EPhys Matlab Toolbox

Electrophysiology Experiment Software for TDT

Designed & Written by Daniel Stolzberg, Ph.D.

daniel.stolzberg@gmail.com

last updated on 4/22/2013

Copyright © by Daniel Stolzberg, Ph.D. 2013

Table of Contents

[List of Figures 3](#_Toc352334696)

[Introduction 4](#_Toc352334697)

[Hardware Recommendations, Software Requirements & Installation 5](#_Toc352334698)

[Hardware recommendations 5](#_Toc352334699)

[Required software 5](#_Toc352334700)

[Obtaining and Using SVN 5](#_Toc352334701)

[Adding EPhys software to the MATLAB path 6](#_Toc352334702)

[EPhys Toolbox Overview 7](#_Toc352334703)

[Creating a New Experiment 7](#_Toc352334704)

[RPvds Requirements 7](#_Toc352334705)

[Protocol Design Utility (ProtocolDesign) 9](#_Toc352334706)

[Creating a New Protocol 9](#_Toc352334707)

[Adding/Removing Modules 9](#_Toc352334708)

[Working With Parameter Tags 9](#_Toc352334709)

[The $ flag 10](#_Toc352334710)

[Function column 10](#_Toc352334711)

[Values column 10](#_Toc352334712)

[Buddy column 11](#_Toc352334713)

[Rand column 11](#_Toc352334714)

[WAV column 11](#_Toc352334715)

[Calib. (calibration) column 12](#_Toc352334716)

[Protocol Design Options 13](#_Toc352334717)

[EPhys Control Panel (EPhysController) 14](#_Toc352334718)

[Calibration Utility (CalibrationUtil) 16](#_Toc352334719)

[Tank Registration (TankReg) 17](#_Toc352334720)

[Database Browser (DB\_Browser) 18](#_Toc352334721)

# List of Figures

[Figure 1. SVN Checkout 6](file:///C:\MATLAB\work\ephys\EPhys_Manual.docx#_Toc352334605)

[Figure 2. Unpopulated (top) and populated (bottom) device navigator 7](file:///C:\MATLAB\work\ephys\EPhys_Manual.docx#_Toc352334606)

[Figure 3 Required RPvds Components 8](file:///C:\MATLAB\work\ephys\EPhys_Manual.docx#_Toc352334607)

[Figure 4 Add/Remove module 9](file:///C:\MATLAB\work\ephys\EPhys_Manual.docx#_Toc352334608)

[Figure 5 Example RPvds circuit using parameter tags (ParTag component) 9](file:///C:\MATLAB\work\ephys\EPhys_Manual.docx#_Toc352334609)

[Figure 6 Protocol Design Utility parameter table with example of parameter values controlling “Frequency” and “Duration” parameter tags (ParTag) specified in the RPvds circuit. 10](file:///C:\MATLAB\work\ephys\EPhys_Manual.docx#_Toc352334610)

[Figure 7 Example of a compiled protocol 11](file:///C:\MATLAB\work\ephys\EPhys_Manual.docx#_Toc352334611)

[Figure 8 Protocol Design options 13](file:///C:\MATLAB\work\ephys\EPhys_Manual.docx#_Toc352334612)

[Figure 9 EPhys Control Panel 15](file:///C:\MATLAB\work\ephys\EPhys_Manual.docx#_Toc352334613)

# 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 (ProtocolDesign) (p. 9).

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 **Error! No bookmark name given.** (p. 10) 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 (CalibrationUtil) (p. 16) automates calibration of acoustic sources and creates standardized calibration files which can be used to normalize sound levels of parameters (such as stimulus frequencies).

Finally, the use ofo high-channel electrode arrays has dramatically increased the size and complexity of electrophysiological datasets. Efficiently maintining, managing, and accessing these very large datasets is greatly facilitated by the use of a common database structure. To address this, the EPhys toolbox includes GUIs which can be used to upload acquired data and stimulus parameters to an SQL server as well as GUIs to navigate, access, and visualize stimulus-evoked responses.

# 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.

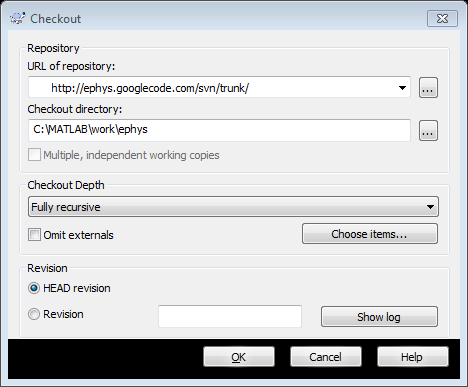
If installing Matlab for the first time, it is recommended to Matlab install to a directory such as “C:\MATLAB\R2012b” (or whatever the current version) and not to the default “Program Files” directory. Installing into the default “Program Files” directory may cause problems with permissions when updating with TortoiseSVN 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.

Figure 1. SVN Checkout

* 1. URL of repository: [http://ephys.googlecode.com/svn/trunk/](https://code.google.com/p/ephys/source/checkout)
  2. Checkout directory: C:\MATLAB\work\ephys

1. 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. This requires a Google account.

## Adding EPhys software to the MATLAB path

To make sure MATLAB recognizes the software, open MATLAB, type pathtool in the command window, and add the directory of the ephys svn checkout (suggested, C:\MATLAB\work\ephys). Click the Save button and close the path tool dialog. Finally, type the command ephys\_startup in the command window and appropriate paths will automatically be added to the path.

# 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 (**Error! Reference source not found.**). 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 Figure 2). 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.



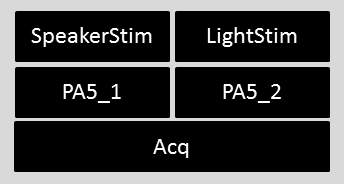


Figure 2. Unpopulated (top) and populated (bottom) device navigator

NOTE: Programmable Attenuator PA5 modules should be labeled PA5 followed by a suffix if desired as is in the example in Figure 2.

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 next.

### RPvds Requirements

For the majority of expeirment designs, the EPhysController (disucssed on page 14) controls parameter values and timing of stimuluation of the TDT hardware modules (defined in the OpenWorkbench Device Navigator; see page 7). EPhysController first updates all parameter tags on real-time modules as well as any PA5 programmable attenuators, and then uses the **zBusB** trigger issue a synchronised trigger pulse across all real-time modules. In order to accomplish this from Matlab the custom scripts named ZBUSB\_ON and ZBUSB\_OFF (see Figure 3) must be included on one of the RPvds modules in use.

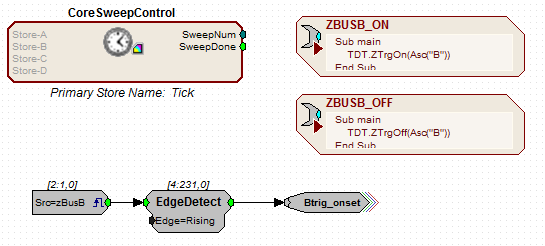


Figure 3 Required RPvds Components

The **CoreSweepControl** macro (left component in Figure 3) handles module synchronization and is required to be included in every RPvds file. This macro is available in RPvds by clicking the Components menu and selecting Circuit Macros.

Note: If using multiple RPvds circuits, each CoreSweepControl macro must have a unique primary store name. This can be modified by double clicking the macro object in RPvds, clicking the Setup tab, and then Change button. Any four letter name can be used (eg, Tock).

*Note: When using high sampling rates on RX generation devices, the CoreSweepControl macro utilizes too much of the processing capabilites. Modified versions of this macro are included in the “ephys\examples\useful macros” directory and may be used to resolve this issue.*

# 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. 7). 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 **Error! No bookmark name given.** (p. 10) toolbar will launch the utility.

### Adding/Removing Modules

Creating modules in OpenWorkbench was covered in an earlier section (see Creating a New Experiment, p 7). 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.

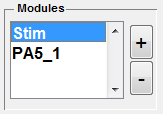


Figure 4 Add/Remove module

### Working With Parameter Tags

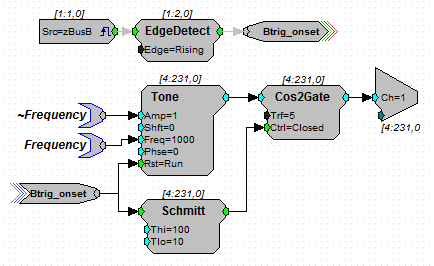


Figure 5 Example RPvds circuit using parameter tags (ParTag component)

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

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 tag is discussed on page 12).

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.

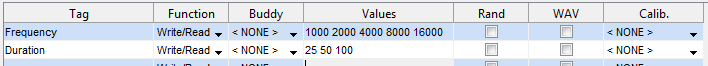


Figure 6 Protocol Design Utility parameter table with example of parameter values controlling “Frequency” and “Duration” parameter tags (ParTag) specified in the RPvds circuit.

### The $ flag

Some experiment designs require the values of parameters to be specified when beginning the recording. This can be done by prepending a $ (dollar sign) to a parameter tag name (a $ should also prepend the parameter tag name in the corresponding RPvds file); for example, $Frequency or $Duration. The compile at runtime option must be checked to enable this functionality. A prompt will automatically appear for the user to enter a value for the parameters with the $ flag when clicking the **Record** button on the EPhysController GUI (see page 14Figure 9) to begin an experiment.

### 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 numeric 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 6, we could have also entered: 2000\*2.^(0:3) with the same result. Another example of valid input: logspace(log10(1000),log10(42000),40).

When using the $ flag with a parameter tag, the value entered will be the default values when starting an experiment (see page 10 for details).

Clicking the **View Compiled** button (located below the table) brings up a new window with all permutations of specified parameter (Figure 7). 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.

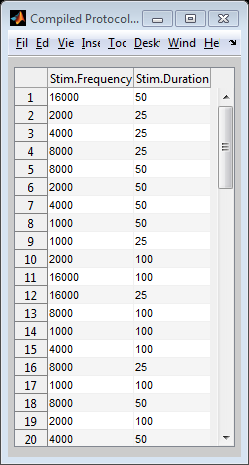


Figure 7 Example of a compiled protocol

### 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.

### 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 occur during playback resulting in unintended stimulus artifacts.

### Calib. (calibration) column

The **Calib.** field is used to select a calibration file to be associated with a parameter tag. Calibration files are generated using the Calibration Utility (CalibrationUtil) (p. 16). Clicking the drop-down box in this column will launch a Windows dialog requesting the user to locate a calibration file (\*.cal) which was created using the CalibrationUtility. Select the calibration file to be associated with the parameter in the same row. When compiled (after saving or clicking the ‘View Compiled’ button), an additional parameter will be automatically generated which will contain the calibrated value. This associated parameter will have the same name as the parameter but will be prefixed with a tilde (~).

For example: The circuit in Figure 5 (page 9) demonstrates a basic gated tone function. The Frequency parameter tag is linked to the **Freq** input of the Tone component and a second parameter labeled ~Frequency is linked to the **Amp** input of the Tone component. This associated parameter should not be explicitly added to the parameter table in the Protocol Design Utility, but will be automatically generated when adding a calibration to the Frequency parameter. The values generated for the ~Frequency parameter will be selected or calculated from the normalization curve in the calibration file. The result of this will be that each tone frequency presented during an experiment will be generated at a voltage which ould produce a sound level (for example, 80 dB SPL) set during the calibraiton procedure. A PA5 programmable attenuator can then be used to attenuate the voltage to the intended sound level during an experiment.

Note: Values which are not explicitly calibrated are calculated from the the voltage normalization curve using a piecewise cubic hermite interpolating polynomial (the PCHIP function in Matlab).

A calibration file can be used in conjunction with parameters using the $ flag in the same manner as with the normal parameters as described above. For example, $Frequency would be associated with the calibrated ~$Frequency parameter in RPvds.

## Protocol Design Options

A few options (Figure 8) go along with each protocol file which control presentation of trials. The **Randomize** option will reorder the sequence of trials in a pseudorandom fashion.

By default the order of trials is compiled (meaning the actual trial sequence is generated) when the protocol file is saved. This will ensure that the same sequence of trials (even if randomized order) will be presented each time the experiment is run. This behavior can be altered by enabling the **Compile at Run-Time** option. When enabled, the protocol is compiled (the trial sequence generated) each time the experiment is run. This has some advantages if using a custom script.

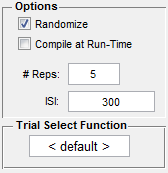


Figure 8 Protocol Design options

The **# Reps** field accepts a scalar value which indicates how many times to present each combination of parameters. A value of 5 would repeat all combinations of parameters 5 times.

The inter-trigger interval (IT**I)** field accepts either a single scalar value or two scalar values separated by a space (eg. 200 500). If a single value is specified, then a constant time interval will be used between stimulus triggers. If two values are entered, then a random time interval will be used between trials within the specified range. The field is in milliseconds.

Finally, the name of a custom written funtion can be specified in the **Trial Select Function**. This function supplants and should extend the default trial selection functionality. This is an advanced topic which should be tested thoroughly before running an experiment. If a custom function is specified then it must be somewhere on the Matlab path.

# EPhys Control Panel (EPhysController)

The EPhys Control Panel controls physiology experiments in conjunction with OpenEx software. To launch the program, enter EPhysController into the Matlab command window and hit enter. To begin an experiment:

1. OpenProject must be open on the same computer and the OpenWorkbench Device Manager (see on page 7) should be set for the intended experiment.
2. Create a new tank or select an existing tank using the left panel of the EPhys GUI.
3. Click the **Locate Protocols…** button and select a directory containing one or multiple protocol files.
4. Click **Record** to begin an experiment.
5. Progress of the experiment will be displayed at the bottom of the EPhysController and the software will automatically stop after finishing and reset for the next protocol.

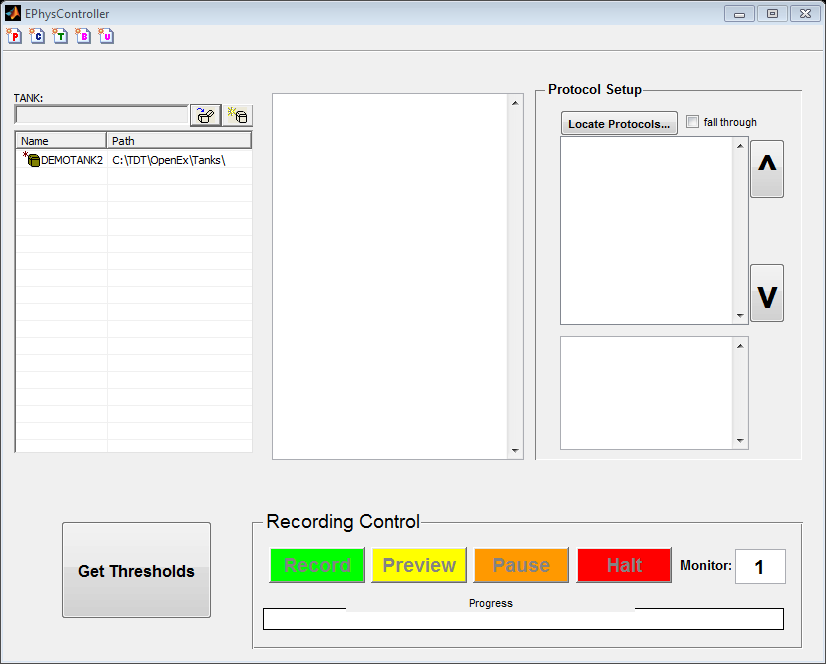
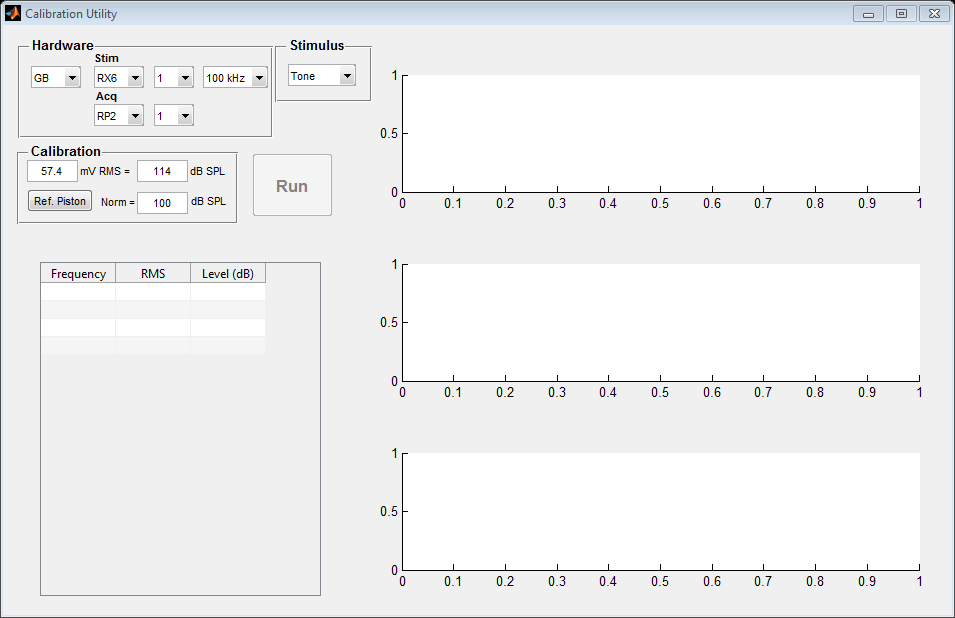
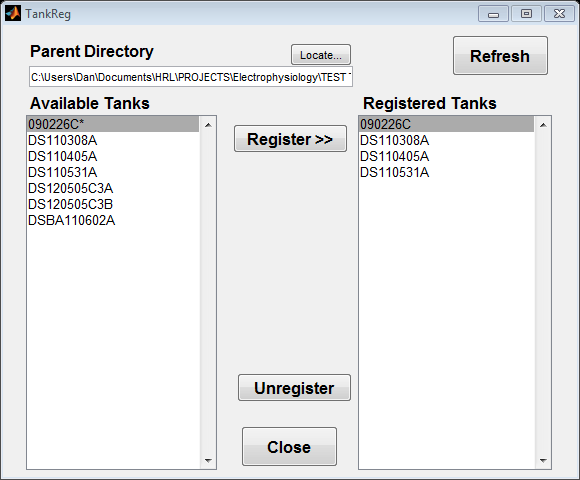


Figure 9 EPhys Control Panel

# Calibration Utility (CalibrationUtil)



# Tank Registration (TankReg)



# Database Browser (DB\_Browser)

