***Photometry Extract and Analysis Protocol – CAS v01***

Fiber photometry extract hub that is compatible with multiple analysis platforms. Zip folder for extract hub (20231102\_PhAT\_extract\_CAS) contains scripts for:

1. Extracting green/red/isosbestic traces from Synapse and storing automatically generated (e.g., TTL triggered) time stamps in file outputs compatible with CAS analysis, Bruchas lab analysis, and PhAT GUI
2. Importing existing time stamp files (BORIS or txt file formatted) and filtering them to only retain the desired subset of time stamps. Saves in BORIS and txt file format regardless of input type.
3. Filtering linear track-generated time stamps of multiple behavioral events and converting them into BORIS and txt file formats.

Included example data: a spreadsheet for photometry trial info (nameGenerator\_sample\_v02.csv) and multiple different types of recordings to demonstrate how to use different features of the extract hub (red vs green sensors, different LED channel names, automatically generated TTL time stamps vs manually input text files).

**Setup: Program and data location**

1. Unzip the Extract Hub, store it on your computer in your MATLAB path. Make sure the subfolder ‘functions’ containing the ‘TDTbin2mat’ folder exists. The three main scripts, ‘tdt\_Extract\_CAS\_v06.m,’ ‘PhAT\_StampFilter\_CAS\_v04.m,’ and ‘LinearTrack\_StampFilter\_CAS\_v01.m’ should also be in the main Extract Hub folder.
2. Use the folder ‘name\_spreadsheets’ to contain the csv spreadsheet(s) storing your photometry trial info. You may prefer to have different csvs for different sets of experiments for organization purposes, but a single spreadsheet for all recordings would also work. It can be, but does not need to be, a subfolder of the Extract Hub (just be sure to update the pathname pointing to this folder in both scripts, described later in the protocol).
3. Use the folder ‘data’ to contain all data outputs from these scripts. It can be, but does not need to be, a subfolder of the Extract Hub.
   1. *Example: The included ‘data’ folder is already setup and ready to use. The ‘sample\_data’ folder is also setup properly as an example, already containing the results for the example data.*
4. Make sure the following 7 subfolders are within the ‘data’ folder to contain the different forms of data outputs: ‘Bruchas\_matfiles\_extracted’; ‘CAS\_matfiles\_extracted’; ‘figures\_rawtrace’; ‘PhAT\_matfiles’; ‘PhAT\_stamp\_csv’; ‘PhAT\_stream\_csv’; ‘timestamps’; ‘lineartrack\_behevents’
   1. Note: data subfolders MUST be named in this way to be compatible! Once data is exported to these folders you can copy the exported files to other locations if you prefer.
   2. *The included ‘data’ folder contains the necessary subfolders, as does ‘sample\_data’ with the sample data results already exported to them.*

**Enter trial data in csv spreadsheet**

1. Create a csv that will contain your photometry trial data. Each row corresponds with one recording.
   1. *Example:* *‘nameGenerator\_sample\_v02.csv’ in the ‘name\_spreadsheets’ subfolder.*
2. Enter the data for each of your trials in its own row. If a recording does not have information for one of the columns, enter ‘NA’. Do not enter any values with spaces in any cells (exception: notes column).
   1. See Table 1 for explanation of what information to enter in each column.
3. There cannot be empty rows in between rows containing trial info, if you would like to have some separation between sets of recordings enter a row of ‘NAs’ or similar in between.
   1. *Example: rows 6 and 11 in the sample spreadsheet ‘nameGenerator\_sample\_v01.csv’*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column Title** | **Information to Include** | **Additional Notes** | **Must contain non-NA value?** | **Example** |
| tankname | name of tank folder containing TDT recording | - | Y | Carrie\_TwoGFP\_DualColor-230725-101147 |
| blockname | name of file in tank for this specific recording | - | Y | gcamp\_260517M3-230728-105424 |
| animal\_ID | descriptive name of the recording subject | user preferred format should be fine, I use virus\_cage#  \_sex\_animal# | Y | gcamp\_260517M3 |
| expname | descriptive name of the type of experiment | - | Y | TailLift10s\_Dual470 |
| date | date the experiment was recorded | okay to use user preferred date format (e.g. YYYYMMDD) | Y | \_Jul28 |
| timestamps | file name of the text file containing the associated time stamps | can leave NA at first then enter later if using extract code to generate. | N | gcamp\_260517M3  \_Jul28\_taillift\_times.txt |
| eventduration | duration of behavioral event of interest (e.g. tail lift) in seconds | can use average if not consistent, this value won’t affect analysis in any way | Y | 10 |
| sessionlength | length of entire recording in minutes, rounded down to nearest minute | do not set this to a number greater than your session’s actual length in minutes | Y | 20 |
| delay1 | delay (if any) between start of photometry and start of time stamps in text file, in minutes | - | N | 0 |
| fp\_system | descriptive name of photometry equipment used | this is just for note-keeping, no functional purpose | N | BruchasCartDual470 |
| isosbestic | TDT storename associated with your isosbestic channel, if recorded | must enter a storename to extract isosbestic trace | N | 405A |
| green | TDT storename associated with your green channel, if recorded | must enter a storename to extract green trace | N | 470A |
| red | TDT storename associated with your red channel, if recorded | must enter a storename to extract red trace | N | 565B |
| TTL1 | TDT storename associated with your first TTL/epoch, if recorded | must enter a storename to extract TTLs | N | Epo1 |
| TTL2 | TDT storename associated with your second TTL/epoch, if recorded | must enter a storename to extract TTLs | N | Epo2 |
| TTL1\_beh | Behavior encoded by the first TTL/epoch | used to name time stamp files of this behavior | N | solenoid |
| TTL2\_beh | Behavior encoded by the second TTL/epoch | used to name time stamp files of this behavior | N | lick |
| TT1\_type | Denotes whether behavior encoded by first TTL/epoch is ‘state’ (has start/stop times) or ‘point’ (discrete event) | must be either state or point if extracting TTL1, NA if not | N | state |
| TTL2\_type | Denotes whether behavior encoded by second TTL/epoch is ‘state’ (has start/stop times) or ‘point’ (discrete event) | must be either state or point if extracting TTL2, NA if not | N | point |
| lintrack\_vidname | name of the trial/subject video used in the linear track behavioral event spreadsheet associated with this trial | only necessary if trial was recorded with linear track | N | gcamp\_260517M3-230728-105424.analysis |
| lintrack\_eventname | name of the xlsx spreadsheet containing this trial’s linear track behavior/event data | only necessary if trial was recorded with linear track | N | 20231102\_behevents.xlsx |
| extracted | leave empty to start, but add ‘yes’ after extracting as a reminder that it’s already been done | this is just for note-keeping, no functional purpose | N | yes |
| notes | any additional notes about the signal, experiment, recording day, etc. that you may want record of | does not need to follow any specific formatting | N | animal had weak signal |

Table 1: Explanation of information recorded in each spreadsheet column.

**Setup extract script with your paths**

1. Open tdt\_Extract\_CAS\_v06.m in MATLAB. All lines where you would potentially need to make changes when using this script (either once to setup your paths or regularly on a trial-by-trial basis) are contained between the %%%%%% (lines 31 to 71). The rest of the script should never need to be adjusted to work as described here.
2. On line 37, update nameGenerator to the name of your trial info csv that was generated in the previous step.
3. On line 47, update path\_to\_data.tank with the pathname to your main tanks folder where the TDT recordings are stored.
   1. *The included example has the tanks as a subfolder in the sample\_data folder, but this is not necessary. The tanks folder can be at any location on your computer.*
4. On line 50, update path\_to\_data.spreadsheet with the pathname to the folder containing your spreadsheet.
5. On line 54, update path\_to\_data.store with the pathname to your data folder created in step 3 of this protocol’s first section.

**Run the extract script**

1. Open tdt\_Extract\_CAS\_v06.m in MATLAB.
2. On line 33, update the ‘trials’ variable so that it is equal to the rows of your spreadsheet that you would like to extract.
   1. *Example: trials = [2:5 7:10] to extract rows 2, 3, 4, 5, 7, 8, 9, and 10.*
3. On line 37, check that the nameGenerator variable is set to the name of the csv that you are trying to extract data for.
4. On line 40, set the ‘resampf’ variable to the factor by which you want to down sample your data. 100 is my default.
5. On line 44, check that ‘save\_data’ is set to 1. This will allow the generated data files to be saved to the data locations setup in the first section. If you want to just check a trial’s variables without saving the resulting data files, set save\_data to 0.
6. Hit run, the script will run through the loop for each trial you requested to extract and save the streaming data and TTL time stamps automatically.
7. The following files will ALWAYS be saved:
   1. Mat file to the Bruchas\_matfiles\_extracted folder containing the variables Dts, raw405, raw470, raw565, subdat, data1, and data2. (Note: raw 470 contains whatever was saved in the ‘green’ channel, even if the LED was not 470. Similar concept applies for raw405 and raw565) compatible with standard Bruchas lab analysis.
   2. Mat file to the CAS\_matfiles\_extracted folder containing FPvalues, FPvarnames, and Fs compatible with CAS custom analysis.
   3. Svg image to the figures\_rawtrace folder containing a plot of the raw channels.
   4. Mat file to the PhAT\_matfiles folder containing the structs block\_stamps and block\_streams with all of the recorded signal and time stamp data. (Note: this is not uploaded to the PhAT GUI for analysis, but it contains all of the trial’s info in PhAT format in a way that’s easy to view and work with in MATLAB).
   5. Csv file to the PhAT\_stream\_csv folder containing the raw trace data in PhAT format, down sampled at your ‘resampf’ factor.
8. The following files are SOMETIMES saved depending on if the trial had automatic time stamping:
   1. Csv file to the PhAT\_stamp\_csv folder containing the automatically generated time stamp series in BORIS format (which is compatible with PhAT).
   2. Txt file to the timestamps folder containing the automatically generated time stamp series in a list (standard Bruchas/CAS analysis format).
9. Examples for all these file types can be found in the sample\_data subfolders.

**Setup time stamp script with your paths**

1. If your generated time stamps contain excess stamps that you would like to filter out (e.g., keep only time stamps that are at least 10 seconds apart) and/or to convert a manually entered text file of time stamps into BORIS format for use with PhAT, use PhAT\_StampFilter\_CAS\_v04.m.
2. Open PhAT\_StampFilter\_CAS\_v04.m in MATLAB.
3. On line 74, update nameGenerator to the name of your trial info csv that was generated in the previous step.
4. On line 77, update path\_to\_data.spreadsheet with the pathname to the folder containing your spreadsheet.
5. On line 81, update path\_to\_data.store with the pathname to your data folder created in step 3 of this protocol’s first section.

**Run the time stamp filtering script**

1. Open PhAT\_StampFilter\_CAS\_v04.m in MATLAB.
2. On line 26, update the ‘trials’ variable so that it is equal to the rows of your spreadsheet that you would like to extract.
   1. *Example: trials = [2:5] to extract rows 2, 3, 4, and 5.*
3. Adjust the source time stamp parameters (lines 29 to 50) to the appropriate settings.
   1. If you are filtering down TTL-generated time stamps, on line 33 set input.phatmat = 1. Make sure to also set input.txt = 0. On line 35, set input.series to the TTLs the trim settings should be applied to.
      1. *Example 1: Air puff was recorded as only TTL, input.series = {‘TTL\_1’}.*
      2. *Example 2: Solenoid and lick were both recorded as TTLs and both should be filtered, input.series = {‘TTL\_1’ ‘TTL\_2’};*
      3. *Example 3: Solenoid and lick were both recorded as TTLs but only licks should be filtered, input.series = {‘TTL\_2’};*
   2. If you are converting a manually generated time stamp text file into BORIS format (and/or filtering it down to a subset of stamps), set input.txt = 1. Make sure to also set input.phatmat = 0. Set input.beh to the name of the behavior encoded by the time stamps (e.g. ‘airpuff’)
      1. If the text file name is stored in the timestamps column of your csv spreadsheet, the script can read and open that file name automatically, in which case input.txt\_auto = 1.
      2. If your txt file isn’t named on the spreadsheet or if you have a separate time stamp series you want to input manually, set input.txt\_manual = 1 and set input.txtfile to the name of the desired text file.
      3. Note for manually entering a text file: while the original file will not be altered, the generated filtered txt/BORIS files will be named according to the convention of the spreadsheet row, not based on the original file name.
      4. *Example: A manually entered time stamp text file containing tail lift times that is not on the spreadsheet. Input.phatmat = 0; input.txt = 1; input.beh = ‘taillift’; input.txt\_auto = 0; input.txt\_manual = 1; input.txtfile = ‘taillifttimes.txt’;*
      5. *Example: A manually generated time stamp text file containing airpuff times that is named in the spreadsheet’s timestamp column. Input.phatmat = 0; input.txt = 1; input.beh = ‘airpuff; input.txt\_auto = 1; input.txt\_manual = 0; input.txtfile =[ ];*
4. Adjust the filtering variables (lines 51 to 70) to the desired parameters.
   1. To exclude all time stamps occurring before a designated time in the recording, set trim.minstart to the minimum session time that must be reached before time stamps are kept.
      1. *Example: For the air puff recordings in the sample data, a TTL was recorded when the Arduino program started even though the first puff occurred after a 5 minute baseline. To exclude time stamps prior to the first actual puff, trim.minstart = 250;*
   2. Set trim.interval to the minimum time that must pass in between time stamps that are kept.
      1. *Example: To analyze signal around lick bouts instead of to each individual lick event, define a lick bout threshold as lasting no more than 10s, trim.interval = 10;*
   3. Set trim.laststamp to the minimum distance from the end of the recording (in seconds) that the last kept time stamp should be.
      1. *Example: If you analyze data in 30s time windows around each event, but the last recorded time stamp occurred in the last 10s of the recording, you will not be able to generate your full desired time window around each stamp. Set trim.laststamp = 31;*
5. On line 69, check that ‘save\_data’ is set to 1. This will allow the generated data files to be saved to the data locations setup in the previous section. If you want to just check a trial’s variables without saving the resulting data files, set save\_data to 0.
6. Hit run, the script will run through the loop for each trial you requested to filter.
7. The new filtered time stamps are saved as both text files and BORIS formatted csv files automatically.

**Setup linear track stamp filter script with your paths**

1. Open LinearTrack\_StampFilter\_CAS\_v01.m in MATLAB.
2. On line 51, update nameGenerator to the name of your trial info csv that was generated in the previous step.
3. On line 54, update path\_to\_data.spreadsheet with the pathname to the folder containing your spreadsheet.
4. On line 58, update path\_to\_data.store with the pathname to your data folder created in step 3 of this protocol’s first section.

**Run the linear track stamp filter script**

1. Open your raw behavioral events xlsx book from the linear track (example in sample\_data 🡪 lineartrack\_behevents folder: ‘behavevents.xlsx’).
2. Create a new xlsx book (example in sample\_data 🡪 lineartrack\_behevents folder: ‘20231102\_behevents.xlsx’), copying over *only* the sheets of individual behavioral events (in example, sheets 2-6 from behavevents.xlsx).
   1. Rename the sheets in your new book to just the name of the behavior they encode.
   2. Create a title row at the top of each sheet with ‘vidname’ in column 1, ‘start’ in column 2, and ‘stop in column 3 (see 20231102\_behevents.xlsx for example).
      1. This new behavioral events book may be used for multiple different trials, as many different ‘vidnames’ could be stored in one xlsx book. Make sure that for a given linear track trial, the trial’s nameGenerator spreadsheet row contains the video name matching its corresponding vidname from the xlsx book in the lineartrack\_vidname column.
      2. Make sure that the filename for the new, simplified linear track xlsx book created in this step is stored in the lintrack\_eventname column of the nameGenerator spreadsheet.
         1. *See rows 20 and 21 of nameGenerator\_sample\_v02.csv as examples.*
3. Open LinearTrack\_StampFilter\_CAS\_v01.m in MATLAB.
4. On line 21, update the ‘trials’ variable so that it is equal to the rows of your spreadsheet that you would like to extract.
   1. *Example: trials = [20 21] to get linear track time stamp data for rows 20 and 21.*
5. Adjust the filtering variables (lines 32 to 43) to the desired parameters.
   1. To exclude all time stamps occurring before a designated time in the recording, set trim.minstart to the minimum session time that must be reached before time stamps are kept.
   2. Set trim.interval to the minimum time that must pass in between time stamps that are kept.
   3. Set trim.laststamp to the minimum distance from the end of the recording (in seconds) that the last kept time stamp should be.
6. On line 46, check that ‘save\_data’ is set to 1. This will allow the generated data files to be saved to the data locations setup in the previous section. If you want to just check a trial’s variables without saving the resulting data files, set save\_data to 0.
7. Hit run, the script will run through the loop for each trial you requested to filter.
8. The new filtered time stamps are saved as both text files and BORIS formatted csv files automatically, for each behavior from the xlsx book.

The data is now ready for analysis with PhAT or custom MATLAB scripts.

**Test functionality on included sample data**

You can test that everything is working properly by running the two scripts on the sample data spreadsheet and saving the results to the ‘data’ folder – check that your run creates similar results to the included results for the same dataset in the ‘sample\_data’ folder.

*Test extract script:*

1. Open tdt\_Extract\_CAS\_v06.m, after you’ve made all of your necessary adjustments above. Set nameGenerator to the sample csv (‘nameGenerator\_sample\_v02.csv’), trials to [2:5 7:10 12:15 17 18 20 21], and resampf to 100.
2. Make sure the path\_to\_data.store location points to your data folder and not the sample\_data folder, otherwise your run will just overwrite the results already included in sample\_data.
3. Run the extract script.
   1. For each of the 12 recordings, check that results showed up in your data folders for Bruchas\_matfiles\_extracted, CAS\_matfiles\_extracted, figures\_rawtrace, PhAT\_matfiles, and PhAT\_stream\_csv.
   2. The extract script will only generate timestamps from TTL recorded inputs. In the sample data, only the airpuff recordings (trials 12:15) used a TTL, so you should see 4 files corresponding to the airpuff recordings in the PhAT\_stamp\_csv and timestamps folders.
4. Compare the data output files to those in the sample\_data, they should be the same.

*Test stamp filter script:*

1. Open PhAT\_StampFilter\_CAS\_v04.m, after you’ve made all of your necessary adjustments above. Set nameGenerator to the sample csv (‘nameGenerator\_sample\_v02.csv’).
2. First, we will test the ability to convert a manually generated text file of time stamps to BORIS format.
   1. In the sample\_data 🡪 timestamps folder, you will find 8 text files ending in ‘taillift.txt’ corresponding to the file names in the spreadsheet’s timestamp column for the 8 tail lift recordings (rows 2:5 and 7:10). Copy these 8 files to your data 🡪 timestamps folder so the stamp filter script can locate and read them.
   2. In PhAT\_StampFilter\_CAS\_v04.m, set trials = [2:5 7:10], input.phatmat = 0, input.txt = 1, input.beh = ‘taillift’, input.txt\_auto = 1, input.txt\_manual = 0, input.txtfile = [ ].
   3. These settings will cause the script to assume the input time stamp source is a text file instead of a PhAT mat file containing TTL generated time stamps. It will automatically read the names of the text file associated with each recording from the timestamps column of the spreadsheet.
   4. The trim settings are not important for this sample data, since the tail lifts were performed at user-determined intervals so there isn’t a possibility of any occurring too early/late in the session or at too close of an interval. It’s still fine to leave the ‘trim variables at their defaults (trim.minstart = 50, trim.interval = 10, and trim.laststamp = 30), we will just know that the imported time stamps already meet these specifications.
   5. Set trim.name = ‘\_filtered’, and make sure save\_data = 1. Run the script.
   6. There should now be 8 new text files ending in ‘filtered\_stamps.txt’ in the data 🡪 timestamps folder. As mentioned in 2d, these are identical to the original input text files since the original data already met the time stamp filtering parameters.
   7. Additionally, there should be 8 new csv files in the data 🡪 PhAT\_stamp\_csv folder, containing the filtered tail lift time stamps in BORIS format.
3. Next, we will test the ability to import TTL-generated stamps from a mat file (created during the extract step), filter them, and save them as both text and csv (BORIS) files.
   1. Set trials = [12:15], input.phatmat = 1, input.series = {‘TTL\_1’}, input.txt = 0, input.beh = [ ], input.txt\_auto = 0, input.txt\_manual = 0, input.txtfile = [ ].
   2. These settings will cause the script to assume the input time stamp source is the PhAT formatted mat file associated with the recording, containing TTL generated time stamps. Only a single TTL was used in these experiments marking the airpuff, so we are only filtering for TTL\_1.
   3. The TTL\_1 series marked each of the 15 air puffs animals received, but it also recorded a time stamp when the Arduino air puff program was started. We do not want this start time stamp included when looking at the signal around behavioral events, since there was not actually an air puff delivered for this first stamp.
   4. The first actual air puff does not occur until 5 minutes after starting the Arduino program. Set trim.minstart = 300 to remove any time stamps from the first 5 minutes of the recording. For this specific example the air puffs occurred 45-75s apart so there shouldn’t be any within 10s of each other, but it’s fine to keep trim.interval = 10. Similarly, a 5 min post air puff baseline was recorded so no air puffs should occur in the last 30s of the recording, but it’s fine to keep trim.laststamp = 30.
   5. Set trim.name = ‘\_filtered’, and make sure save\_data = 1. Run the script.
   6. In the data 🡪 timestamps folder, there should now be 4 new text files ending in ‘filtered\_airpuff\_timestamps.txt’, each with the first time stamp from the original text file filtered out.
   7. In the data 🡪 PhAT\_stamp\_csv folder, there should be 4 new csv files containing the filtered air puff time stamps in BORIS format.
4. Finally, we will test the ability to import TTL-generated stamps from a mat file and save them as state type variables in the BORIS format.
   1. Set trials = [17 18], input.phatmat = 1, input.series = {‘TTL\_1’ ‘TTL\_2’}, input.txt = 0, input.beh = [ ], input.txt\_auto = 0, input.txt\_manual = 0, input.txtfile = [ ].
   2. These settings will cause the script to assume the input time stamp source is the PhAT formatted mat file associated with the recording, containing TTL generated time stamps. Solenoid and lick TTLs were recorded for these trials.
   3. The TTL\_1 series marked each of the solenoid openings and TTL\_2 marked all licks, but both also recorded a time stamp when the Arduino sucrose program was started. We do not want this start time stamp included when looking at the signal around behavioral events, since there was not actually a solenoid opening or lick occurring during this first stamp.
   4. The first actual solenoid opening/lick access period does not occur until 5 minutes after starting the Arduino program. Set trim.minstart = 300 to remove any time stamps from the first 5 minutes of the recording. We want to group all lick and solenoid openings occurring within 10s of each other to a single lick or solenoid bout, so set trim.interval = 10. A 5 min post sucrose baseline was recorded so no licks or openings should occur in the last 30s of the recording, but it’s fine to keep trim.laststamp = 30.
   5. Set trim.name = ‘\_filtered’, and make sure save\_data = 1. Run the script.
   6. In the data 🡪 timestamps folder, there should now be 4 new text files ending in ‘filtered\_lick\_’ or ‘filtered\_solenoid\_’ timestamps.txt, each with the first time stamp from the original text file filtered out.
   7. In the data 🡪 PhAT\_stamp\_csv folder, there should be 2 new csv files containing both the filtered lick and solenoid time stamps in BORIS format. Make sure that for each event, both a Start and Stop time are recorded.

*Test linear track stamp filter script:*

1. In the sample\_data 🡪 lineartrack\_behevents folder, you’ll find the xlsx book ‘20231102\_behevents.xlsx.’ Copy this to your data 🡪 lineartrack\_behevents folder.
2. Open LinearTrack\_StampFilter\_CAS\_v01.m, after you’ve made all of your necessary adjustments above. Set nameGenerator to the sample csv (‘nameGenerator\_sample\_v02.csv’).
3. For the example data, you can set trim.minstart, trim.interval, and trim.laststamp all to [ ]. Set trim.name = ‘\_filtered’; and save\_data = 1;
4. Run the script. This should generate 2 BORIS formatted csv files in the data 🡪 PhAT\_stamp\_csv folder and 10 new text files of behavior start times in the data 🡪 timestamps folder.