* nsweeps + new_mean * nsize) /
(nsweeps + nsize);
                end
            end

            nsweeps = nsweeps + nsize;

            % plot the preview N waves in gray
            t_ms = 1000*(1:numel(avg_data)) / r.fs;
            plot(t_ms, cell2mat(prevWaves(cache_ind)')','Color', [.85 .85 .
```

```
85]); hold on;

            % plot the average signal in thick blue
            plot(t_ms, avg_data, 'b', 'LineWidth', 3); hold off;

            % finish up plot
            title(sprintf('nsweeps = %d, last %d shown', nsweeps, N));
            xlabel('Time, ms','FontSize',12)
            ylabel('V', 'FontSize', 12)
            temp_axis = axis;
            temp_axis(1) = t_ms(1);
            temp_axis(2) = t_ms(end);
            axis(temp_axis);

            % force the plots to update
            try
                snapnow
            catch
                drawnow
            end

            % for publishing, end early
            if nsweeps > 30
                t.SYN.setModeStr('Idle'); % set to idle mode
                break
            end
        end
    end
end
```

Runtime Output

Online Raster PSTH Example

Import snippet and epoc data into MATLAB using SynapseLive while the experiment is running
Generate peri-event raster and histogram plots over all trials
Good for stim-response experiments, such as optogenetic or electrical stimulation, where you need immediate visual feedback

Download M File

Download Experiment File

Housekeeping

Clear workspace and close existing figures. Add SDK directories to MATLAB path.

```
close all; clc;
[MAINEXAMPLEPATH,name,ext] = fileparts(cd); % \TDTMatlabSDK\Examples
[SDKPATH,name,ext] = fileparts(MAINEXAMPLEPATH); % \TDTMatlabSDK
addpath(genpath(SDKPATH));
```

Setup

Setup the variables for the data you want to extract. We will extract channel 1 from the eNe1 snippet data store, created by the PCA Sorting gizmo, and use the Tick as our stimulus onset.

```
REF_EPOC = 'Tick';
EVENT = 'eNe1';
CHANNEL = 1;
SORTCODE = 0; % set to 0 to use all sorts
TRANGE = [-0.3, 0.8]; % start time, duration
DO_RASTER = 1; % set to 0 to only see histogram
```

Setup SynapseLive

```
t = SynapseLive('MODE', 'Preview', 'EXPERIMENT', 'RasterPSTHdemo'); % we will
default to 'Preview' mode
t.NEWONLY = 0;  % read all available events in the block on every iteration
t.TIMESTAMPSONLY = 1;  % we don't care what the snippets look like, just
their timestamps
t.TYPE = {'snips', 'epocs', 'scalars'}; % we only care about these types of
events
t.VERBOSE = false;
```

```
Connected to TANK: F:\Tanks\RasterPSTHdemo-220120-082308, BLOCK: TempBlk
Waiting for initial data.............done
```

The Main Loop

```
% Set figure size base on number of plots
if DO_RASTER
    h = figure('Position',[100, 100, 500, 800]);
else
    h = figure('Position',[100, 100, 500, 400]);
end
while 1

    % slow it down a little
    pause(1)

    % get the most recent data, exit loop if the block has stopped.
    if isempty(t.update)
        break
    end

    % read the snippet and event timestamp data.
    r = t.get_data(EVENT);
    if isstruct(r)
        if ~isnan(r.ts)

            % do our timestamp filtering
            if DO_RASTER
                data = TDTfilter(t.data, REF_EPOC, 'TIME', TRANGE);
            else
                data = TDTfilter(t.data, REF_EPOC, 'TIME', TRANGE, 'TIMEREF',
1);
            end

            % do our channel and sort code filtering
            if SORTCODE ~= 0
                i = find(data.snips.(EVENT).chan == CHANNEL & data.snips.
(EVENT).sortcode == SORTCODE);
            else
                i = find(data.snips.(EVENT).chan == CHANNEL);
            end

            % grab only the relevant timestamps
            try
                TS = data.snips.(EVENT).ts(i);
                if isempty(TS)
                    continue
                end
            catch
                continue
            end

            % that's it for the data extraction, now we plot
            num_trials = size(data.time_ranges, 2);
            if DO_RASTER
```

```
                % match timestamp to its trial
                all_TS = cell(num_trials, 1);
                all_Y = cell(num_trials, 1);
                for trial = 1:num_trials
                    trial_TS = TS(TS >= data.time_ranges(1, trial) & TS <
data.time_ranges(2, trial));
                    all_TS{trial} = trial_TS - data.time_ranges(1, trial) +
TRANGE(1);
                    all_Y{trial} = trial * ones(numel(trial_TS), 1);
                end
                all_X = cat(1, all_TS{:});
                all_Y = cat(1, all_Y{:});

                % plot raster
                subplot(2,1,1)
                hold on;
                plot(all_X, all_Y, '.', 'MarkerEdgeColor','k', 'MarkerSize',
10)
                line([0 0], [1, trial], 'Color','r', 'LineStyle','-',
'LineWidth',3)
                axis tight; axis square;
                set(gca, 'XLim', [TRANGE(1), TRANGE(1)+TRANGE(2)]);
                ylabel('trial number')
                xlabel('time, s')
                title(sprintf('Raster ch=%d sort=%d, %d trials', CHANNEL,
SORTCODE, num_trials))
                hold off;
                TS = all_X;
                subplot(2,1,2)
            end

            % plot PSTH
            NBINS = 20;
            if NBINS > 0
                hist(TS, NBINS);
                hold on;
                N = hist(TS, NBINS);
                line([0 0], [0, max(N)*1.1], 'Color','r', 'LineStyle','-',
'LineWidth',3);
                hold off;
                axis tight; axis square;
                set(gca, 'XLim', [TRANGE(1), TRANGE(1)+TRANGE(2)]);
                ylabel('number of occurrences')
                xlabel('time, s')
                title(sprintf('Histogram ch=%d sort=%d, %d trials', CHANNEL,
SORTCODE, num_trials))
            end

            % force the plots to update
            try
                snapnow
```

```
            catch
                drawnow
            end

            % for publishing, end early
            if num_trials > 30
                t.SYN.setModeStr('Idle'); % set to idle mode
                break
            end
        end
    end
end
```

Runtime Output

Online Stream Example

Read a streaming data store into MATLAB from Tank server during a recording

Download M File

Download Experiment File

Housekeeping

Clear workspace and close existing figures. Add SDK directories to MATLAB path.

```
close all; clc;
[MAINEXAMPLEPATH,name,ext] = fileparts(cd); % \TDTMatlabSDK\Examples
[SDKPATH,name,ext] = fileparts(MAINEXAMPLEPATH); % \TDTMatlabSDK
addpath(genpath(SDKPATH));
```

Setup

```
EVENT = 'EEG1';
t = SynapseLive('MODE', 'Preview', 'EXPERIMENT', 'OnlineStreamDemo');
t.VERBOSE = false;
```

```
Connected to TANK: F:\Tanks\OnlineStreamDemo-220120-091932, BLOCK: TempBlk
Waiting for initial data.............done
```

Main Loop

```
first_pass = true;
while 1

    % slow it down
    pause(1)

    % get the most recent data, exit loop if the block has stopped.
    if isempty(t.update)
        break
    end

    % grab the latest events
    r = t.get_data(EVENT);
    if isstruct(r)

        % plot them
        ts = linspace(t.T1, t.T2, max(size(r.data))-1);
        plot(ts, r.data(:,1:end-1)')
        title(EVENT)
        xlabel('Time, s')
        ylabel('V')
        axis tight

        % force the plots to update
        try
            snapnow
        catch
            drawnow
        end
    end

    % for publishing, end early
    if t.T2 > 30
        t.SYN.setModeStr('Idle'); % set to idle mode
        break
    end
end
```

Runtime Output

APIStreamer Examples

Stream Data with APIStreamer

Stream multi-channel data directly from the APIStreamer gizmo running on the hardware
Good for visualization and online analysis

Download M File

Download Experiment File

Housekeeping

Clear workspace and close existing figures. Add SDK directories to Matlab path.

```
close all; clc;
[MAINEXAMPLEPATH,name,ext] = fileparts(cd); % \TDTMatlabSDK\Examples
[SDKPATH,name,ext] = fileparts(MAINEXAMPLEPATH); % \TDTMatlabSDK
addpath(genpath(SDKPATH));
```

Variable Setup

load and run demo experiment

```
syn = SynapseAPI('localhost');
if syn.setCurrentExperiment('APIStreamerMCDemo') == 0
    error('trouble loading experiment');
end
syn.setModeStr('Preview');
```

Setup APIStreamer

```
s = APIStreamer('GIZMO', 'APIStreamerMC1', 'HISTORY', 1, 'CALLBACK',
@my_api_callback);
```

```
APIStreamer on
Resetting APIStreamer
16 channels in 1600000 sample buffer at 6103.515625 Hz
APIStreamer off
```

The Main Loop

```
while 1
    [data, ts] = s.get_data();
    plot(ts, data);
    axis tight; xlabel('time, s'); ylabel('V')

    % force the plots to update
    try
        snapnow
    catch
        drawnow
    end

    % for publishing, end early
    if max(ts) > 30
        s.stop()
        syn.setModeStr('Idle'); % set to idle mode
        break
    end
end
```

```
new data:    1000 samples
```

```
new data:    1000 samples
```

```
new data:    1000 samples
```

```
new data:    1000 samples
```

```
new data:    1000 samples
```

```
new data:    1000 samples
```

```
new data:    1000 samples
```

```
new data:    1000 samples
```

```
new data:    1000 samples
```

```
new data:    1000 samples
```

```
new data:    1000 samples
```

```
APIStreamer off
```

Runtime Output

Spectrum Analyzer

Read a single channel and plot the power spectrum over time
Good for visualization and online analysis

Download M File

Download Experiment File

Housekeeping

Clear workspace and close existing figures. Add SDK directories to MATLAB path.

```
close all; clc;
[MAINEXAMPLEPATH,name,ext] = fileparts(cd); % \TDTMatlabSDK\Examples
[SDKPATH,name,ext] = fileparts(MAINEXAMPLEPATH); % \TDTMatlabSDK
addpath(genpath(SDKPATH));
```

Variable Setup

load and run demo experiment

```
syn = SynapseAPI('localhost');
if syn.setCurrentExperiment('APIStreamer1ChDemo') == 0
    error('trouble loading experiment');
end
syn.setModeStr('Preview');
pause(4);
```

Setup APIStreamer

```
s = APIStreamer('GIZMO', 'APIStreamer1Ch1','HISTORY', 30, 'DO_FFT', 1,
'WINSIZE', 1, 'FREQ', [1 20]);

h = figure;
set(gcf, 'Position', [800, 200, 600, 900])
```

```
APIStreamer on
Resetting APIStreamer
1 channels in 100000 sample buffer at 6103.515625 Hz
APIStreamer off
```

The Main Loop

```
while size(findobj(h)) > 0 % run while the figure is open

    [fft_data, ts, fft_freq] = s.get_data();
    if ~any(fft_data(:)), continue, end

    psdx = fft_data/(s.fs*size(fft_data,2));
    %20*log10(fft_data) for power in dB

    last_psd = psdx(:,end);

    spectral_peak = max(last_psd);
    peak_freq = fft_freq(last_psd == spectral_peak);

    % calculate power and normalize color bar
    Power = fft_data;

    if any(isinf(Power(:))), continue, end

    Z_STD = 6 * std(Power(:));
    Z_MEAN = mean(Power(:));

    % plot it
    subplot(2,1,1);
    plot(fft_freq, last_psd);
    grid on
    title('Periodogram Using FFT')
    xlabel('Frequency (Hz)')
    ylabel('Energy/Frequency (1/Hz)')

    subplot(2,1,2);
    imagesc(ts, fft_freq, Power, [Z_MEAN-Z_STD, Z_MEAN+Z_STD]);
    h_colorbar = colorbar;
    colormap(jet);
    set(gca,'fontsize',12);
    xlabel('Time (s)','fontsize',15);
    ylabel('Frequency (Hz)','fontsize',15);
    ylabel(h_colorbar,'Energy','Rotation',-90,'fontsize',
15,'VerticalAlignment','baseline');
    title(sprintf('Spectrogram'))
    axis xy;

    % force the plots to update
    try
        snapnow
    catch
        drawnow
    end

    % for publishing, end early
    if max(ts) > 30
```

```
        s.stop()
        syn.setModeStr('Idle'); % set to idle mode
        break
    end
end
```

```
APIStreamer off
```

Runtime Output