EPhys Matlab Toolbox

Electrophysiology Experiment Software for TDT

Designed & Written by Daniel Stolzberg, Ph.D.

daniel.stolzberg@gmail.com

# Table of Contents

[Table of Contents 2](#_Toc344906157)

[Introduction 3](#_Toc344906158)

[Hardware Recommendations, Software Requirements & Installation 4](#_Toc344906159)

[Hardware recommendations 4](#_Toc344906160)

[Required software 4](#_Toc344906161)

[Obtaining and Using SVN 4](#_Toc344906162)

[EPhys Toolbox Overview 6](#_Toc344906163)

[Creating a New Experiment 6](#_Toc344906164)

[Protocol Design Utility (ProtocolDesign) 7](#_Toc344906165)

[Creating a New Protocol 7](#_Toc344906166)

[Adding/Removing Modules 7](#_Toc344906167)

[Working With Parameter Tags 7](#_Toc344906168)

[Function column 8](#_Toc344906169)

[Values column 8](#_Toc344906170)

[Buddy Column 9](#_Toc344906171)

[Control Panel (EPhysController) 10](#_Toc344906172)

[Calibration Utility (CalibrationUtil) 11](#_Toc344906173)

[Tank Registration (TankReg) 12](#_Toc344906174)

# Introduction

The EPhys toolbox for Matlab integrates with Tucker-Davis Technologies (TDT) OpenEx software package and real-time processing hardware to facilitate the design and control of electrophysiology experiments. The toolbox includes a control panel which runs the experiment, a protocol design utility for generation parameters for an experiment, and a utility for the automated calibration of simple acoustic stimuli. Some familiarity with the Matlab environment and basic principles of RPvds circuit design are important for implementing the EPhys toolbox. Creating new experiment protocols, however, requires no programming by the user; however, advanced dynamic Matlab scripts can be written to create more complex experiments.

There were several motivations for the designing the EPhys toolbox to extend OpenEx’s capabilities. One motivation was to facilitate the design of complex multiparametric experiment protocols in a simple but powerful graphical user interface (GUI). This is accomplished with the Protocol Design Utility (p. 7).

A second motivation was to enable the use of high sampling rates (~200 kHz on some hardware) required for producing ultrasonic stimuli on the TDT real-time modules. The Control Panel (p. 9) accomplishes this by offloading stimulus triggers to the host computer. This approach, which has certain considerations discussed later, reduces the overhead (number of circuit components) required on to run an RPvds circuit which frees up additional processing power for circuit components required for various stimulation paradigms. An additional advantage of using the host computer to control stimulus triggering is the added capability for using custom Matlab scripts to dynamically select the next stimulus parameters based on previously acquired data (such as spike counts, field potential power, button presses, etc).

Thirdly, a Calibration Utility (p. 10) automates calibration of acoustic sources and creates standardized calibration files which can be used to normalize sound levels of parameters (such as stimulus frequencies).

# Hardware Recommendations, Software Requirements & Installation

## Hardware recommendations

The minimum requirements of EPhys toolbox are those of the installed version of Matlab and TDT software. The EPhys toolbox was developed on a standard quad-core processor personal computer with 4 GB of memory running Windows 7 64-bit operating system.

## Required software

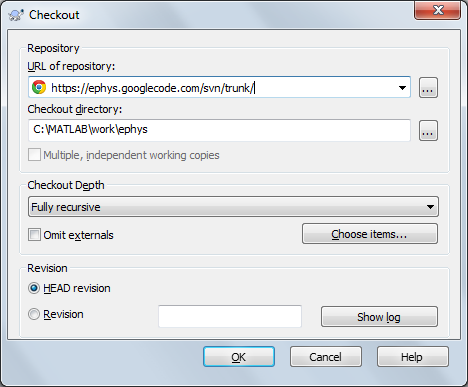
The EPhys toolbox was designed in Matlab version R2011a and higher. The EPhys toolbox does not require any additional toolboxes to function; however, EPhys uses the OpenDeveloper ActiveX controls which can be obtained from TDT. OpenDeveloper is designed to interface with electrophysiology data structures and the OpenEx software suite to run an experiment.

If installing Matlab for the first time, I find it best practice to install to a directory such as “C:\MATLAB\R2012b” and not to the default “Program Files” directory. Installing into the default “Program Files” directory may cause problems with permissions on Windows Vista and later.

Currently, the EPhys toolbox is distributed using a software versioning (svn) system hosted by GoogleCode and additional software will be required to obtain the latest releases of the EPhys toolbox software (see next section).

## Obtaining and Using SVN

The software versioning system (svn) requires third-party software to be installed locally on the computer which will be used with the EPhys toolbox. TortoiseSVN is a free software package and can be downloaded here: <http://tortoisesvn.net/downloads.html>. Download and install the latest TortoiseSVN software version for your operating system (32-bit or 64-bit) using default options. Once installed, follow these steps to obtain the EPhys software:

1. Create a new folder called “work” in the Matlab root directory such as “C:\MATLAB\work”
2. Open the work folder in a Windows Explorer, right-click in the white space to open a context menu, and click the “SVN Checkout…” menu item.
3. Enter the following information into the Checkout prompt and click OK.
   1. URL of repository: <https://ephys.googlecode.com/svn/trunk/>
   2. Checkout directory: C:\MATLAB\work\ephys
4. A new dialog will appear. Enter the username and password (contact Dan if you don’t have this information
5. A dialog should appear as the EPhys software is downloaded.

The versioning system allows those using the software to easily update files for updates and bug fixes by right-clicking the parent folder (such as: “C:\MATLAB\work\ephys”) and selecting “SVN Update” from the context menu. This system also provides the opportunity for distributed collaboration on future software developments.

A wiki page can be found at <https://code.google.com/p/ephys/wiki/Intro>. This wiki provides a place where issues can be raised, discussed, and addressed with software fixes.

# EPhys Toolbox Overview

The EPhys toolbox for Matlab is designed to integrate with the OpenEx software package provided by TDT. The OpenWorkbench program, included in the OpenEx package, should be used to define which TDT real-time hardware will be used during an experiment. Please refer to the TDT OpenWorkbench documentation for details on how to do this.

### Creating a New Experiment

The procedure for designing new experiments with EPhys begins with assigning labels and RPvds files to the TDT hardware modules using the Device Navigator of OpenWorkbench (TDT). While labels are arbitrary in reality, descriptive labels should be used in practice. One example would be to use the label **Stim** for a module (such as an RP2.1, RX6, RZ6, or other module) which will be used to control stimulation during the experiment, such as a speaker or light. If multiple stimulus modules are to be used in an experiment then names such as **Stim1** and **Stim2**, or **SpeakerStim** and **LightStim** can be used (). One or multiple data acquisition modules (such as an RX5, RZ5, or other module) can be assigned a label in a similar manner (**Acq**, for acquisition device, is used in ). Note that not all modules need to be populated. The labels assigned in the Device Navigator of OpenWorkbench will be used to direct parameters during protocol design using the EPhys toolbox.



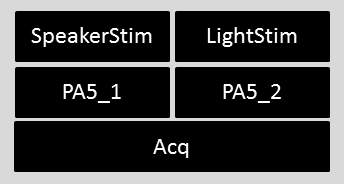


Figure . Unpopulated (top) and populated (bottom) device navigator

An RPvds file (with the extension .rcx) should be assigned to each real-time module which is defined in the Device Manager (see TDT documentation for details). The conventions for designing RPvds files for use with OpenWorkbench are outlined in the TDT documentation and should be followed. There are a few requirements for RPvds circuit designs when using EPhys to control experiments that will be discussed later in this manual (\*\*\*ADD\*\*\*).

# Protocol Design Utility (ProtocolDesign)

The Protocol Design Utility is a graphical user interface which is used to specify the values of parameters for an experiment. A simple example of this would be to specify all frequency and sound level combinations for a tone-evoked receptive field experiment. Instead of manually specifying every frequency/level combination, of which there may be hundreds or thousands of combinations, one can use the Protocol Design Utility to specify which frequencies and which sound levels should be presented and the utility permutes these parameters to create a list of stimuli for the experiment.

## Creating a New Protocol

Before creating a new protocol, an OpenWorkbench project should be created as descirbed above (see Creating a New Experiment on p. 6). The Protocol Design Utility can be launched in two ways. The command ProtocolDesign can be entered into the Matlab command window directly. Alternatively, clicking the “**P**” icon on the Control Panel (p. 9) toolbar will launch the utility.

### Adding/Removing Modules

Creating modules in OpenWorkbench was covered in an earlier section (see Creating a New Experiment, p 6). The labels for modules defined in OpenWorkbench can be added to or removed from the protocol design by clicking the “**+**” or “**-**“ buttons, respectively. Clicking the “**+**” button will launch a dialog box in which the user should enter the label (capitalization matters) of one module at a time.

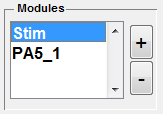


Figure Add/Remove module

### Working With Parameter Tags

Once a module has been added to the protocol, parameters can be specified using the table on the right of the Protocol Design Utility.

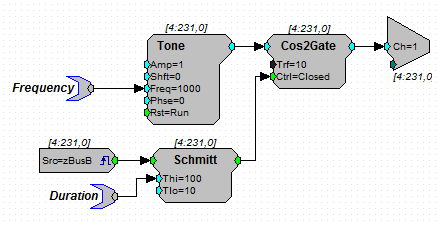


Figure Example RPvds circuit using parameter tags (ParTag component)

The RPvds circuit ot the right is a simple example of an RPvds circuit which plays tones of various frequencies and durations. The RPvds circuit to the right is one realization of such a design. The parameter tags (ParTag components) called “Frequency” and “Duration” are linked to the **Freq** parameter of the Tone generating component and the **Thi** parameter of the Schmitt logic component, respectively.

The “Frequency” and “Duration” parameter tags are entered (case sensitive) into the “Tag” column in the Protocol Design Utility parameter table. The other columns of the table are used to specify the values and how the parameter functions.

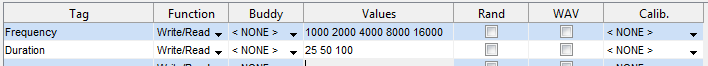


Figure Protocol Design Utility parameter table with example of parameter values controlling “Frequency” and “Duration” parameter tags (ParTag) specified in the RPvds circuit.

### Function column

The **Function** column can take one of three options selected from a drop-down list. This field indicates whether the parameter tag is updated before each trial (“Write”) or the value is read by EPhys Control Panel after each trial (“Read”). If both functions are required, then the “Write/Read’ option should be selected. While most parameters will simply be used to update the parameter tags on the RPvds circuit, the “Read” functionality is useful for dynamic scripts.

### Values column

The **Values** column is the primary focus of the parameter table. This field is used to specify the values that the parameter (specified in the **Tag** column of the same row) will have for a protocol. In the example parameter table above, the “Frequency” parameter tag is associated with the values: 1000 2000 4000 8000 16000 (separated by spaces). These values will be used to update the “Frequency” parameter tag in the RPvds circuit example above. The same goes for the “Duration” parameter tag for the values: 25 50 100 (specified in milliseconds according to the Schmitt component requirements).

Instead of manually specifying a list of variables, the **Values** column will accept any Matlab code which evaluates to a scalar or vector. Any code which runs as a line item, including function calls, in the Matlab command window will work in this field. For example, instead of specifying the values manually as in Figure 4, we could have also entered: 2000\*2.^(0:3) with the same result. Another example of valid input: logspace(log10(1000),log10(42000),40)

Clicking the **View Compiled** button (located below the table) brings up a new window with all permutations of specified parameter (Figure 5). In this Compiled Protocol table, each row starts with a trial number with each subsequent column a parameter value. Note that the total number of trials displayed in the table will be truncated if over 2000 trials are generated. The figure caption will indicate the total number of trials.

### Buddy column

The **Buddy** column of the parameter table is used to have the values of more than one parameter co-vary. In other words, making a buddy variable will make the values of two or more parameters to always be the same. One stipulation is that the parameters with a selected buddy variable must have the same number of values. Multiple buddy variables can be specified.

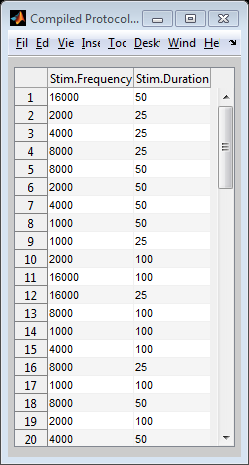


Figure Example of a compiled protocol

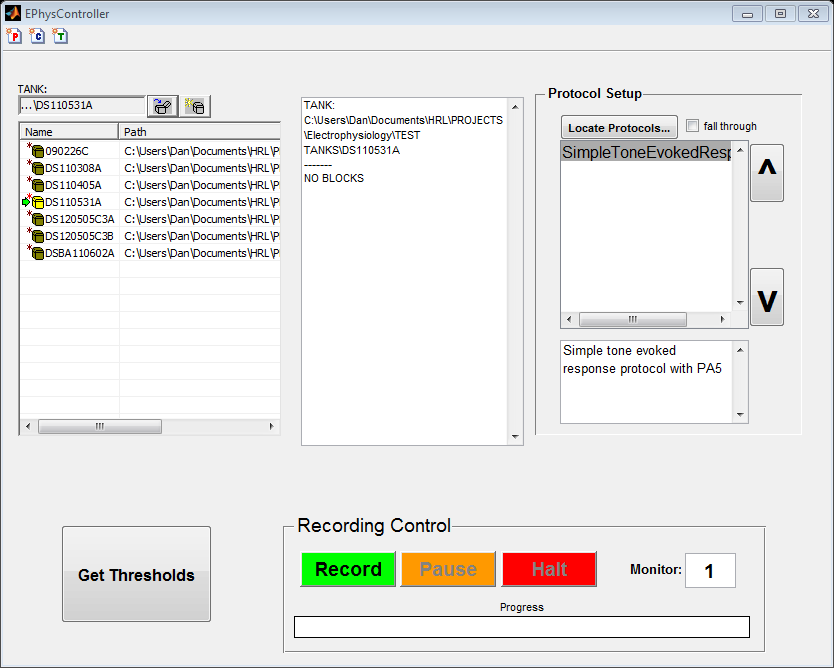
### Rand column

When activated, the **Rand** field works in conjunction with the **Values** field to provide a random value, selected at run time, between two values specified. For example, if a parameter tag is to be updated with a random value between 50 and 200, then enter 50 200 in the **Values** field and check the **Rand** field. Before the parameter is updated during the experiment, a random value will be selected from a uniform distribution (using the Matlab rand function) with the lowest possible value of 50 and a highest possible value of 200. This behavior is indicated on the compiled table (click View Compiled) as the two scalar values [50 200].

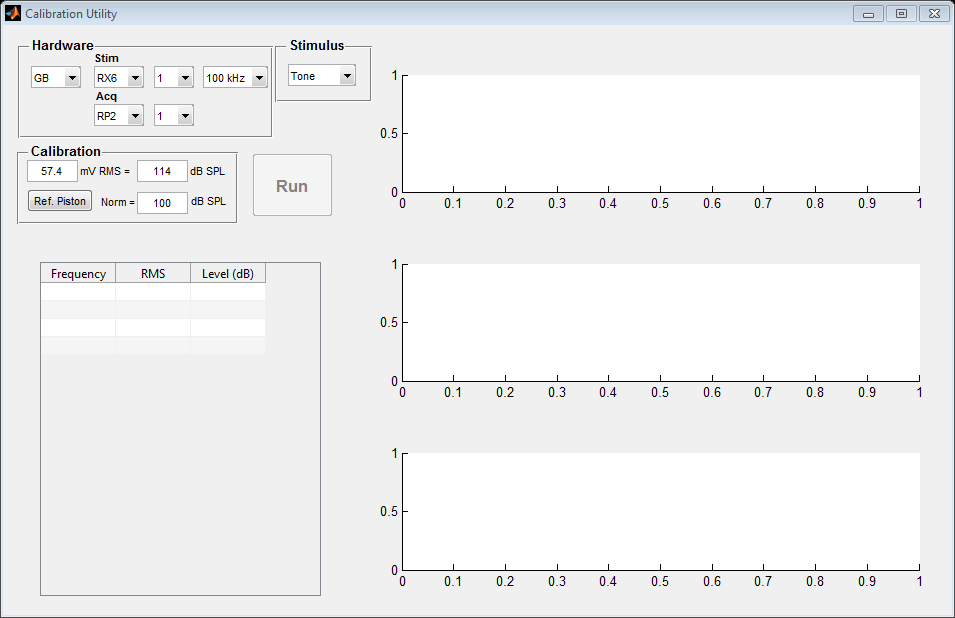
### WAV column

Activating the **WAV** field will launch a new window. Use this window to add WAV format files to a list. The file names, sampling rate, and duration of each WAV file will be displayed in the table in the new window. The order in which the WAV files are listed in the table are the order they will be specified for the parameter for which the WAV field was clicked. Please note that the sampling rate of the WAV file should be the same as the TDT real-time module will be running. If this is not followed, then aliasing will likely occur during playback.

# Control Panel (EPhysController)



# Calibration Utility (CalibrationUtil)



# Tank Registration (TankReg)

