Ephys pipeline overview

Folder structure

Inputs	**Info**
Y:\Data\TDTtanks\Monkey_phys	Raw TDT data
Y:\Data\Sortcodes\Monkey_phys	sorting related files
Y:\Data\Monkey	behavioral data
\Dropbox\DAG\phys\Monkey_phys_dpz\Electrode_depths.mat	electrode depths documentation
\Dropbox\DAG\phys\Monkey_phys_dpz\Same_cells.mat	same cells across blocks assignment
Outputs	**Info**
Y:\Data\Monkey_phys_mat_from_TDT	recorded data, arranged in the same trial structure as the behavioral data
Y:\Data\Monkey_phys_combined_monkeypsych_TDT	trial structures with combined behavioral and ephys data
Y:\Data\All_phys_preprocessing_log\Monkey_phys	automatically created log files of all preprocessing performed
\Dropbox\DAG\phys\Monkey_phys_dpz\plx_files.xls	(for multiple sortings:) used plx file documentation
\Dropbox\DAG\phys\Monkey_phys_dpz\Mon_sorted_neurons.xls	block/run/unit electrode locations, IDs, ratings, and procedures information

General Workflow

1. Fill in Electrode depth file (required as first step for WC pipeline)
2. WC: Create waveclus pre-clustering files (implemented in phys_gui)
3. WC: Run waveclus and sort for all channels, don't forget to save (wave_clus3new3, MATLAB 2014+)
    - Preferentially separate spike shapes in difficult cases
4. Create plxfiles (implemented in phys_gui)
5. Check and clean PLX file and **TAKE NOTES** (single/SNR/stability rating)
6. Now we need to assess which units are the same within and across blocks (based on channel, electrode depths, spike shapes and potentially tuning, carefully check notes for this). For that purpose you might want to do the following steps in any order (potentially even going back and forth):
    - Create combined (phys+behavior) files (implemented in phys_gui)
    - Assign same cells across blocks in Same cells file
    - Plot single unit tuning assuming cells are unique in every block (implemented in phys_gui) - This is meant to help assessing if two "units" in the same block might actually be the same (if they have very similar tuning properties)
    - Re-sort PLX file (combining same cells sort codes of same block same channel)
7. If you have finalized the sorting, create combined (phys+behavior) files (implemented in phys_gui) (yes, once again, now that you have a different amount of cells)
8. Complete Same cells file
9. Automatically update the automatic_sorting sheet from the Excel sorting table using phys_gui
10. Copy the automatic_soting entries to the final_sorting sheet of the Excel sorting table and

complete with manual entries.

Synchronization

- TDT data is stored as a continuous data (starting from start of recording)
- behavioral data is stored per trial (starting from first trial)

For combining both, we first convert TDT data to trial format (function TDT_trial_struct.m) preferentially using epocs store (Tnum and SVal) information.
Note: There is a flag for using the continous state information stream (stream_state_info), but it is permanently turned off, as the epoc information is already extracted from the continuous state information within the TDT circuit.
Importantly, trial start reference is STATE 2 (fixation acquisition). Trial end is STATE 2 of the next trial. Therefore, ITI between trial 1 and 2 as well as STATE 1 (initiation) of trial 2 are found at the end of trial 1.
This also means that data recorded before 1st trial's state 2 can not be stored in the trial structure and is saved in a separate variable (First_trial_INI) instead.

Associated code

Data conversion functions are located in Github External_modified/PLXTDT repository

- PLX2SPK
- PLX2TDT
- SEV2mat_working
- SPK2PLX
- TDT2PLX
- TDTbin2mat_working
- WC32SPK
- WC32SPK_concatenated
- WC32SPK_directly

The remaining code related to phys preprocessing (not including Waveclus and Plexon) can be found on Github Phys_preprocessing repository

- Core functions
    1. phys_gui_working.m
    2. phys_gui_execute.m
    3. TDT_trial_struct.m
    4. ph_combine_MP_and_TDT_data
    5. DAG_update_sorting_table.m
- Waveclus pipeline specific
    1. DAG_WC3_preprocessing.m
    2. DAG_parse_data_tdt.m
    3. DAG_SpikefilterChan.m
    4. WC32SPK_directly.m
    5. ph_readout_broadband_lag.m

- Plexon pipeline specific
    1. DAG_create_PLX.m
    2. DAG_update_plx_file_table.m
    3. ph_get_new_plx_extension.m
- Extra standalone functions
    1. DAG_derive_TDT_streamer_broadband_lag.m
    2. ph_debugging_GUI.m
    3. DAG_take_over_sortcode_PLX2PLX.m
- Only of historical relevance
    1. DAG_move_sorting_files.m Was used to transfer sortcodes to their current destination
    2. ph_derive_electrode_depth.m Was used to create the first electrode_depths file extracting from sorted_neurons table
    3. ph_derive_same_cells.m Was used to create the first same_cells file extracting from sorted_neurons table
    4. ph_simulate_history.m Was used to create the first log files