

# **SykofizX Manual**

---

***DT***

SykofizX Manual – Version 2.0

**Copyright**

© 2003 Tucker-Davis Technologies, Inc. (TDT). All rights reserved.

No part of this manual may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying and recording, for any purpose without the express written permission of TDT.

**Licenses and Trademarks**

Microsoft Excel, Windows, Windows 98, Windows NT, Windows 2000, Windows ME, Windows Vista, and Windows XP are registered trademarks of Microsoft Corporation.

# Table of Contents

<b>Before You Begin</b> .....	<b>7</b>
<i>System Requirements</i> .....	7
<i>Setting Up Your System</i> .....	7
<b>Introduction</b> .....	<b>9</b>
<i>Data Collection</i> .....	10
<i>Experimental Design</i> .....	11
<i>System Calibration</i> .....	13
<i>SykofizX Architecture</i> .....	14
<i>Data Analysis</i> .....	15
<i>Associated File Types</i> .....	16
<b>User's Guide</b> .....	<b>17</b>
<i>Data Collection Mode</i> .....	17
Selecting an Existing Data File.....	18
Creating a New Data File.....	18
Evaluation Utilities.....	19
Data Collection Window.....	23
Data Collection Toolbar.....	27
Data Storage.....	29
Remote Data Collection.....	29
<i>Calibration Mode</i> .....	29
Plugin Selection.....	29
Stimulus Configuration.....	30
Microphone and Sound Level Meter Specification.....	31
Hardware I/O Configuration.....	33
Calibration Data Collection.....	33
Calibration Data Storage.....	34
Use of Calibration Data in Stimulus Generation.....	34
<i>Experimental Design Mode</i> .....	35
Selecting an Existing Design File.....	36
Creating a New Design File.....	36
Experiment Design Wizard Navigation.....	37
General Experiment Configuration.....	37
Stimulus Generation Configuration.....	38
Presentation Paradigm.....	41

Subject Interface Configuration .....	44
Stimulus Timing Configuration .....	47
Response Timing Configuration .....	48
Define Experimental Variables .....	49
Independent Variable Configuration.....	51
Practice Trials Configuration .....	53
Additional Values to Store in Data Files .....	54
Condition Grid .....	55
Saving Design Files.....	56
Evaluation Utilities .....	56
<i>Custom Scripting</i> .....	57
Saving Custom Scripts .....	57
Stimulus Generation Scripts.....	58
Subject Interface Scripts .....	63
<i>Standard Plugins</i> .....	69
Stimulus Generation.....	71
Presentation Paradigm.....	78
Subject Interface .....	84
Independent Variable Configuration.....	90
Calibration .....	97
<i>Data Analysis</i> .....	99
Loading a Subject Data File .....	99
Navigating a Subject Data File .....	100
Exporting Data from a Subject Data File.....	101
Selecting a File Type.....	102
<b>Sample Designs</b> .....	<b>105</b>
<i>Hardware Settings</i> .....	105
<i>System 3</i> .....	106
Example 1 - Amplitude Modulation Detection Using a Forced Choice Paradigm and Keyboard-Video-Mouse.....	106
Example 2 - Amplitude Modulation Detection Using a Forced Choice Paradigm and Lowpass Filtering.....	107
Example 3 - Amplitude Modulation Detection Using a Forced Choice Paradigm and Response Box .....	108
Example 4 - Amplitude Modulation Detection Using a Same-Different Paradigm and Response Box .....	109
Example 5 - Amplitude Modulation Detection Using an RCO File .....	110
Example 6 - Amplitude Modulation Detection Using a Same-Different Paradigm and Keyboard-Video-Mouse.....	111
Example 7 - Amplitude Modulation Detection Using a Forced Choice Paradigm and a Reminder Interval .....	112

Example 8 - Amplitude Modulation Detection Using a Yes-No Paradigm and Keyboard-Video-Mouse .....	113
Example 9 - Word Recognition Task with Rhyming Foils .....	114
Example 10 - Vowel Identification .....	116
Example 11 - Classification Experiment - Noun or Verb Recognition Task.....	117
Example 12 - Tone in Noise Detection.....	119
Example 13 - Just Noticeable Difference Test.....	120
Example 14 - Loudness Matching.....	121
Example 15 - Magnitude Estimation .....	122
Example 16 - Amplitude Modulation Detection Using a Yes-No Paradigm and a Secondary Open Response .....	123
<i>System II</i> .....	124
Example 1 - Amplitude Modulation Detection Using a Forced Choice Paradigm and Keyboard-Video-Mouse.....	124
Example 2 - Amplitude Modulation Detection Using a Forced Choice Paradigm and Lowpass Filtering.....	125
Example 3 - Amplitude Modulation Detection Using a Forced Choice Paradigm and Response Box .....	126
Example 4 - Amplitude Modulation Detection Using a Same-Different Paradigm and Response Box .....	127
Example 6 - Amplitude Modulation Detection Using a Same-Different Paradigm and Keyboard-Video-Mouse.....	128
Example 7 - Amplitude Modulation Detection Using a Forced Choice Paradigm and a Reminder Interval .....	129
Example 8 - Amplitude Modulation Detection Using a Yes-No Paradigm and Keyboard-Video-Mouse .....	130
Example 9 - Word Recognition Test .....	131
Example 10 - Vowel Identification .....	133
Example 11 - Classification Experiment.....	134
Example 12 - Tone in Noise Detection.....	136

~

# Before You Begin

## *System Requirements*

### **Computer:**

- Operating System: Microsoft Windows 98, NT, 2000, ME, XP
- CPU: Pentium II processor or higher
- RAM: 64 MB
- Disk Space: 15 MB

### **Software:**

- Microsoft Scripting engine
- Microsoft XML support (included with Microsoft Internet Explorer 5.5 and later)
- TDT System II and/or System 3 drivers
- TDT ActiveX controls (for use with System 3 hardware)

### **Signal Generation Hardware/Software:**

- TDT System II or System 3 hardware
- TDT System 3 Driver version 44 or higher

### **Subject Interface Hardware:**

- Keyboard/Video/Mouse (host PC, remote PC, or via KVM extender)  
Or
- External I/O devices (button box, switch, potentiometer, lights, feeders, etc.)

## *Setting Up Your System*

### **Install Software**

If you haven't done so, install the most current versions of each of the following on your PC: TDT Drivers, RPDvsEx, ActiveX, and SykofizX. Instructions for installing the TDT Drivers software and setting up your system can be found in the TDT System 3 Installation Guide document.

The SykofizX application, by default, is stored in the **C:\TDT\SykofizX** directory.

### **Test Communication**

Test to ensure that your PC is correctly communicating with your hardware using the zBUSmon program (found in the C:\TDT\zDrv3 directory, for default installations). The zBUSmon display should show a map of your TDT System 3 devices, including your programmable hardware. Check the **Show Version** check box to ensure that the version displayed for your hardware matches the currently installed TDT driver version. If not, reprogram the device microcode following the instructions found in the System 3 Installation Guide provided with your system.

### **Launching SykofizX**

SykofizX can be launched from the MS Windows Start Menu, a desktop icon, or by double-clicking the SykofizX.exe file. SykofizX will automatically recognize the available programmable TDT hardware devices.

**Important!:** Before running your experiment or any of the sample designs, ensure that your hardware is properly configured. See Hardware Settings, page 105 for more information.



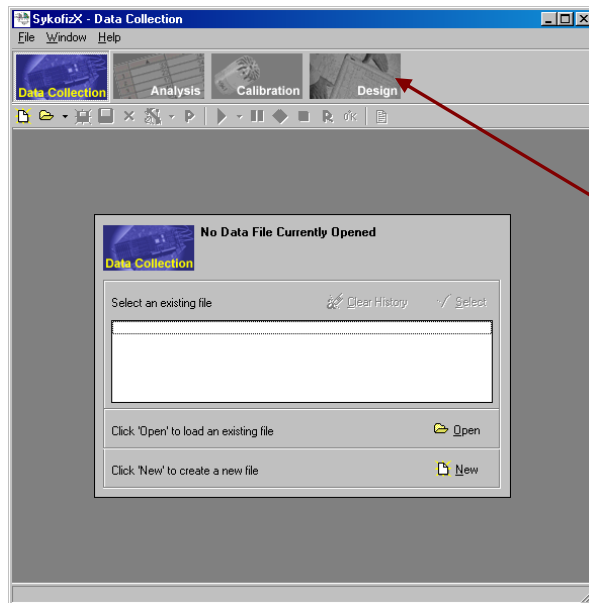


# Introduction

SykofizX is a comprehensive software application for designing and conducting psychophysical experiments in audition. This application was designed to meet the specific needs of scientists conducting experiments in speech perception and psychoacoustics. SykofizX was written for a wide variety of users, ranging from those who prefer a "turn-key" application to those who desire complete control over any aspect of the experimental process. SykofizX provides tools for customizing entire experimental designs from beginning to end, including a wide array of available psychophysical methods, extensive flexibility in signal generation, multi-modal subject interface, complete data acquisition control, data analysis and export, and system calibration. Standard with the SykofizX application are flexible subject interface modules suitable for testing of infant and adult human listeners as well as animal subjects.

SykofizX is uniquely extensible via an array of available features including:

- Support for TDT System II and System 3 hardware as well as APOS emulation with System 3 hardware
- Modular architecture
- Interchangeable standard plugins
- Vast libraries of standard and customizable scripts
- Flexible data export for off-line analysis
- Support for interleaving conditions during data collection
- Full network compatibility for remote access and control
- Powerful XML data storage format



The SykofizX application is separated into four functional units:

- Data Collection
- Data Analysis
- Calibration
- Design

A toolbar provides access to the features associated with each unit.

The plugin architecture allows the user to modify existing experimental designs to create novel designs with minimal effort.

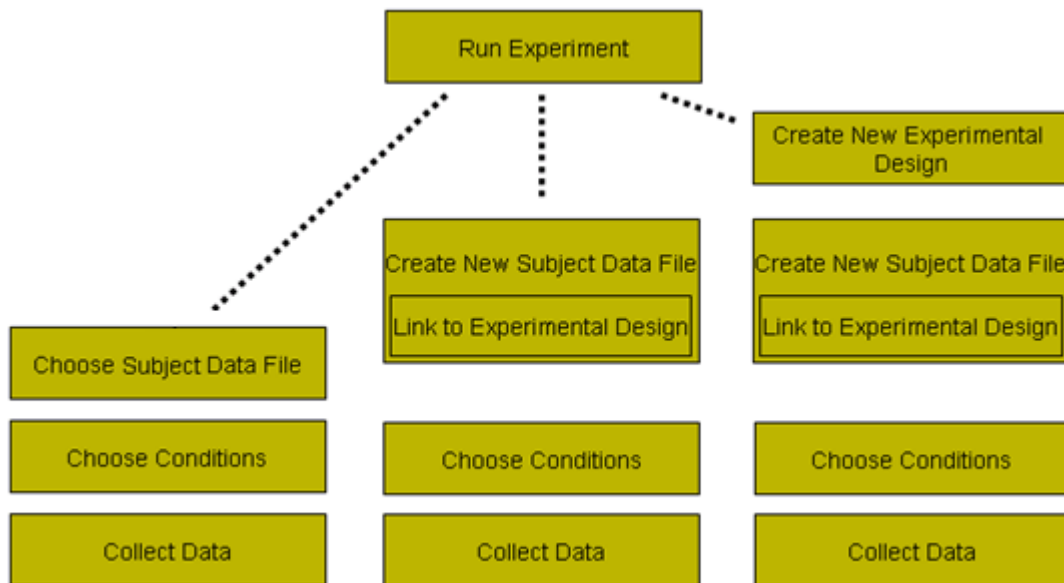
Notable features include:

- Running multiple experiments simultaneously using independent TDT hardware setups
- Simultaneous use of Data Collection, Analysis, and Design modes

- Fully integrated and easy-to-use calibration mode
- Flexible subject interface configuration suitable for psychophysics and speech perception research with human adult, infant, and animal subjects
- Multiple presentation and independent variable manipulation methods
- Multiple data display features from Data Collection and Analysis modes
- Convenient evaluation utilities available from Data Collection and Design modes for verification, demonstration, and subject training
- Running conditions under random, sequential, or custom order during data collection
- Complete control over stimulus and subject response timing
- Support for .wav and other stimulus file formats
- Support for interleaved tracking

## Data Collection

To run an experiment, one need only open a subject data file, choose conditions to be run from a list, and collect data! The subject data file contains a complete copy of the experimental design, any runtime parameters, and the data for each experimental condition. As data is collected, the subject data file expands. This format allows the subject data file to be completely portable and ensures that a record of the specific experimental parameters is stored with the data. Furthermore, this comprehensive subject data file provides the necessary trial-by-trial information to enable interleaved tracking as well as pause and resume features. The diagram below outlines the data collection process.



Once an experimental design has been developed, and a subject data file has been created, the user may initiate data collection. In data collection mode, the user also has a number of Evaluation Utilities to facilitate setup and stimulus verification as well as subject instruction, training, and practice.

SykofizX supports data collection from single or multiple subjects, each with their own experimental design. The application can control multiple, independent hardware systems via network interface, allowing the experimenter to mix experimental designs, TDT System II and System 3 hardware, subject interface configurations, and subject interface software routines.

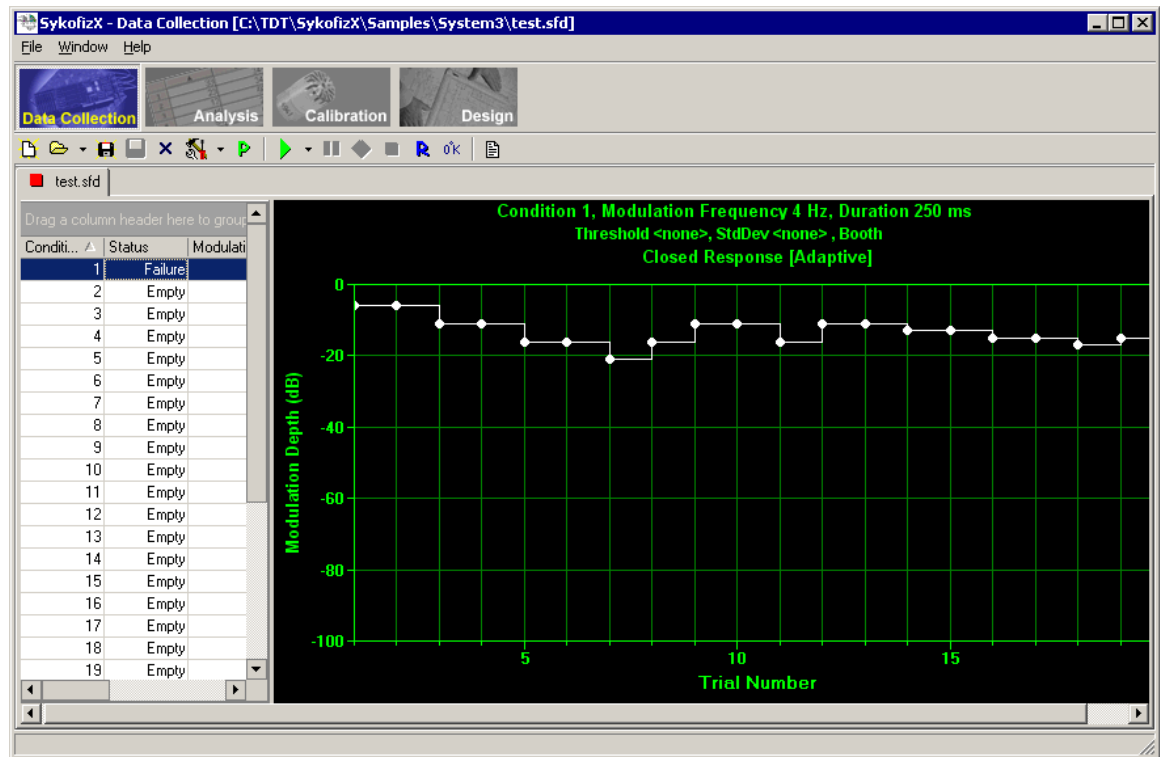
While highly unlikely, one of the many possible uses of the SykofizX application deployed on a single computer is to simultaneously run a human adult subject on a speech-perception experiment using System 3 hardware, an infant adult on a pitch perception experiment using System II hardware, and an animal subject on a localization experiment using System 3 hardware and the array processor emulator!

Immediately prior to data collection, a variety of options are available to the experimenter for:

- recording subject or experimenter supplied information
- previewing stimuli
- previewing stimulus presentation options
- previewing subject interface functions
- listener training (shaping) via sample or practice trials under either manual or automated control

The SykofizX application displays the data collection parameters for each subject on a different tab in the control window, as shown in the figure below. Data collection for each listener can be initiated, paused, and/or halted separately or simultaneously. Each tab contains a grid listing all of the experimental conditions for a given experimental design, and any condition-specific parameters. The experimenter may select one or more of these conditions for the current test session. Multiple conditions may be run in sequential order, random order, or a user-defined order.

The experimenter may view a custom dynamic display of current data values in graphical or tabular format depending on the nature of the data being collected.



## Experimental Design

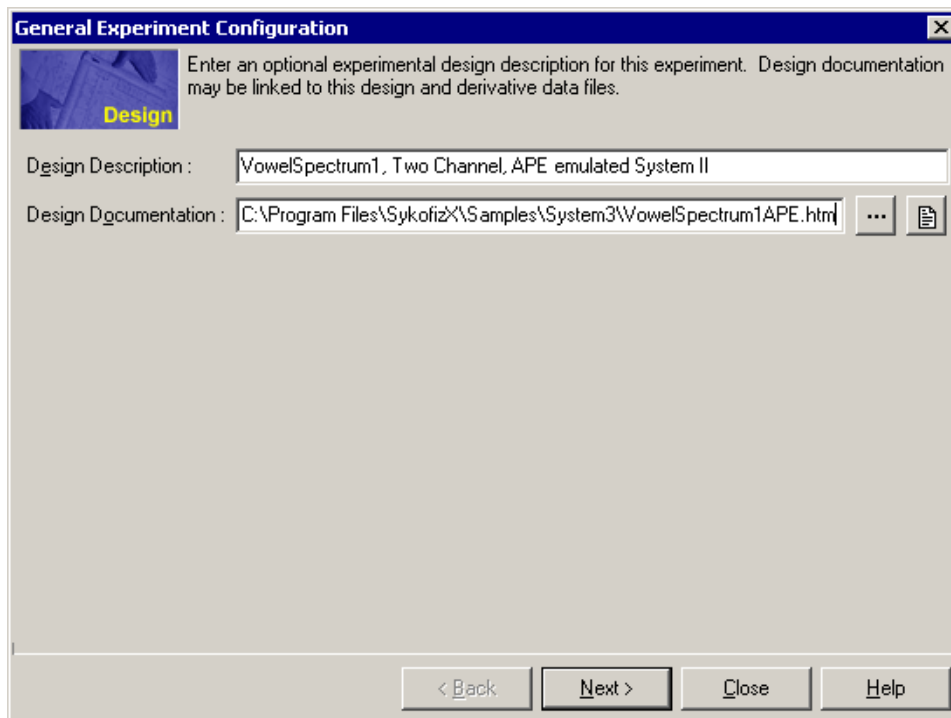
The design of a new experiment involves configuration of each aspect of the desired psychophysical methods, including stimulus specification and the selection and configuration of options related to the presentation paradigm, stimulus timing, as well as subject interface and response collection.

Historically the two most common approaches to experimental design have been to either set up an experiment using the limited functionality of a turn-key application or to program your own custom application. SykofizX eliminates the need for extensive programming yet maximizes flexibility. SykofizX achieves its unique combination of power, flexibility, and ease of use by decomposing psychophysical experiments into several fundamental elements common to all psychophysical procedures. The end user can then mix and match various functional units, with varying degrees of customization, to easily create novel experiments.

The ten functional units of a psychophysical experiment include:

- General Experiment Configuration
- Stimulus Generation Configuration
- Presentation Paradigm
- Subject Interface Configuration
- Stimulus Timing Configuration
- Response Timing Configuration
- Define Experimental Variables
- Independent Variable Configuration
- Practice Trials Configuration
- Additional Values to Store in Data Files

The creation of a new experimental design involves specification of parameters within each of the functional units. SykofizX simplifies this process by walking the experimenter through a series of intelligent design wizards (example shown below) that make available all options that are compatible with the specifications configured on previous wizards. Design wizards have extensive validation logic that helps users configure experiments quickly and accurately.



Once configured, the experimental design may be saved and used to create new subject data files for data collection and storage. Existing experimental designs may be loaded, re-configured, and saved as new designs for subsequent experiments.

## System Calibration

The calibration portion of the application functions as a second, simpler "experimental" design that interacts with the application and associated hardware to collect and save acoustic calibration data. In calibration mode, the application again walks the user through a series of intelligent wizards that ask the user to specify the stimulus generation hardware (e.g., System 3), the type of calibration to be conducted, the transducer(s) to be calibrated, the microphone(s) to be used, and the method of determining reference voltage (e.g., piston phone or manufacturer specification). The calibration mode supports manual and automatic calibration data collection.

**Stimulus Configuration**

The next step in defining a calibration is to specify the type of stimulus to use and the data collection mode. Choose Discrete to calibrate with narrow band signals, otherwise choose broadband.

**Stimulus Class**

Discrete (e.g. Pure Tone)

Broadband (e.g. White Noise)

**Data Collection Mode**

Manual Data Entry

Automatic Data Collection

**Scaling**: Logarithmic

**Points/octave**: 10

**Min frequency**: 100

**Max frequency**: 16000

**Number of Averages**:

< Back   Next >   Close   Help

Following configuration of the calibration process, a second series of wizards walks the user through the actual calibration process with detailed instructions for proper hardware coupling.

**Transducer calibration**

Confirm the following output settings before calibrating ER-2

- The calibration stimuli will be delivered by D/A channel 1
- There is no programmable attenuation used with this transducer.
- There is 0 dB of gain (excluding programmable attenuation).
- A voltage of 1 V RMS will be delivered to ER-2
- Couple ER-2 to ER-7C
- Make sure your microphone output is connected to A/D channel 1
- There is 0 dB of gain in the A/D path

Continue   Abort

As the calibration data are collected, values are displayed in tabular format as well as graphically. The data may then be saved in a simple calibration file (.bcf) for use by the SykofizX application during stimulus generation. Those files can be re-loaded at a later date for examination, or exported to another software application for analysis. In addition to the raw calibration data, the stimulus generation unit may specify a specific frequency or range of frequencies and request that the application return either attenuation (dB) or digital scaling (linear) factors.

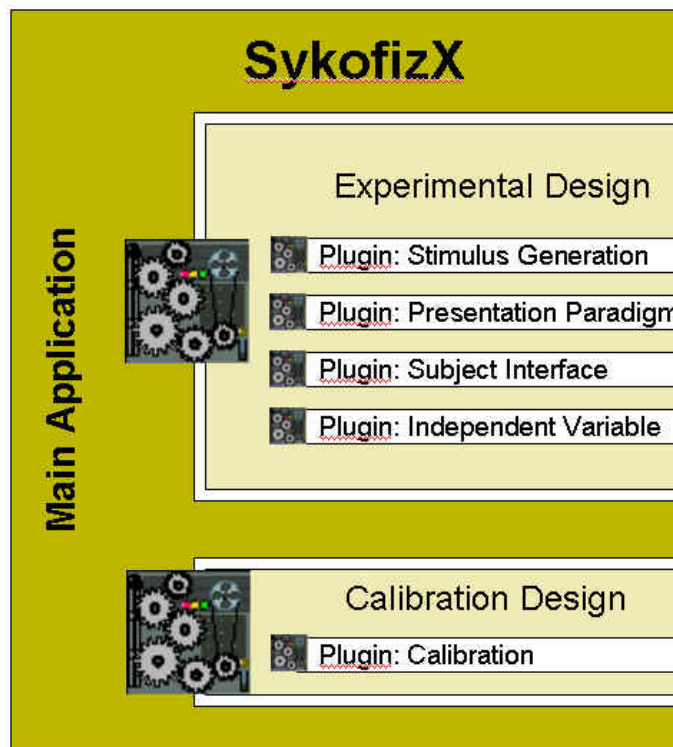
## SykofizX Architecture

Critical in the experimental design and calibration processes is the selection of appropriate plugins used for:

- Stimulus Generation
- Stimulus Presentation
- Subject Interface
- Independent Variable Configuration

A *plugin* is simply a collection of instructions within a single (DLL) file that dictates the behavior of the application when that plugin is referenced. All plugins have a set of parameters that allow them to communicate with the main application. Individual experimental designs may share a great deal of common functionality but differ substantially in one or more aspects of the design. Exchanging plugins relevant to specific parts of the design allows the user to reconfigure the experimental design as needed without having to re-create an entirely new design from scratch.

The diagram below illustrates this architecture, in which plugins are selected from a list of available plugins to create an experimental or calibration design. The application then executes the design, including the commands specified in a particular plugin.



Together the presentation paradigm and independent variable configuration plugins specify the nature of the psychophysical procedure to be used. These include a variety of standard procedures such as same-different, yes-no, forced-choice, rating, identification, categorization, magnitude

estimation, and magnitude production. These procedures may be used in combination with fixed and adaptive independent variable control. The application supports simultaneous collection of both closed and open-set subject responses as well.

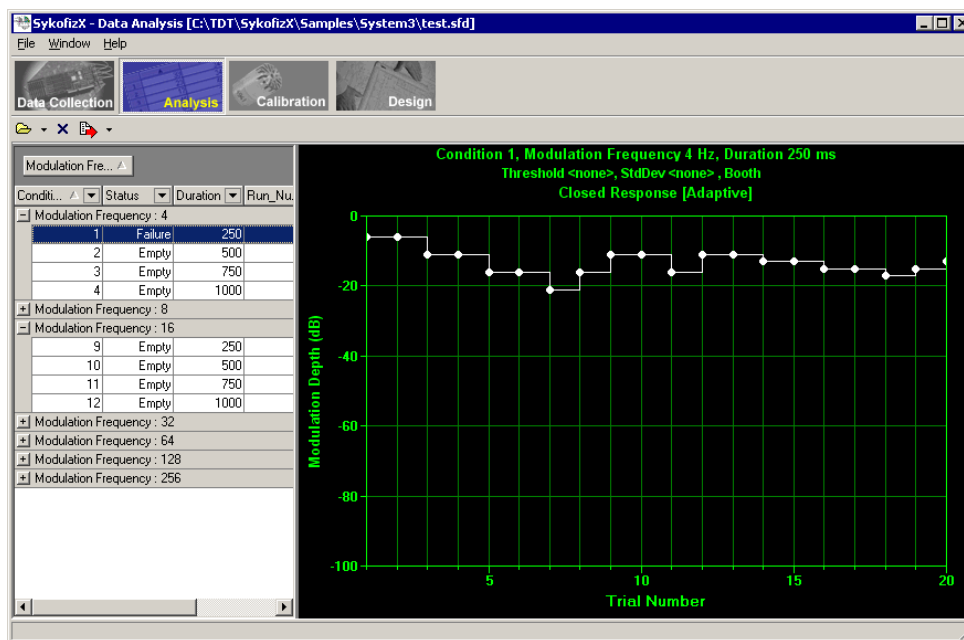
The stimulus generation and subject interface plugins are compatible with TDT System 3 as well as System II hardware. Using System 3 hardware, a combination of RCO files created in the TDT RpvdsEx application and custom scripts created within the SykofizX application allow the user to generate and control stimuli and the digital I/O necessary for specific subject interface requirements. Using System II hardware, extensive customization can be achieved using standard AP2 APOS commands written in a simple scripting environment.

For added convenience, the same custom scripts used with System II hardware may be used with System 3 hardware in association with an AP2 emulator (APE) and the APE.rco RCO file. In this case, the commands normally executed on the AP2 are actually executed on the CPU of the PC and a final stimulus buffer is passed to the RP2 device. In each case, the script routines can be mixed, matched, and modified by the user to create novel designs.

In addition to digital I/O via the TDT System 3 and System II platforms, subject interface routines support a variety of KVM (keyboard, video, and mouse) configurations. The KVM subject interface plugin also includes a configuration utility that allows the user to design a variety of KVM I/O configurations using combinations of static images, animations, movies, and audio clips. Subject interface script routines are easily customized to work with other hardware I/O systems such as button boxes, switches, feeders, and lights.

## Data Analysis

The data analysis mode allows the user to load individual subject data files and evaluate or export the data. The data export features support a variety of data organization schemes ranging from trial-by-trial data lists for all conditions to simple summary statistics for select conditions. The export features support a variety of file formats including text files (.txt), Microsoft Excel spreadsheet files (.xls), and extendable markup language (.xml) files. Unlike the data collection mode, data analysis does not require specific hardware to be present, providing flexible off-line interaction with data. For added convenience, the default view and alternate visualization options in the data analysis mode are similar to those in data collection mode.



Subject data files are stored in the convenient .xml format. Along with a record of the complete experimental design, the data file contains a trial-by-trial record for each experimental condition as well as appropriate summary statistics. The data analysis mode allows the user to filter the data files prior to exporting so that only desired information is ported to files suitable for other applications such as spreadsheet, data base, graphics, or statistical applications.

## ***Associated File Types***

### **Experimental Design Files (.sfx)**

Experimental design files store an entire experimental design in .XML format. The experimental design includes a list of plugins used, any parameter values specified, user-defined scripts, references to the appropriate sound files, image files, calibration files, etc. Each new experimental design must be stored in a new design file with the .sfx extension.

### **Subject Data Files (.sfd)**

Subject data files store a copy of the entire experimental design as well the data collected for a single subject. The stored data includes detailed trial-by-trial records as well as summary data (e.g., thresholds) for each experimental condition specified at design time. All records for a single subject on a given experiment will be stored in a single data file.

### **Calibration Files (.bcf)**

Calibration files contain a record of transducer calibration information for each transducer calibrated in a given calibration session. For each transducer, the calibration file has output (dB SPL) as a function of frequency normalized to a 1 volt transducer input.

### **KVM Configuration Files (.kvm)**

KVM configuration files store the keyboard/video/mouse subject interface configuration created in the KVM configuration utility.

### **Scripting Library Files (.sc)**

Script libraries (collection associated of scripts) may be exported from the Design file to independent script library files (.sc) for convenient importing into other designs.

### **RCO Digital I/O files (.rco)**

The TDT RPvdsEx control object file used by SykofizX to control System 3 hardware devices.

### **Miscellaneous Files**

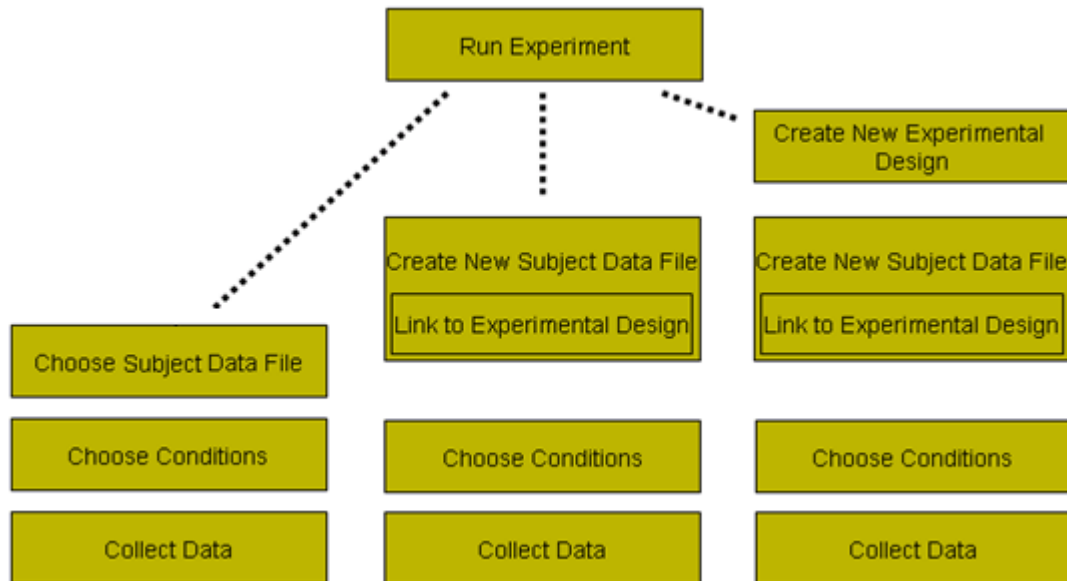
One may use a variety of other file types for the storage of stimulus related data, images and animations used with the subject interface, and the documentation associated with a given experiment.



# User's Guide

## Data Collection Mode

The subject data file contains a copy of the experimental design, any runtime parameters, and the data for each experimental condition. As data is collected, the data file expands. This format allows the data file to be completely portable and ensures that a record of the specific experimental parameters is stored with the data. Furthermore, this comprehensive data file provides the necessary trial-by-trial information to enable interleaved tracking as well as pause and resume features. The diagram below outlines the data collection process.



In data collection mode, the user also has a number of Evaluation Tools to facilitate setup, stimulus verification, subject instruction, training, and practice.

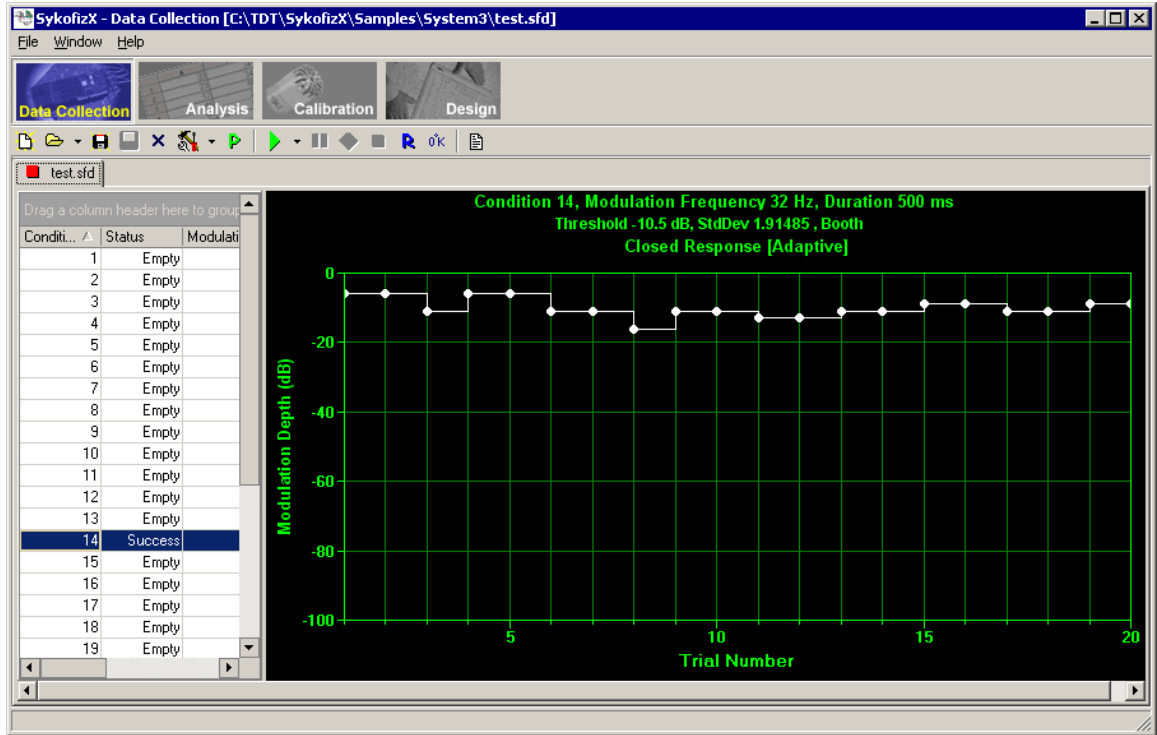
SykofizX supports data collection from single or multiple subjects, each with their own experimental design. The application can control multiple, independent hardware systems via network interface, allowing the experimenter to mix experimental designs, TDT System II and System 3 hardware, and subject interface hardware and their software routines.

Immediately prior to data collection, the **Data File Additional Values** window appears and additional subject related information may be entered. The Evaluation Tools are also launched, allowing the experimenter to:

- preview stimuli
- preview stimulus presentation options
- preview subject interface functions
- implement listener training (shaping) via sample or practice trials under either manual or automated control

The SykofizX application displays the data collection parameters for each subject on a different tab in the control window, as shown in the figure below. Data collection for each listener can be initiated, paused, and/or halted separately or simultaneously. A grid on each tab lists all of the experimental conditions and their parameters for a given subject data file. The experimenter may select one or more of these conditions for the current test session.


Multiple conditions may be run in a fixed or random order. The experimenter may view a custom dynamic display of current data values in graphical or tabular format depending on the nature of the data being collected. The specific conditions and condition sequence can be modified at any time during data collection.



The SykofizX application defaults to the Data Collection mode. From any other mode, switch to the data collection mode by clicking the Data Collection icon in the main application window or select Collect from the Window drop-down menu.

## Selecting an Existing Data File


An existing data file may be selected in one of three ways:

- Click the **Open** button  on the **Button Control Toolbar** and navigate to the desired file.
- Click the **tabs** located above the **Condition Grid** to select currently loaded data files.
- If no data files are currently loaded, click the **Data Collection Icon**. From the main application window, choose the **Open** icon on the toolbar and navigate to the desired file.

## Creating a New Data File

Recall that a subject data file is associated with an experimental design, so creating a new subject data file involves making that association.

A new subject data file may be created in one of two ways:


1. From the window in the center of the main application window, choose **New**.
2. Click the **New** button  on the **Control Toolbar**.

- a. Next, an experimental design must be chosen and linked to the subject data file. Use the new navigation window to locate and select the appropriate design file. If that file does not exist, it must be created in the Design mode before creating the subject data file.
- b. Finally, select a destination location and file name for the new subject data file.

The evaluation utilities wizard will automatically launch after the new subject data file is saved.

## Evaluation Utilities

In data collection mode, the user also has a number of evaluation tools available through the

**Evaluation Utilities**  button to facilitate setup, stimulus verification, subject instruction, training, and practice. Immediately prior to data collection, a variety of options are available to the experimenter for:

- recording subject or experimenter supplied information
- previewing stimuli
- previewing stimulus presentation options
- previewing subject interface functions
- listener training (shaping) via sample or practice trials under either manual or automated control

## Data File Additional Values

In Design mode, the experimenter may have specified information known as Additional Values (shown below) to be added to a subject data file at run time, such as Subject Name, Subject Age, Treatment Group, etc. The first run-time wizard that appears prompts the user to enter those Additional Values (see figure below). These values may be specified as optional or required during experimental design. If there are no additional values, that wizard is not displayed.

**Data File Additional Values**

**Data Collection** Please specify any additional values to be stored in subject data files. Additional values allow you to specify information relevant to your experiment. Examples of additional values include Name, Age, Treatment, etc.

Design Description :  
Ex01\_S3\_APE\_AMD\_Comp\_FC\_UD\_KVM.sfx

**Values saved once per data file**

Name	Test Subject 1
------	----------------

**Values saved once per condition**

Booth	1
-------	---

**Values saved once per trial**

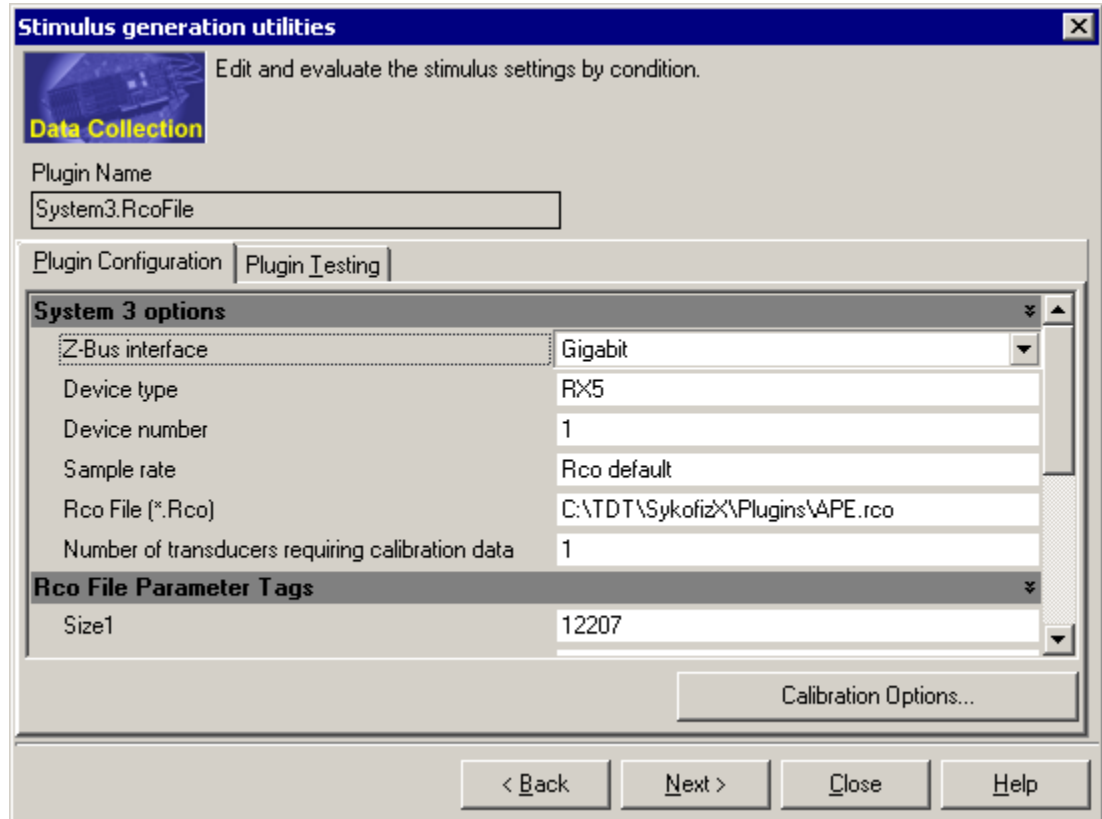
**User defined parameters**

Name	
Name	
Name	
Name	
Name	

< Back    Next >    Close    Help

## Stimulus Generation Utilities

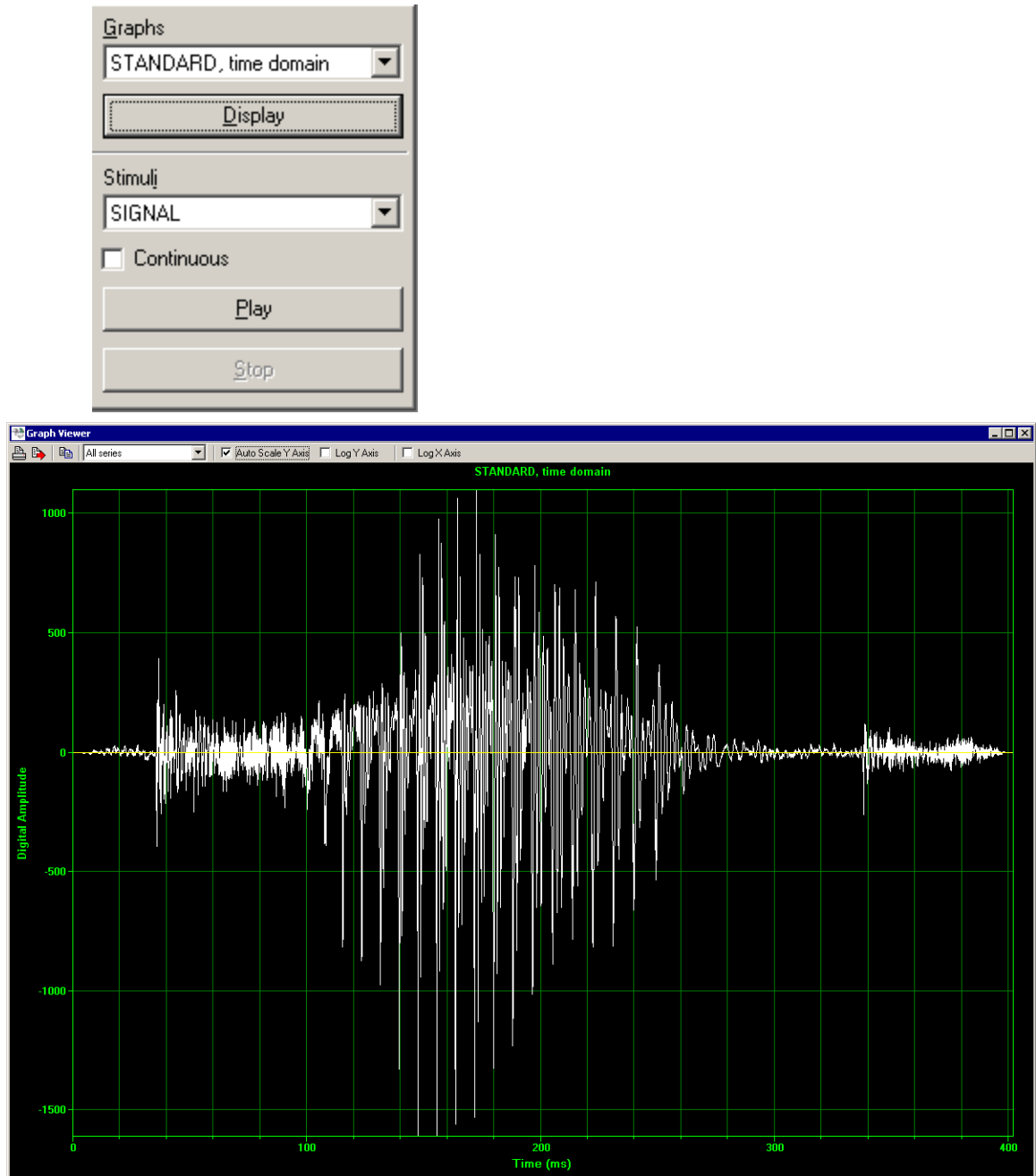
The next wizard displays Stimulus Generation Utilities. The first tab, **Plugin Configuration** (shown below), allows the user to view default stimulus hardware settings and other stimulus related settings. The user can modify these settings if they were made visible at runtime by the stimulus generation plugin specified in the experimental design. In most cases, these parameters will not need to be modified at runtime, and in some cases they should not be modified at runtime.



The second Stimulus Runtime Settings tab is labeled **Plugin Testing**. This tab displays any experimental conditions currently selected as well as the independent variable and its default value. If no conditions are selected, the first condition is listed by default.

To preview stimuli:

1. Choose the particular stimulus to be previewed from the drop down list labeled **Stimuli**. If the stimuli are generated by the TDT System II AP2 or the System 3 APE, they may be previewed graphically.
2. Select the stimulus to be displayed and the time or frequency domain from the drop-down menu labeled **Graphs**, and then depress the **Display** button to view the stimuli. The top of each display has several controls for customizing and/or printing the display.

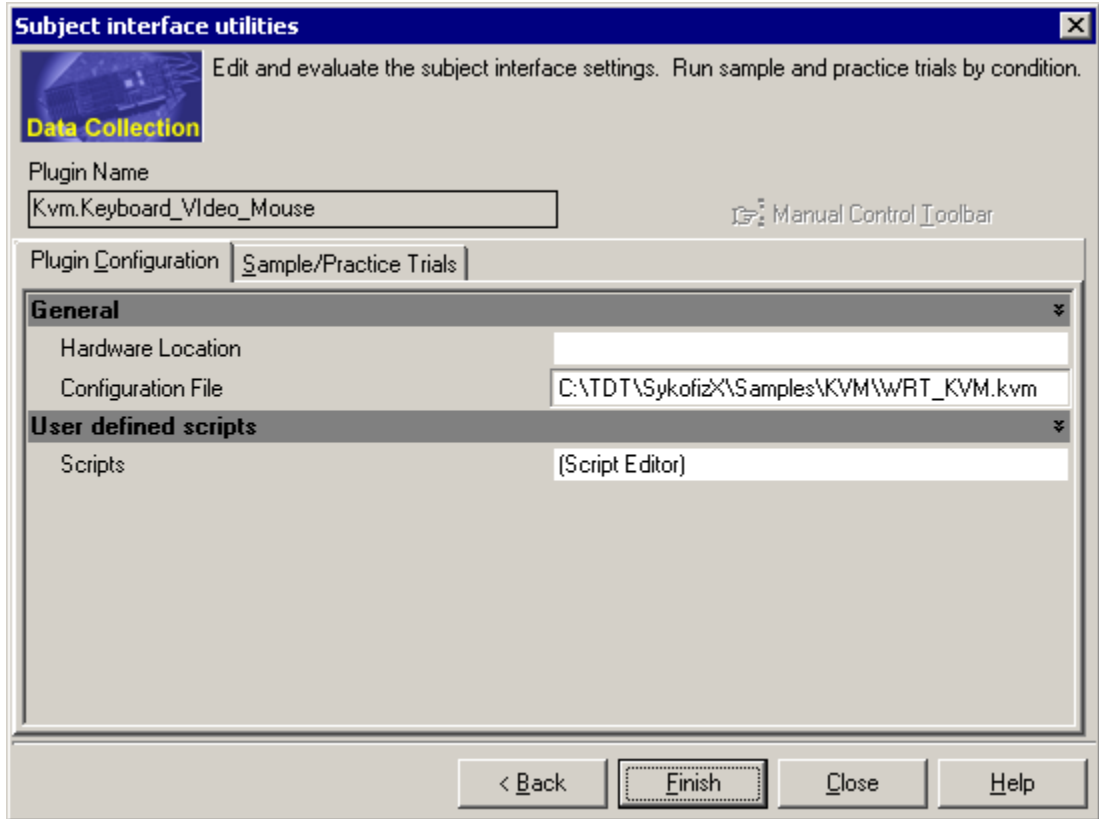


To preview the stimuli externally (e.g., to play the sounds or to route stimuli to an oscilloscope, voltmeter, analyzer, etc.), choose the stimulus class and select the **Play** button. Checking the **Continuous** box prior to selecting **Play** will play the sounds repeatedly until the **Stop** button is clicked. The initial independent variable value for a particular design (specified at design time) is displayed at the top of the Plugin Testing tab. The user may change this value as desired during stimulus evaluation by entering a new value in the edit box provided.

### Subject Interface Utilities

Next is the Subject Interface Utilities Settings wizard. Again, the **Plugin Configuration** tab allows the user to view default subject interface hardware, a configuration file, and any associated scripts. Settings on this tab can be modified if those parameters were made visible at runtime by

the stimulus generation plugin specified in the design. In most cases, these parameters will not need to be modified at runtime, and in some cases they should not be modified at runtime.



The second Stimulus Interface Utilities tab is labeled **Sample/Practice Trials** (shown below). This tab displays any experimental conditions currently selected, the independent variable, and its default value. If no conditions are selected, the first condition is listed by default.

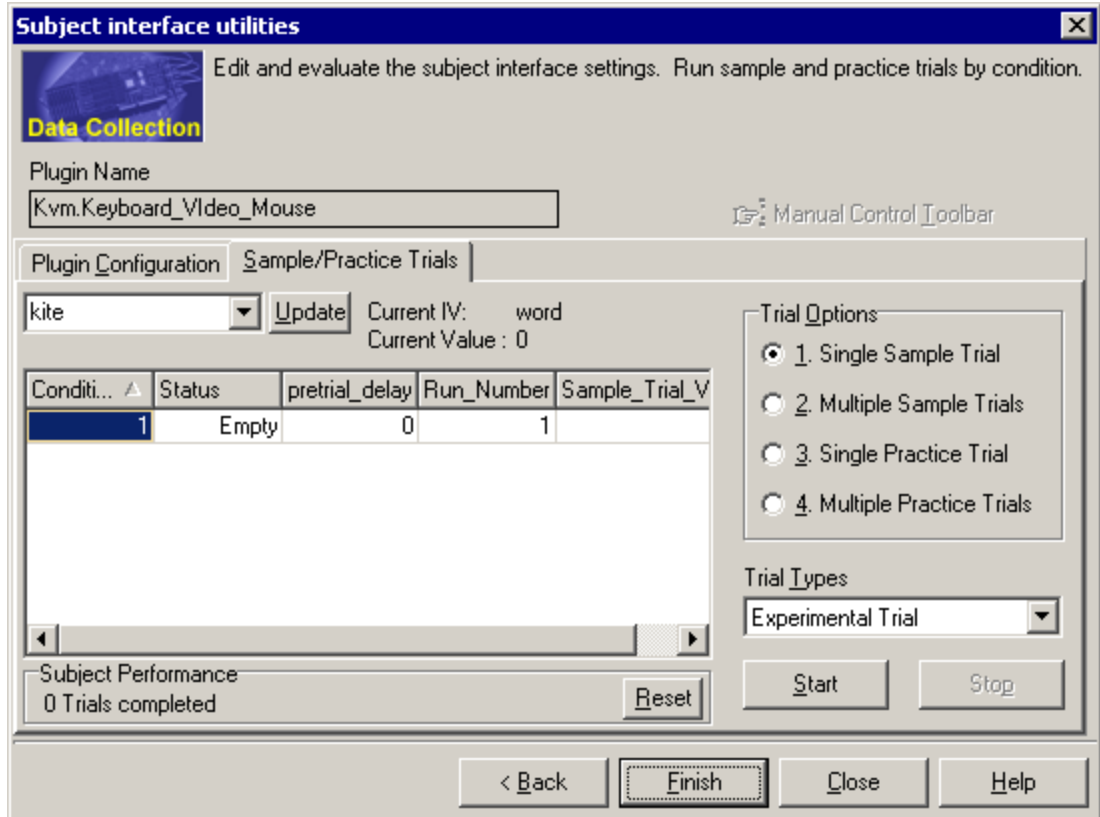
The options on this tab serve several purposes.

- One can verify the implementation of an entire trial, including stimulus generation, timing, and subject interface.
- In addition, one can select various Trial Options to assist in the instruction and training of subjects.

One may wish to start with single sample trials during initial subject instruction. Sample trials maintain the prescribed trial structure up to the last presentation interval, but do not include the subject response or feedback portions of a trial. Thus, they allow passive exposure to a trial.

Multiple sample trials might be used to allow the listener (or experimenter) to listen to sample trials in succession while focusing on some salient stimulus feature. Practice trials include the entire trial structure, permitting the listener to respond to the stimulus. During practice trials, a record of the listener's performance is displayed in the lower left panel of this wizard window.

Here, practice trials will obey the logic specified in Design mode, which may include a criterion number of correct responses in a criterion number of trials before the practice trials automatically stop. Of course, the Stop button will allow the experimenter to manually stop the practice trials at any time. Note that once a trial has begun, pressing stop will prevent the initiation of any subsequent trials but will not halt the current trial.



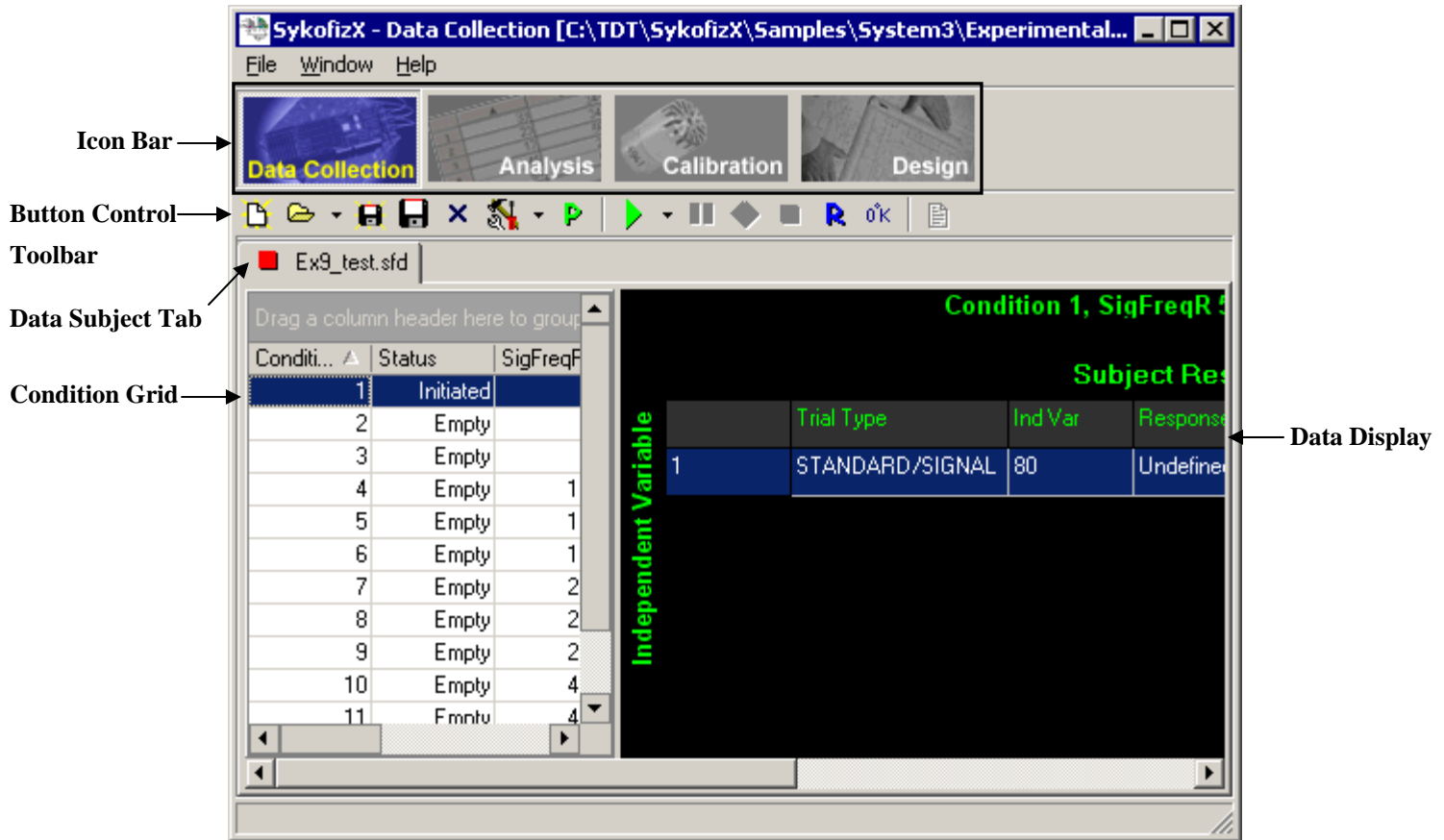
The Trial Types menu displays all of the available trial types and defaults to the Experimental Trial type. All experiment designs will have defined an Experimental Trial. Some designs may also have alternate trial types that may be used in instruction, training, or design evaluation.

For example, during a training trial for a two interval, forced-choice design, the experimenter may want the target stimulus to always be presented in the same observation interval, rather than being randomly presented in one of two intervals as it would be in an experimental trial. Thus, one might design a Training Trial type with the target locked in one or another interval.

Likewise, in a same-different paradigm, the experimenter might want to present the listener with each of the trial options prior to testing. Thus, trial types such as AA, AB, BA, and BB might be defined and selected here for presentation.

## Data Collection Window

The data collection window (shown below) has three main regions: the **Control Toolbar**, the **Condition Grid**, and the **Data Display**. The condition grid and data display correspond to the active subject data file - the tab that currently has focus. Note that up to 255 subject data files can be loaded at once. The tab attached to each subject data file displays the name of the data file and its status (play, pause, or halt). The proportion of the screen occupied by the condition grid and data display may be changed by clicking the center divider and dragging the divider in one direction or the other.




## Condition Grid

The Condition Grid has multiple functions. In addition to displaying all of the conditions for an experiment, the experimenter can select the conditions to be tested, re-order the conditions, and monitor the status of each condition.

### Condition ID

Each condition in an experiment has a unique Condition ID that determines the order of conditions listed in the Condition Grid. Conditions can be rearranged without changing the Condition ID. Once the data file is closed, the rearrangement is lost. To permanently change the Condition ID, and thus the condition order in the Condition Grid and in the subject data file, select the **Freeze**

**Condition Order** button  from the toolbar.

### Condition Status

Each condition has four possible status values:

- Empty: The condition has not been initiated, and thus has no data associated with it.
- Initiated: The condition has been initiated but has not been completed
- Success: The condition has been successfully completed
- Failure: The condition has terminated unsuccessfully. For example, this might occur if an adaptive track fails to converge on a threshold.

### Condition Parameters

From one to n columns to the right of the Status column may represent condition parameters. In the example below, modulation frequency and noise carrier bandwidth are condition parameters in a sinusoidal amplitude modulation detection experiment.



### Run Number

The same condition may be listed more than once. Each repetition is listed on a separate row and differentiated by a run number from one to n.

### Sample Trial Value

If sample trials were specified in the experimental design, then the value of the independent variable to be used in the sample trials was specified as well, and is listed here. Note that the user can change this value at runtime as needed if the "edit conditions grid" feature was selected when creating the experimental design.

### Initial Value

Some experimental designs specify a default initial value of the independent variable. This can be changed by the user at runtime if the "edit conditions grid" feature was selected when creating the experimental design.


### Selecting Conditions

To select a condition, simply click that condition. To select multiple conditions, click the first condition to be selected, and either press the control key and click additional conditions or press the shift key and click a second condition. The latter step will select all conditions between the first and second conditions.


### Rearranging Conditions by the Drop-and-drag Method

Use the left mouse button to click in any cell of a single condition, hold the button down, and drag the condition to another position in the condition grid. When the cursor is over the new location, release the button.

### Rearranging Conditions Randomly

Select the conditions to be re-ordered and press the **Randomize** button  on the Control Toolbar.

### Reordering Conditions

After conditions have been rearranged; clicking the **Freeze Condition Order** button  will permanently reorder the conditions by changing their Condition ID values.

### Grouping Conditions by Condition Parameters

Each condition parameter has a name that is displayed as a column heading in the Condition Grid. Selecting the cell with the column name and dragging it into the gray area above the condition grid causes the Grid to collapse across that parameter. The conditions associated with each parameter value can be expanded one at a time. This feature is particularly useful when the experimenter wishes to run a group of similar conditions sequentially in a single session or rearrange the order of conditions by condition parameter. Likewise, conditions listed in the condition grid can be sorted by condition parameter simply by clicking the column heading of the desired parameter.

For example, click and drag a condition parameter to group the additional conditions based on that parameter.

Individual conditions may be sorted in ascending or descending order.

Here, the condition parameter **Modulation Frequency** is used to group the additional conditions.

ConditionID	Status	Modulation Frequency	Duration	Run_Number
1	Empty	4	250	1
2	Empty	4	500	1
3	Empty	4	750	1
4	Empty	4	1000	1
5	Empty	8	250	1
6	Empty	8	500	1
7	Empty	8	750	1
8	Empty	8	1000	1
9	Empty	16	250	1
10	Empty	16	500	1
11	Empty	16	750	1
12	Empty	16	1000	1
13	Empty	32	250	1
14	Empty	32	500	1
15	Empty	32	750	1
16	Empty	32	1000	1
17	Empty	64	250	1
18	Empty	64	500	1
19	Empty	64	750	1

ConditionID	Status	Duration	Run_Number	Sample Trial Value
+ Modulation Frequency : 4				
+ Modulation Frequency : 8				
- Modulation Frequency : 16				
9	Empty	250	1	-6
10	Empty	500	1	-6
11	Empty	750	1	-6
12	Empty	1000	1	-6
+ Modulation Frequency : 32				
+ Modulation Frequency : 64				
+ Modulation Frequency : 128				
+ Modulation Frequency : 256				

### Data Display

The data display to the right of the condition grid has numerous available formats, depending on the nature of independent variable manipulation specified in the experimental design. The desired data displays are selected and configured at design time on the independent variable design wizard.

- For designs that vary the independent variable adaptively, the data display will take one of two forms: a scatter graph or a data list. The scatter graph displays the independent variable values (y axis) on a trial-by-trial (x axis) basis. The data list displays all data saved on a trial-by-trial basis.
- For designs that have a fixed independent variable within a condition, the display may take the form of a histogram with a single bar displaying a percentage score for each independent variable value or a psychometric function. Other designs may permit a

matrix of responses by independent variable value, trial-type by independent variable value, or a data list table displaying the independent variable value and the corresponding response.

- Designs with fixed independent variables within a condition, such as identification or classification experiments, may display the data as frequency counts in a table or a confusion matrix.
- Data with open response values that are numeric display the results on a graph and values that are strings are displayed in a data list.
- When both open and closed response data are collected on each trial, two displays appear within the data display window.
- If catch trials are enabled, then a third portion of the display window shows the catch trial status.

The data display has relevant condition information near the top of the display and the values shown in the tables and graphs are updated after every experimental trial. The x and y axis ranges and labels can be modified from their default values at design time. At run time, right-clicking the graph displays the available custom display options. For adaptive tracks, the range of values plotted on the abscissa is contracted to reflect the current trial number and expands automatically as necessary. Likewise, the divider between the Condition Grid and the Data Display can be adjusted as needed with the mouse.

While data collection is in progress, the display on the screen depends on the type and configuration of the subject interface specified in the experimental design. For example, if the subject interface is a button box, then the display on the control computer is similar to the one shown above. If the subject interface is a custom KVM interface, then at design time the size of the video display must be specified to occupy a portion of the screen or the full screen.

However, a KVM interface may be deployed on a remote computer via the built in KVM server that is installed with the SykofizX application. This allows the control computer to display the main SykofizX window while the remote (subject) computer displays the KVM interface. Furthermore, KVM extenders (available from many hardware vendors) may be used to add, configure, and control additional KVM devices from a single computer.

## Data Collection Toolbar



Most options within the Data Collection mode are available from the toolbar. Buttons are described from left to right.

### Create a New File

This creates a new subject data file.

### Open

Open an existing subject data file. Next to the Open icon, a drop-down list of recent data files is available for selection.

### Save As

Save an existing subject data file with a new name.

### Save

Save any changes to a subject data file. Note that any newly collected data is automatically saved into the subject data file. However, if one modifies conditions in the condition grid, those changes must be saved manually.

### **Close**

Close the active data file. A warning message is displayed and requires confirmation to close the active data file.

### **Evaluation Utilities**

The Evaluation Utilities icon launches the Evaluation Utilities wizard. Next to the Evaluation Utilities icon is a drop-down list of utilities that will launch the desired utility.

### **Practice**

The Practice icon launches practice trials. Practice trials are identical to experimental trials but no data is saved from practice trial mode.

### **Play**

Play initiates data collection conditions selected in the active subject data file. Next to the Play icon is a drop-down list of alternate play options. These include:

- **Play multiple selected conditions sequentially:** The data for the first selected condition will be collected. Upon completion, the next condition in sequential order will be initiated automatically. This will continue until all conditions have been completed or until the experimenter halts data collection.
- **Play multiple selected conditions in random order:** The data for each of the selected conditions will be collected in random order. Following completion of the first randomly selected condition, data collection for one of the remaining selected conditions will be initiated. This will continue until all conditions have been completed or until the experimenter halts data collection.
- **Play multiple selected conditions, interleaved, in sequential order:** Beginning with the first trial of the first condition selected, each successive trial will jump from condition to condition in sequential order. Data collection will be completed when all trials for all conditions have been completed.
- **Play multiple selected conditions, interleaved, in random order:** Beginning with the first trial of the first condition selected, each successive trial will jump from condition to condition in random order. Data collection will be completed when all trials for all conditions have been completed. Note that due to random condition selection, it is possible that the final trials in a single session may be from the same condition.

When collecting data by interleaving tracks, the data for each condition are stored separately, as if they had been collected sequentially without interleaving.

### **Pause**

Pause temporarily halts data collection for the active subject data file.

### **Resume**

Resume data collection for the active subject data file.

### **Halt**

Halts data collection for the active subject data file. A warning message is displayed and requires an affirmative response to halt data collection.

### **Randomize**

Randomizes the order of all selected conditions for the active subject data file. The new order is reflected in the condition grid in the main application window. The newly determined order is not maintained once the data file is closed.

### **Freeze Condition Order**

Following re-ordering of conditions in the condition grid, this option permanently stores the new condition order in the subject data file, replacing any record of the original order.

### **Documentation**

Launch any documentation associated with the current experimental design and that document will be opened in the default software application (for example, an html file will be opened in the default browser).

## Data Storage

Each subject data file contains a copy of the complete experimental design as well as all subject specific information including Additional Values specified at run time, a detailed trial-by-trial record of the experiment, and condition summaries such as threshold values, final percent correct scores, etc. Subject data files are stored in XML format to maximize portability and readability. One may simply open the file in an internet browser application (or other XML-friendly application) and view the subject data file directly, expanding and contracting nodes as necessary.

The XML format is well suited to filtering and exporting to a variety of other formats or software applications. Subject data files are updated after each experimental trial. This permits data collection to be halted indefinitely and resumed as needed. Trial-by-trial updating enables conditions to be interleaved on a trial-by-trial basis without loss of information and without the need for numerous buffers of indefinite length.

## Remote Data Collection

The SykofizX application supports several means of remote data collection, depending on the available hardware and the requirements of a given experimental design.

## Calibration Mode

The Calibration Mode provides the SykofizX application with a powerful yet elegant way to calibrate transducers, to store and retrieve calibration data, and to use the calibration data during stimulus generation to ensure that each stimulus presented has the desired characteristics. Calibration data is stored in XML format using the file extension .bcf.

The calibration file contains all of the calibration data as well as details about the particular calibration configuration. The calibration data for each transducer is stored in units of dB SPL re: 1V transducer input in a frequency (Hz) dependent manner. By normalizing the transducer transfer function relative to 1V transducer input, the calibration data for any transducer is independent of the associated hardware, maximizing flexibility and portability.

## Plugin Selection

Similar to running a psychophysical experiment, calibration involves stimulus generation, data acquisition, and data storage. Thus, the Calibration mode has many features in common with the Experimental Design and Data Collection modes. Ultimately, the calibration process depends on every element in the stimulus generation and delivery path. Because an infinite number of pathways can be configured, there are two approaches to calibration.

The first involves calibration each time an experiment is run. This is routinely done in certain auditory physiology experiments, but is much less common in behavioral experiments.

The second approach is to normalize the calibration information so that it can be generalized across an unlimited array of stimulus pathways. The latter approach is taken here. The specific steps are detailed below and are outlined as follows:

- During the calibration process, a known voltage is delivered to the transducer via the SykofizX calibration plugin and associated hardware.
- The user must specify the amount of attenuation in the signal delivery pathway.
- A sound level meter coupled to the transducer being calibrated is used to measure the dB SPL value at the transducer output.

- Manual Calibration involves reading the sound level meter and manually recording the meter reading.
- Automated Calibration involves routing the meter output to an A/D device and specifying any attenuation in the signal recording pathway.

Having a record of the voltage delivered to the earphone, the output pathway attenuation, the voltage out of the sound level meter, and the input pathway attenuation, the Calibration plugin then scales the input voltage relative to the output voltage, and stores the transducer output for a nominal 1 volt input. The process makes the assumption of a linear input-output function over the range of measured levels.

The first step in Calibration is plugin selection, which is done in the first calibration wizard. The default choices are System II Calibration, System 3 Calibration, and Demo Calibration.

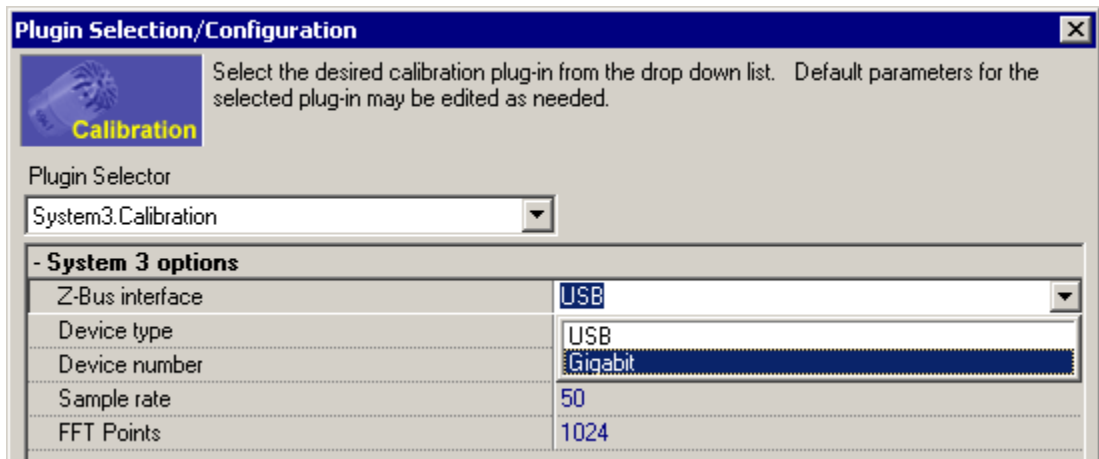
The Demo Calibration plugin allows the user to explore the calibration process without going to the trouble of setting up all of the necessary hardware for actual calibration.

The System II and System 3 Calibration plugins are designed for stimulus generation and data acquisition using devices specific to those TDT model lines.

Typically, System 3 calibration will involve an RP2 (or derivative device) which has both D/A and A/D functionality, supporting automated calibration.

System II calibration is designed for different System II hardware systems that may or may not include A/D and therefore may or may not support automated calibration (an A/D converter is required for automated calibration).

Once the plugin is selected, it must be configured appropriately. The figure below illustrates the configuration parameters available for the System 3 calibration plugin. Each item in the parameter inspector has either a drop-down menu that, when selected, displays the available choices for that parameter, or a data entry field that, when selected, allows the user to change the default settings.



## Stimulus Configuration

The Stimulus Configuration Wizard allows the user to choose from the available stimulus configuration options. The choices include continuous (e.g., noise or clicks) or discrete (e.g., tonal or other narrowband stimuli) stimulus types categorized on the basis of their long-term spectra. There is also a choice of manual or automatic data entry. If the discrete stimulus type is chosen, then the user must select the discrete sample point in the frequency domain, by choosing the interval between sample points and the lower and upper ends of the sampling range.

Finally, if the continuous stimulus type (Broadband) and Automated Data Collection options are chosen, then the user is prompted to enter the number of averages when computing the input spectrum.

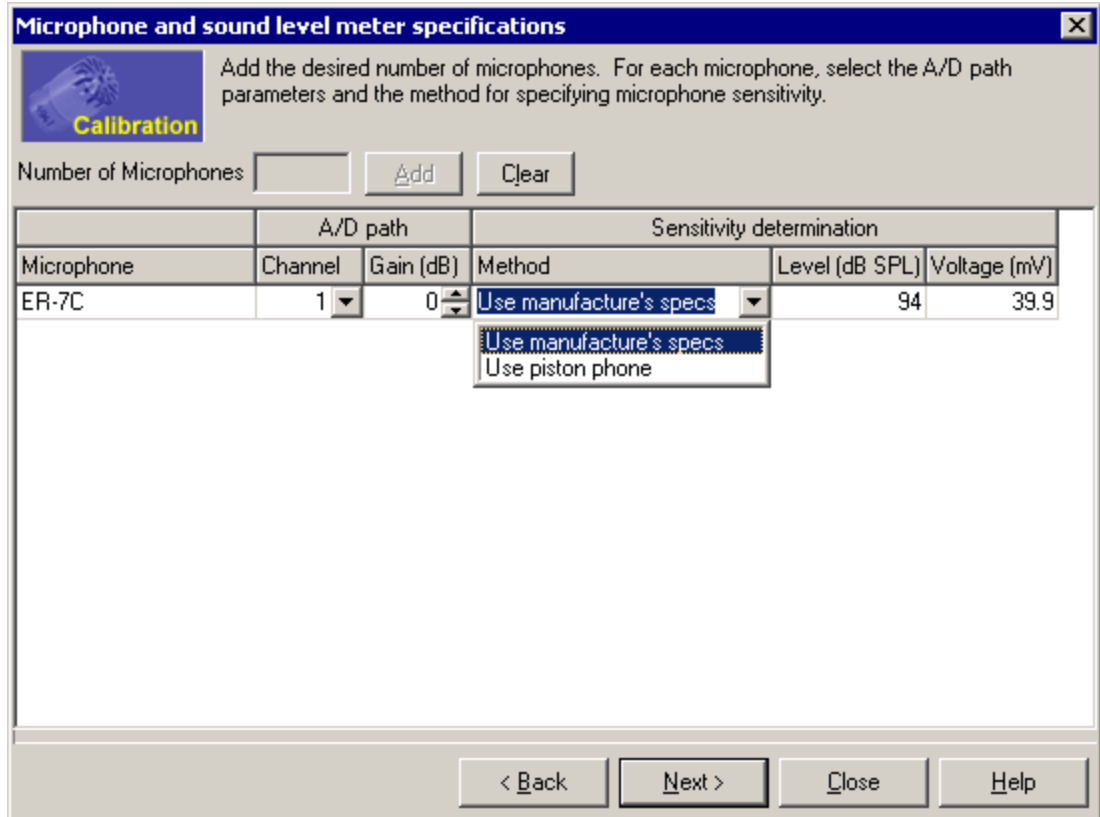
Once Stimulus Configuration is complete, the Next button is used to move to the **Microphone and Sound Level Meter** Specification wizard.

## Microphone and Sound Level Meter Specification

The SykofizX calibration mode supports the use of multiple microphones and calibration of multiple transducers sequentially. Each microphone to be used must be added to the calibration specification. Microphones may be given a name, under the Microphone column and they must be associated with an A/D path if automatic data collection was chosen on the previous wizard. The A/D path consists of the A/D channel number and the gain, if any, in the A/D pathway.

Next, the sensitivity of the recording device (microphone-sound level meter or probe-microphone system) must be determined, either using a reference device (e.g., piston phone or calibration cavity), or the manufacturer's specification for sensitivity. These options are available from the Method drop-down menu. Finally, the level in dB SPL and corresponding voltage as determined by piston phone or manufacturer's specification must be entered.

These steps provide the application with the means to convert dB SPL to a voltage reference for normalization and data storage.



**Number of Microphones**

Enter number of microphones to be used in calibration session (typically 1). Once the number is entered, the Add button is enabled and must be selected. Clear allows all specified microphones to be cleared.

**Microphone**

Name of each microphone to be used.

**A/D Path: Channel**

Choose A/D channel (1 or 2).

**A/D Path: Gain (dB)**

Enter Gain in A/D pathway in dB.

**Sensitivity Determination Method**

Piston phone reference or Manufacturer's Specifications.

**Sensitivity Determination Level (dB SPL)**

dB SPL reference value.

**Sensitivity Determination (Voltage (mV))**

Sensitivity in mV of microphone/sound level meter or probe-microphone system relative to the level defined above. This option is only available if Manufacturer's Specifications is chosen above. When Piston Phone is chosen, a wizard walks the user through the sensitivity determination (connecting piston phone to microphone of SLM, routing output to specified A/D).



## Hardware I/O Configuration

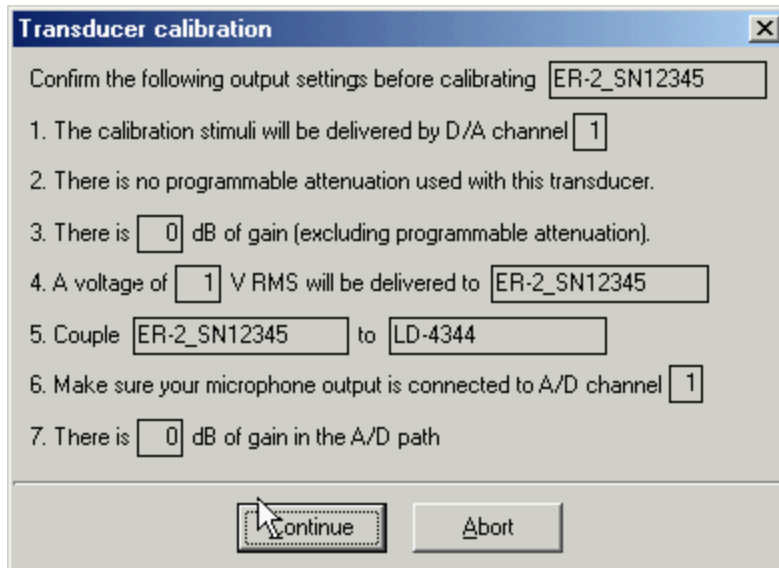
The Hardware I/O Configuration wizard maps the transducer to a microphone and specifies the stimulus generation pathway. Once the number of transducers to be calibrated have been entered and added, a corresponding number of rows will be added to the grid below. Each transducer can be given a name. As in the previous wizard, the D/A channel number can be specified, along with any manual gain in the pathway. If a programmable attenuator is included in the pathway (e.g., PA4 or PA5) the application will include its value automatically. The target voltage to be delivered to the transducer must be specified.

For many systems, 1 volt will be convenient, however, that may not be an appropriate voltage for systems with very sensitive or insensitive transducers. Finally, the microphone to be used must be associated with a particular transducer. While not routinely used by many experimenters, multiple microphones may be used by some experimenters when calibrating different transducer types in one session, or if real-ear calibration measures are being made.

Transducer	D/A to transducer path				Microphone
	DA_Channel	DA_Gain	Attenuator	Target_Voltage	
ER-2_SN12345	1	0	None	1	ER-7C

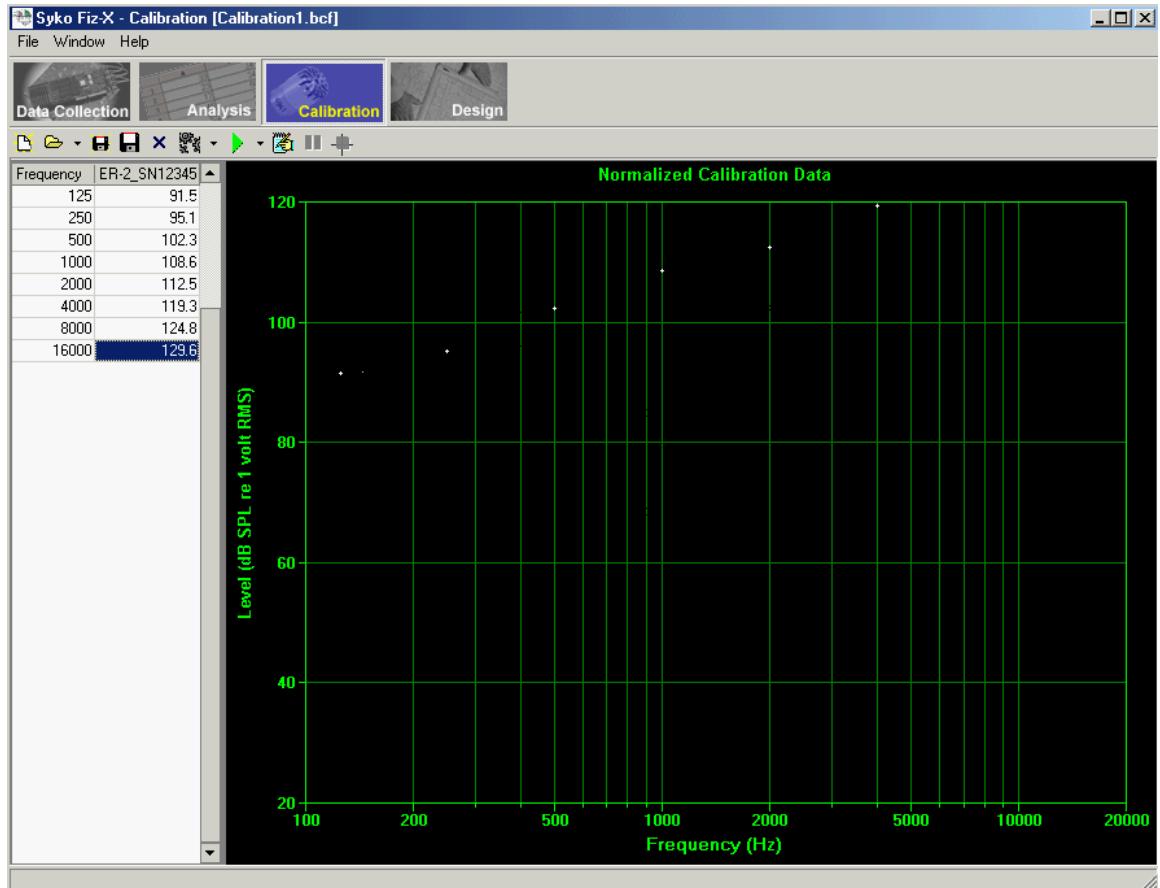
## Calibration Data Collection

The collection of calibration data has been simplified by the use of several instruction windows that walk the user through the calibration process. These windows help to insure that the correct hardware is in the specified I/O path. If multiple transducers are being calibrated, the instruction windows remind the user how to set up the correct path for each separate transducer immediately prior to calibration. Following these instructions will minimize the potential for calibration errors.



During automatic calibration, the transfer functions are dynamically displayed in the table on the left and the graph on the right of the display window. During manual calibration, the user may manually enter the calibration data in the table provided in the display window.

**Note:** The tabular and graphical display of calibration information shows the normalized calibration data. Thus, regardless of the specified target voltage, the data displayed are normalized to a one volt input.



**Note:** The calibration values above are fictitious.

## Calibration Data Storage

The calibration transfer functions displayed in the data table are saved in a calibration file with the extension .bcf. When multiple transducers are calibrated at once, the data for all transducers are saved in the same calibration file. Furthermore, the experimenter may combine or parse calibration files using a text editor. This maximizes portability and can reduce the number of calibration files that must be maintained by a given laboratory. Note that calibration data may be stored on a central file server and accessed remotely via networking.

## Use of Calibration Data in Stimulus Generation

The calibration data collected in calibration mode may be directly accessed during stimulus generation, allowing the application to adjust stimulus output levels dynamically, according to the spectral characteristics of the stimuli being generated and the transfer function of the transducer being used.

The calibration data can be read from the data file in two ways:

- Discrete values: The stimulus generation procedure may request the transducer characteristic at a single frequency.
- Average across a specified range: The stimulus generation procedure may request the transducer characteristics averaged across a range of frequencies.

In both cases, frequencies not present in the transfer function will be approximated by cubic-spline interpolation or extrapolation. The attenuation assigned to frequencies below or above the range of actual values in the transfer function is assigned values equal to the lower or upper limit of the transfer function.

The calibration data can be used to adjust stimulus levels in two ways:

- The stimulus generation procedure can request attenuation values in dB that can be used by external attenuators to achieve a specified output level.
- The stimulus generation procedure can request the linear gain required to digitally scale a stimulus to achieve a specified output level.

See Stimulus Generation Scripts, page 58 for more information regarding usage of calibration information in stimulus generation and presentation.

## ***Experimental Design Mode***

The Experimental Design Mode is perhaps the most powerful segment of the SykofizX application. The built-in flexibility and extensibility can reduce the time it takes to design a new experiment from days or weeks to a matter of minutes or hours. The key to efficient experimental design is the decomposition of all experimental designs into simple modules or functional units that can be configured individually to achieve minor or major modifications in an experimental design. The ten functional units that serve as the basis for all experimental designs created in the SykofizX application are:

- General Experiment Configuration
- Stimulus Generation Configuration
- Presentation Paradigm
- Subject Interface Configuration
- Stimulus Timing Configuration
- Response Timing Configuration
- Define Experimental Variables
- Independent Variable Configuration
- Practice Trials Configuration
- Additional Values to Store in Data Files

These functional units also correspond to the ten wizards that are available when designing a new experiment or modifying an existing experiment. Complete experimental design, from start to finish, involves one or more of the following steps:

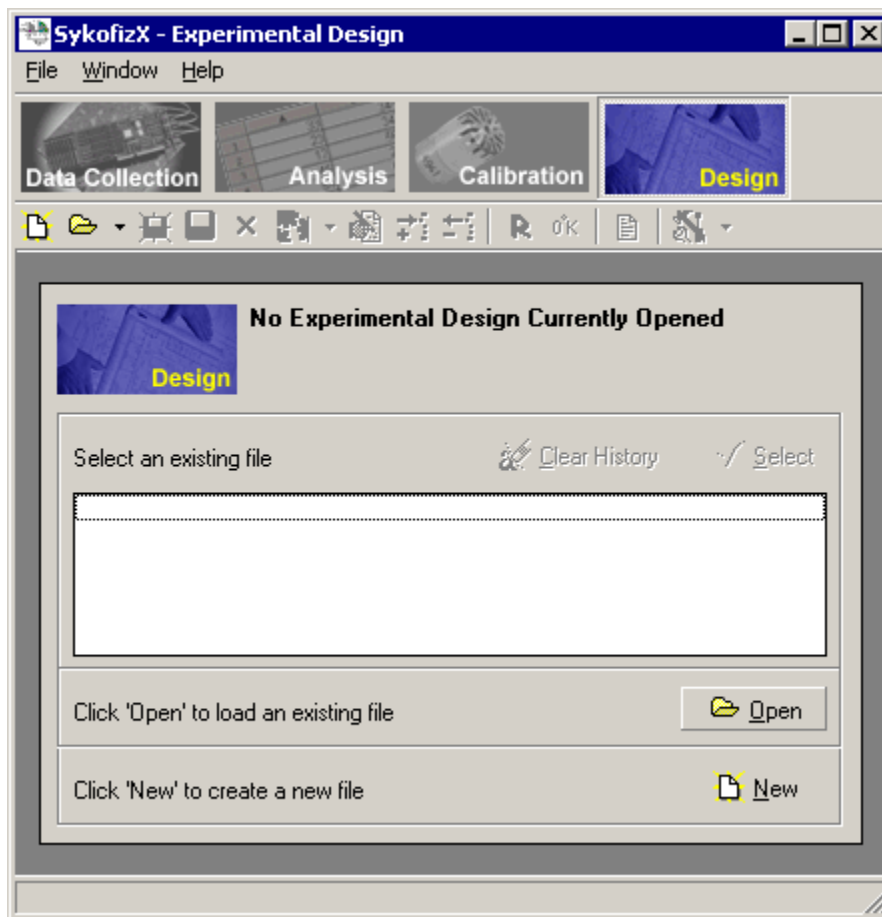
- Parameter specification for each functional unit of the experiment via the experimental design wizards.
- Specification of custom stimulus generation procedures to be used by the chosen stimulus generation plugin.
- The standard stimulus generation plugins include System 3 and System II plugins, to be used when generating stimuli with either System 3 or System II hardware and associated software. The System 3 plugin requires that an RCO file be specified. The System II plugin interacts with the System II AP2 Array Processor via user defined scripts.
- Specification of custom subject interface procedures to be used by the chosen subject interface plugin.
- Analogous to the stimulus generation procedures used with System 3 and System II plugins, subject interface plugins may also employ custom scripts and/or RCO files to interact with System 3 and System II I/O devices. In each case, the specific digital I/O

commands are specified in the scripts, and are routed to either the System II PI2 parallel interface module or the RP2 digital I/O interface via an RCO file.

- An alternative to using TDT Digital I/O capabilities for the subject interface is to design the interface around a computer's keyboard/video/mouse (KVM) capabilities. SykofizX provides a comprehensive utility to design a variety of KVM interfaces. The KVM interface is deployed on a custom KVM server to facilitate remote control of the subject interface. See KVM Subject Interface Scripts, page 68 for more information.

## Selecting an Existing Design File


Upon selecting the Design mode by choosing the **Design** icon at the top of the main SykofizX window, the user is prompted to open an existing Experimental Design file or to create a new Design. To open an existing Design file, select one from the list of recently opened design files or choose the **Open** option.



## Creating a New Design File

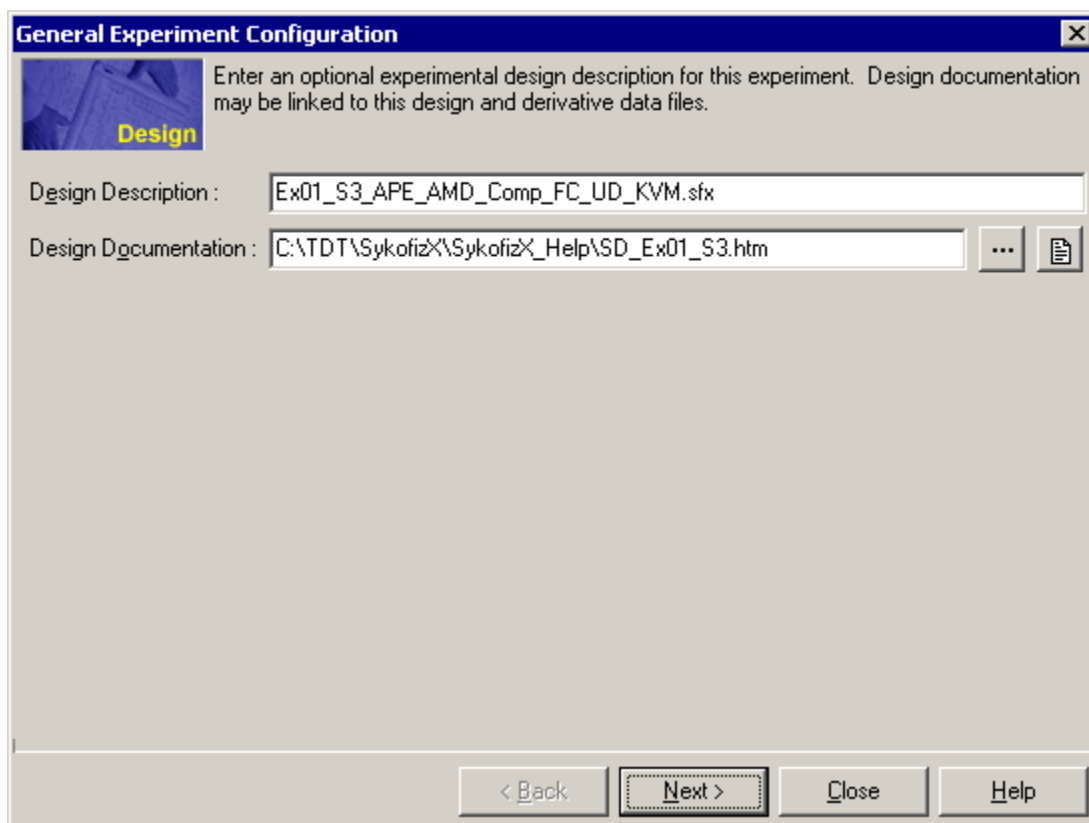
To create a new Experimental Design file, select **New** option. This will launch the design wizards beginning with the General Experiment Configuration Window.

## Experiment Design Wizard Navigation

Selecting **Next** will close the current Experiment Design wizard and launch the next Experiment Design wizard. Selecting **Back**, when available, will return to the previous design wizard. Once a new design is created, the user may return directly to any wizard from the main design window by selecting the desired design wizard from a drop down list or using the **Configure**  button. **Close** will close all Experiment Design wizards and return to the main design window.

## General Experiment Configuration

The first step in creating a new experimental design is to specify a description of the design and to associate user-defined documentation with the experimental design.




### Design Description

A brief description of the design may be typed here. This description will be recorded within the experimental design file and within each subject data file created from this design. Each time the experimental design is opened in Design mode, the description will appear. Likewise, the design description will be available when reading or parsing the subject data files created from this experimental design. The Design Description is optional and may be left empty.

### Design Documentation

Detailed information about the experimental design may be linked to the Experimental Design using the Design Documentation feature. Any file type may be associated with the experimental design, permitting the user to view run-time instructions, circuit diagrams, etc. from the Data Collection mode. A record of the documentation file name and directory path is stored in the Experimental Design.

If an Experimental Design has linked Design Documentation, then the **Documentation**  button on the main application toolbar will be active whenever the Experimental Design (or subject data file linked created from the design) is loaded (i.e., in Design mode, Data Collection mode, and Data Analysis mode).

## Stimulus Generation Configuration

In the Stimulus Generation Configuration wizard, the desired stimulus generation plugin is selected and configured using the plugins parameter inspector.

### Plugin

The desired stimulus generation plugin must be selected from the list of plugins available in the drop-down menu below the Plugin label. The three standard Stimulus Generation Plugins are:

#### **System3.RcoFile**

This plugin uses an RCO file designed in RPvdsEx as an interface between the SykofizX application and the System 3 devices. This is similar to other TDT System 3-compatible software applications.

Several sample RCO files are provided with the SykofizX and the RPvdsEx applications. The RCO files used in association with the SykofizX application are not fundamentally different from any other RCO files. They each use parameter tags to allow the parent application to send information to the hardware about the required stimulus parameter values (such as tone frequency, noise bandwidth, etc.). The System3.RcoFile plugin has the added requirement that each RCO file have output tags that allow the SykofizX application to determine what stimulus classes are supported by the RCO file, which software triggers are used to control stimulus I/O, and to monitor the activity of a certain stimulus class. The sample RCO files installed with SykofizX include files designed to load waveform data files from disk and files that interface with the APE (Array Processor Emulator).

#### **SystemII.ScriptedStimulus**

This plugin provides the user with complete control over stimulus generation via user-defined scripts containing a series of System II APOS and XBdrv commands to control the AP2 and the System II peripheral devices supported by SykofizX. These commands are saved as a collection of scripts and are executed automatically by the SykofizX application.

A variety of sample scripts are provided. In addition to the native APOS data buffer management commands, the SystemII.ScriptedStimulus plugin facilitates stimulus generation by providing optional commands that automatically handle much of the routine data buffer management such as allocation and de-allocation of DAMA buffers, mapping DAMA buffers to D/A channels, and setup of the D/A commands necessary to play sounds via System II D/A devices.

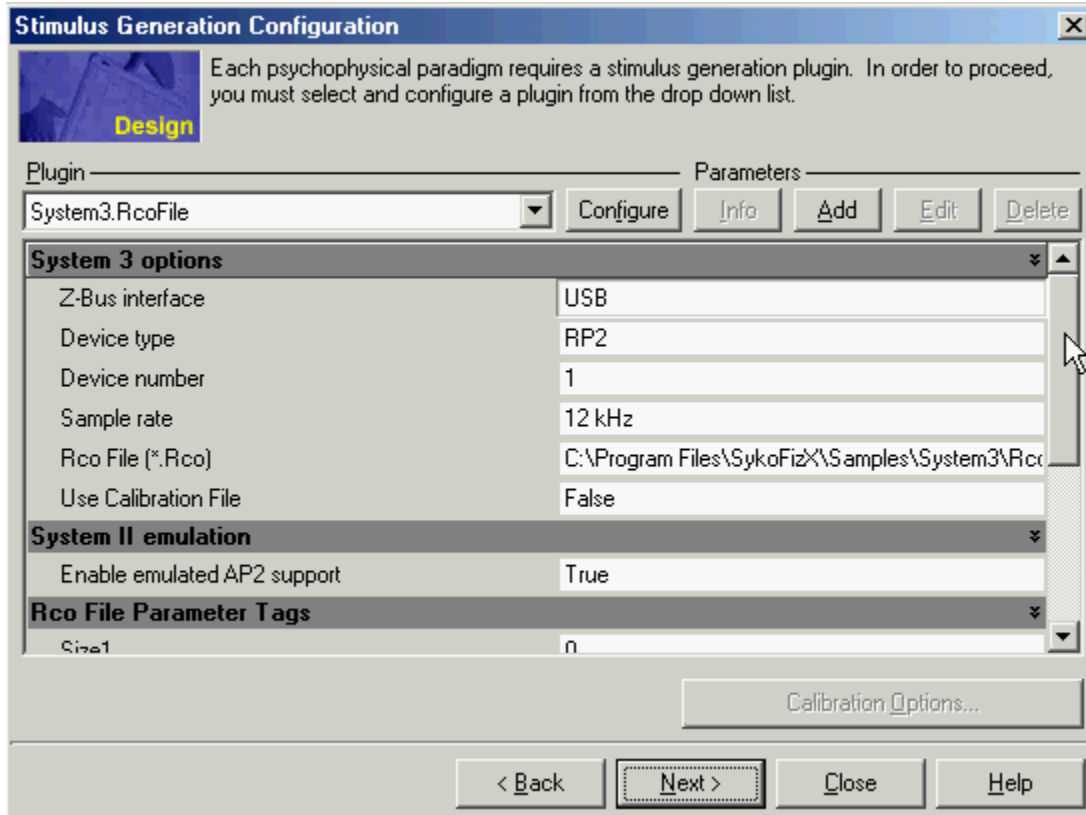
#### **DemoStimulus**

The DemoStimulus plugin allows the user to bypass the stimulus portion of an experimental design so that other aspects of a design can be configured or evaluated before focusing on stimulus generation. Users may find this helpful when exploring the features and functionality provided by the SykofizX application or when working "off line" without TDT System II or System 3 hardware.

### Parameter Inspector

Once a Stimulus Generation plugin has been selected, a parameter inspector is loaded into the main window. Each option available from the plugin appears in the parameter inspector and can be customized by the user. In the example below, the System3.RcoFile plugin has been selected. This plugin has several available parameters grouped by common functionality. The parameters and parameter groups differ across plugins.

This plugin has five parameter groups: System 3 Options, System II Emulation, Rco File Parameter Tags, User Defined Parameters, and User Defined Scripts.



### System 3 Options

TDT System 3 hardware is controlled by the RCO file specified by the user in the RCO File parameter of the System3.RcoFile plugin. The user may choose one of the sample RCO files provided with the SykofizX application or develop an appropriate RCO file using RPvdsEx. The SykofizX application loads the specified file, reads its contents, and interacts with any parameter tags in the RCO file including software triggers. See System 3 Stimulus Generation Scripts, page 60 for more information. Using custom scripts, the user may access to the parameters specified by parameter tags in an RCO file, giving the user an added layer of control over stimulus-application interactions.

System II Array Processor Emulation (APE) allows users to execute APOS (System II Array Processor Operating System) commands specified in custom scripts without actually using an AP2. The APOS commands are executed on the PC's CPU and a stimulus data buffer is "popped" to the RP2. To use the APE, the System3.RcoFile plugin is selected and the APE.rco file, supplied with the SykofizX application, is specified. This allows the user to harness the enormous signal processing power of the APOS and the precision of the System 3 DSPs and digital I/O capabilities.

### System II Options

System II hardware is controlled by scripts associated with the SystemII.ScriptedStimulus plugin. The scripts provide a simple environment for the specification of APOS and XBus commands required by System II devices. See System II Stimulus Generation Scripts, page 59 for more information.

### Parameter Tools

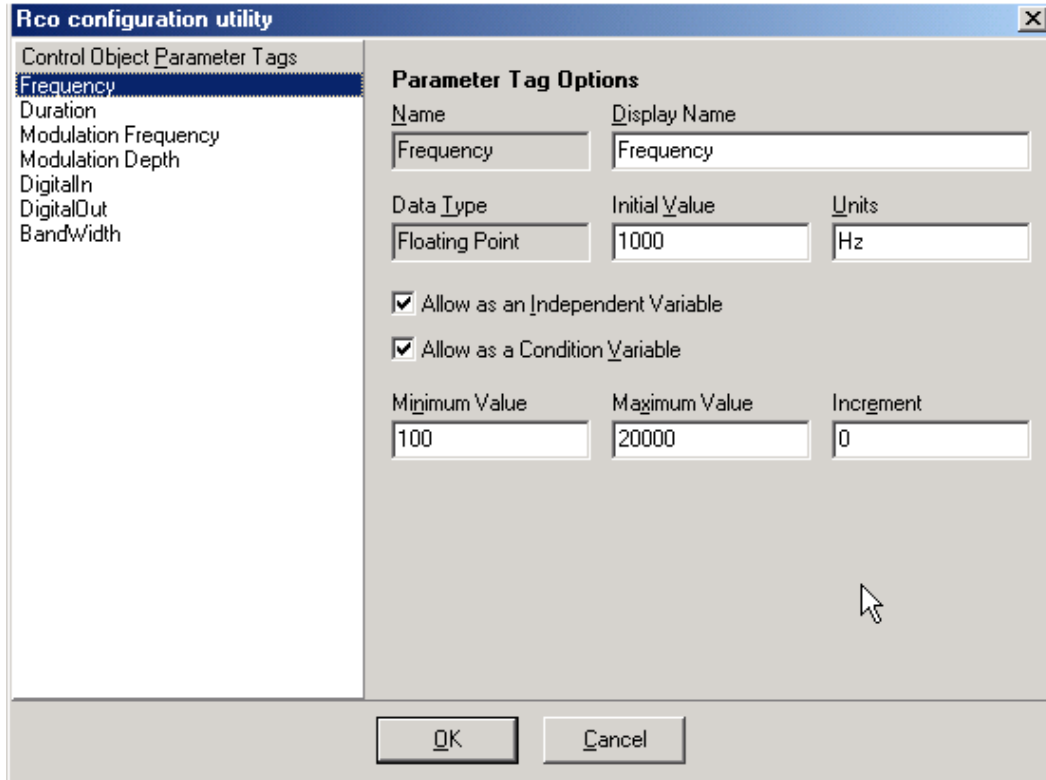
A given stimulus generation procedure may require additional parameters that are not associated with the stimulus generation plugin and thus are not included in the parameter inspector. These additional parameters are defined and modified using the following parameter tools: Configure, Info, Add, Edit, and Delete. Examples of such parameters might include *Stimulus Level* controlled

by programmable attenuators, *Vowel Categories* used as conditions (e.g., /eh/, /oh/, /ih/, /ii/, /ae/, /ah/), etc.

**Note:** It is often the case that during the experimental design process, one may realize the need for user defined stimulus parameters later in the design process (e.g., when selecting condition parameters). When this occurs, simply go back to the Stimulus Generation wizard and add the necessary parameters.

**Configure**

Some plugins provide configuration utilities that are launched using the Configure button on the wizard. For example, the System3.RcoFile stimulus generation plugin allows the user to configure names, ranges, and default values for the parameter tags in a specified RCO file. In this case, clicking the Configure button launches the RCO configuration utility shown below.



The parameter tags made available by the System3.RcoFile plugin that were listed in the parameter inspector of the Stimulus Generation Configuration wizard, are listed again in the upper left of the configuration utility. By selecting any one of the parameter tags listed, the default behavior can be specified and/or modified.

Furthermore, the possible functionality of these parameters can be specified, allowing the parameter to serve either as an Independent Variable or a Condition Variable or both. Later in the design process, parameters allowed to serve as independent or condition variables will be listed among the other parameters that could be chosen as independent or condition variables.

**Info**

Once a parameter in the parameter inspector is highlighted, the Info button will be enabled and a click of that button will display the details of that parameter.

**Add**

Parameters not specified by the plugin may be added at design time. These parameters may not be associated with the plugin itself, but may be important in stimulus generation or may serve as important parameters in choosing the Independent Variable or Condition Variable values.



Note that once the parameter is defined and added, it appears in the main parameter inspector as a User Defined parameter. The Info button provides a display of the parameter specifications. In the example below, the parameter Atten is added and represents attenuation values that may be fed to a programmable attenuator such as the PA5 to control stimulus level. The image below shows how to work with user defined parameters by specifying its options.

#### Delete

The Delete button allows a user defined parameter to be deleted.

#### Scripts

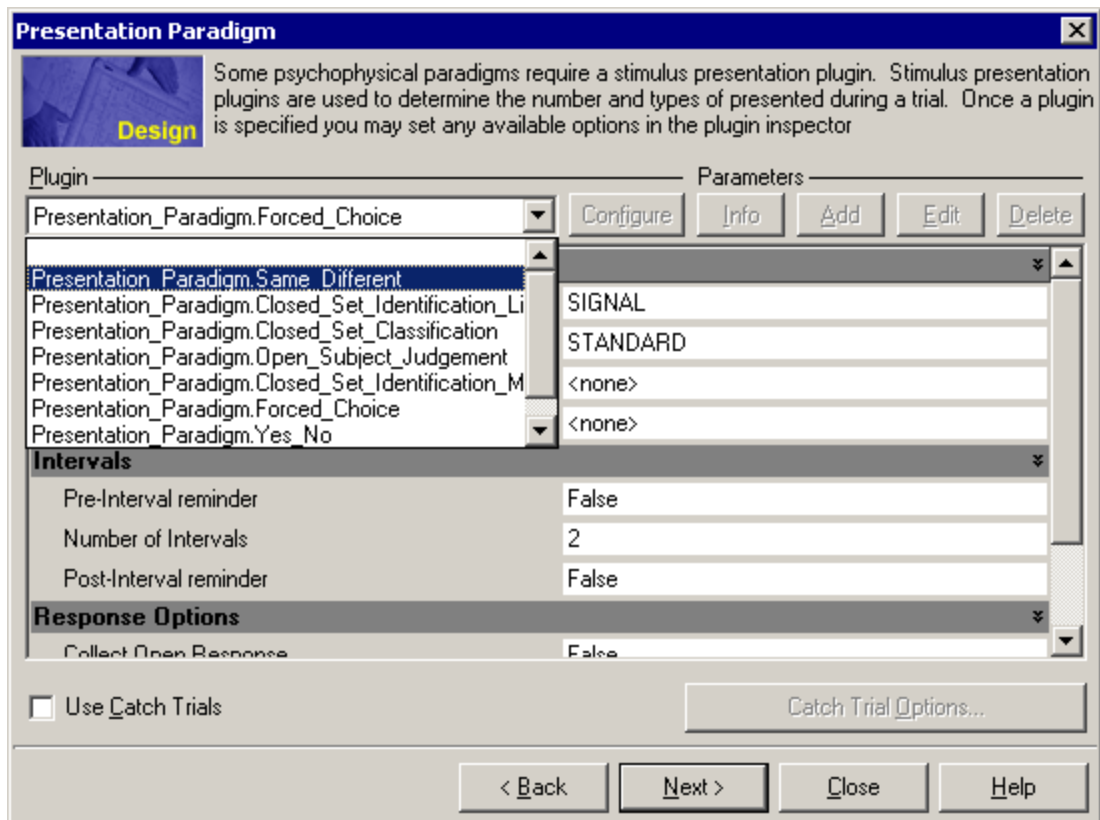
The final parameter at the bottom of the parameter inspector is the Scripts option. The System3.RcoFile plugin supports user defined scripts that may be used to control, among other things, user defined parameters and times at which they occur in the trial sequence.

## Presentation Paradigm

Each experiment requires a presentation paradigm defined in a Presentation plugin. Stimulus presentation plugins are used to determine the number and types of stimuli to be presented during an experimental trial. Once a plugin is chosen, the user may set any available options in the plugin parameter inspector.

#### Plugin

The desired presentation plugin must be selected from the list of plugins available in the drop-down menu below the Plugin label. Presentation plugins are often thought of as the psychophysical method used in an experiment. Below is a list of the available presentation plugins along with a brief description of each. For detailed information about each plugin, see Presentation Paradigm, page 78.



#### Available Paradigms:

##### Yes/No

This is the typical yes/no paradigm that may include a single or multiple observation intervals, with or without reminder intervals.

##### Forced Choice

Subjects are presented with  $n$  observation intervals and are "forced" to choose the target interval from the  $n$  choices. The end of an experimental trial occurs when a listener responds or when the established time-out occurs. Reminder intervals are optional.

##### Same Different

Subjects are presented with two observation intervals and must determine whether they were the same or different. Reminder intervals are optional. A go/no-go type procedure can be created using a pre-interval reminder constructed of multiple standard stimuli.

##### Subject Judgment

Subjects must judge some attribute of the stimulus or stimuli presented. Several different paradigms can be constructed using this plugin, simply by selecting appropriate options and by using specific subject instructions. Examples include stimulus rating, matching, magnitude estimation, and magnitude production.

##### Closed Set Identification - List

On each trial, the subject must choose the response alternative from a fixed set of alternatives that matches the stimulus token. Within a given condition, the number of stimulus possibilities does not correspond to the number of response alternatives. For example, a given condition might include 25 different stimuli and each stimulus might have four stimulus-specific response alternatives. Thus, the task is closed-set identification; however, the stimuli do not form a simple matrix and thus must be displayed as a list of stimulus values and responses. This is analogous to a multiple-choice experiment.

### Closed Set Identification - Matrix

On each trial, the subject must choose the response alternative from a fixed set of alternatives that matches the stimulus token. Within a given condition, a fixed number of stimulus possibilities and a corresponding number of response alternatives are available, creating a square stimulus-response matrix. This type of design is sometimes called a confusion matrix.

### Closed Set Classification

On each trial, the subject must assign the stimulus to one of a set of possible stimulus classes. For example, a single word from a large set of words might be presented on each trial, and the subject might be instructed to classify each word in terms of the age of the speaker: less than 10 years, 10 to 17 years, 17 years to 50 years, or greater than 50 years.

### Parameter Inspector

Once selected, a parameter inspector is loaded into the main window. Each option available from the plugin appears in the parameter inspector and can be customized by the user. Plugins have available parameters grouped by common functionality. The parameters and parameter groups differ across plugins.

### Parameter Tools

A given stimulus generation procedure may require additional parameters that are not associated with the stimulus generation plugin and thus are not included in the parameter inspector. These additional parameters are defined and modified using the following parameter tools: Configure, Info, Add, Edit, and Delete. The standard Presentation Paradigm plugins allow users to set the values of parameters that appear in the Parameter Inspector, but no parameters may be added, edited, or deleted. Thus, only the Info button is enabled when a parameter is selected.

**Presentation Paradigm**

**Design** Some psychophysical paradigms require a stimulus presentation plugin. Stimulus presentation plugins are used to determine the number and types of presented during a trial. Once a plugin is specified you may set any available options in the plugin inspector

Plugin: Presentation\_Paradigm.Forced\_Choice Parameters: Configure Info Add Edit Delete

Stimulus Classes	
Target stimulus class	Noise
Standard stimulus class	ModNoise
Reminder Stimulus Class	<none>
Catch Trial Stimulus Class	<none>

Intervals	
Pre-Interval reminder	False
Number of Intervals	2
Post-Interval reminder	False

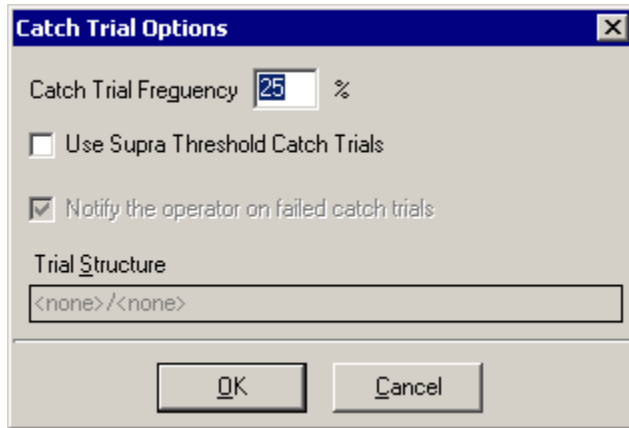
Response Options	
Collect Open Response	False

Use Catch Trials Catch Trial Options...

< Back Next > Close Help

### Catch Trials

Some Presentation Paradigm plugins support Catch Trials. When supported, the **Use Catch Trials** checkbox will be active. Checking this box activates the Catch Trial Options button. When selected, this button launches a window that allows the user to define the desired Catch Trial behavior.



The catch trial frequency must be specified to indicate the percentage of total trials that will be catch trials. By default, catch trials take the form of standard rather than target trials. A record of performance on Catch Trials is displayed in the main application Data Collection display during data collection and those trials are flagged in the subject data file and are not used when computing summary statistics.

Alternatively, some investigators may wish to use catch trials that take the form of target trials, in which case the independent variable is set to be some predefined level that is assumed to be supra-threshold. Like traditional catch trials, responses to supra-threshold catch trials are recorded in the data file but are not used when computing summary statistics. At design time, one may specify whether or not to notify the operator on failed supra-threshold catch trials.

### Data Collection Type

The standard Presentation plugins support two methods of collection data from the subject:

- Closed Response Data limits the subject to the designated response alternatives. This would include experiments with responses such as yes/no, forced choice, same-different, closed-set identification, closed-set classification, closed-set rating, etc.
- Open Response Data allows the user to provide an infinite number of possible responses and often referred to as open-set responses. The format of open response must be either numeric (integer or floating point) or string types. Numeric responses may be restricted to a pre-defined range by specifying the Minimum and Maximum allowable response values.

A given plugin may support both closed and open response types. When supported by one of the standard plugins provided with SykofizX, the closed response is the primary response and the open response is the secondary response. For example, the same different paradigm might also prompt the user for a confidence rating after each trial, and that rating would be recorded as an open response.

## Subject Interface Configuration

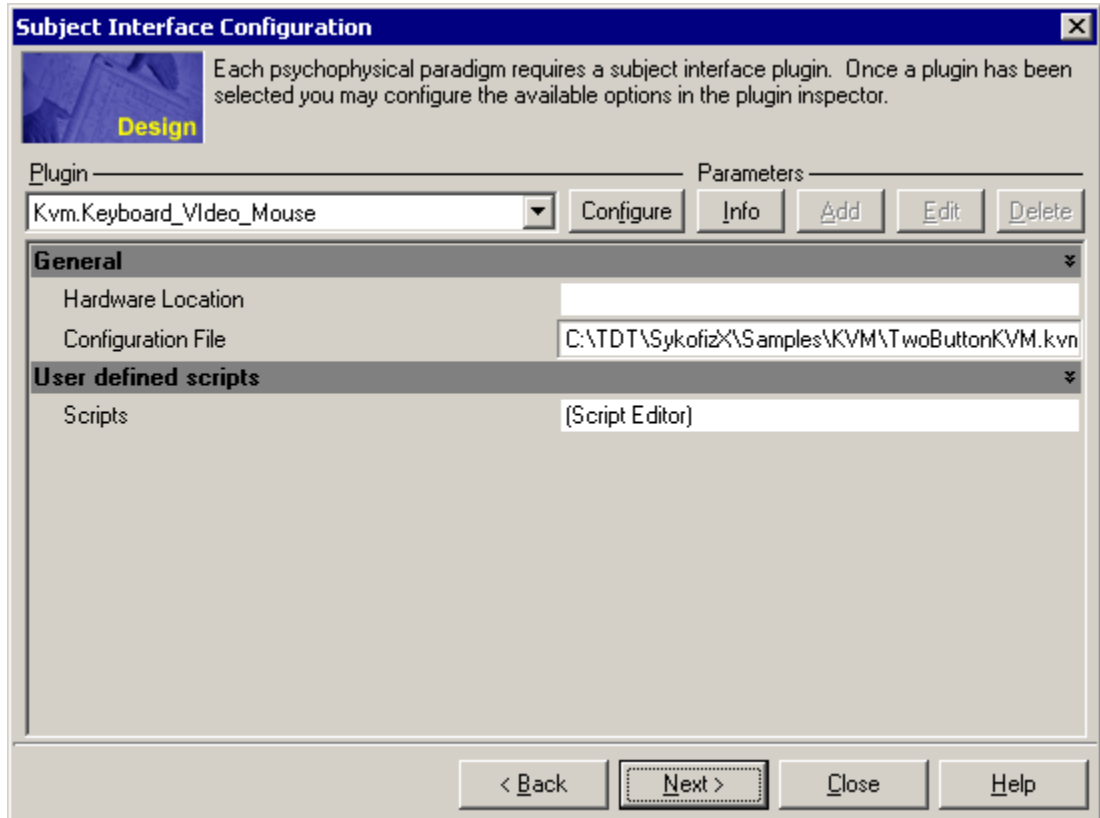
Each experiment requires that a Subject Interface plugin be chosen and configured. The subject interface controls all information that the subject either gives or receives, with the exception of the actual experimental stimulus. In standard human psychophysical experiments, that might include warning markers, interval markers, feedback, and subject responses such as button presses or changes in slider position. SykofizX provides the software required to interact with external hardware.

### Plugin

The desired Subject Interface plugin must be selected from the list of plugins available in the drop-down menu below the Plugin label.

### Parameter Inspector

Once selected, a parameter inspector is loaded into the main window. Each option available from the plugin appears in the parameter inspector and can be customized by the user. Plugins have available parameters grouped by common functionality. The parameters and parameter groups differ across plugins.

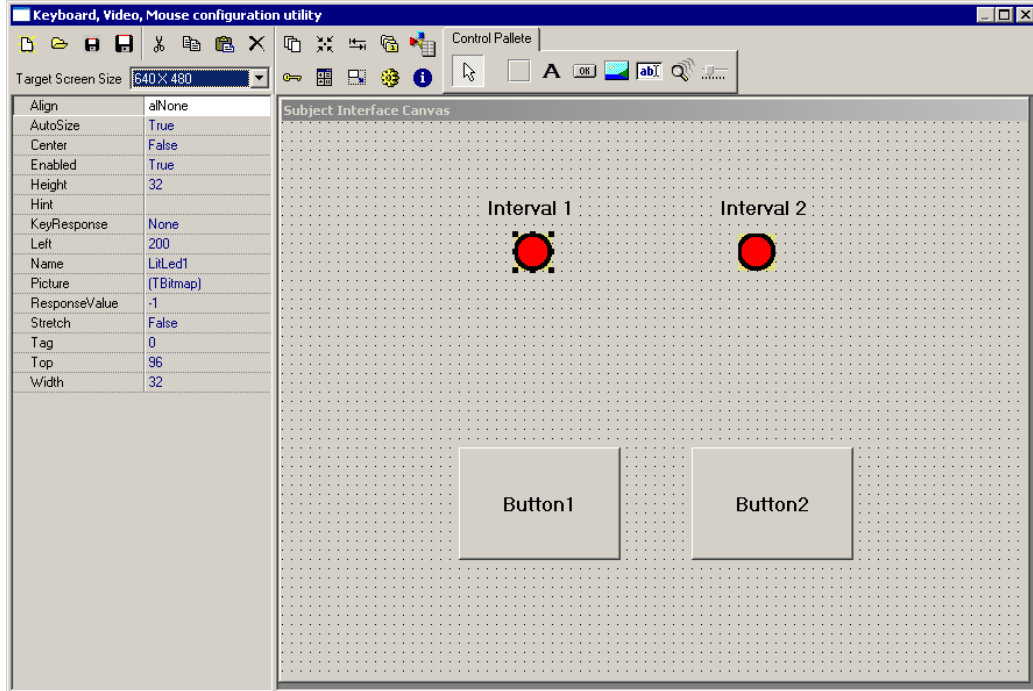


### Primary Plugin Features

There are three primary Subject Interface Plugins. An overview of the features of each primary plugin (and their derivatives) is provided here. Detailed descriptions can be found in Standard Plugins.

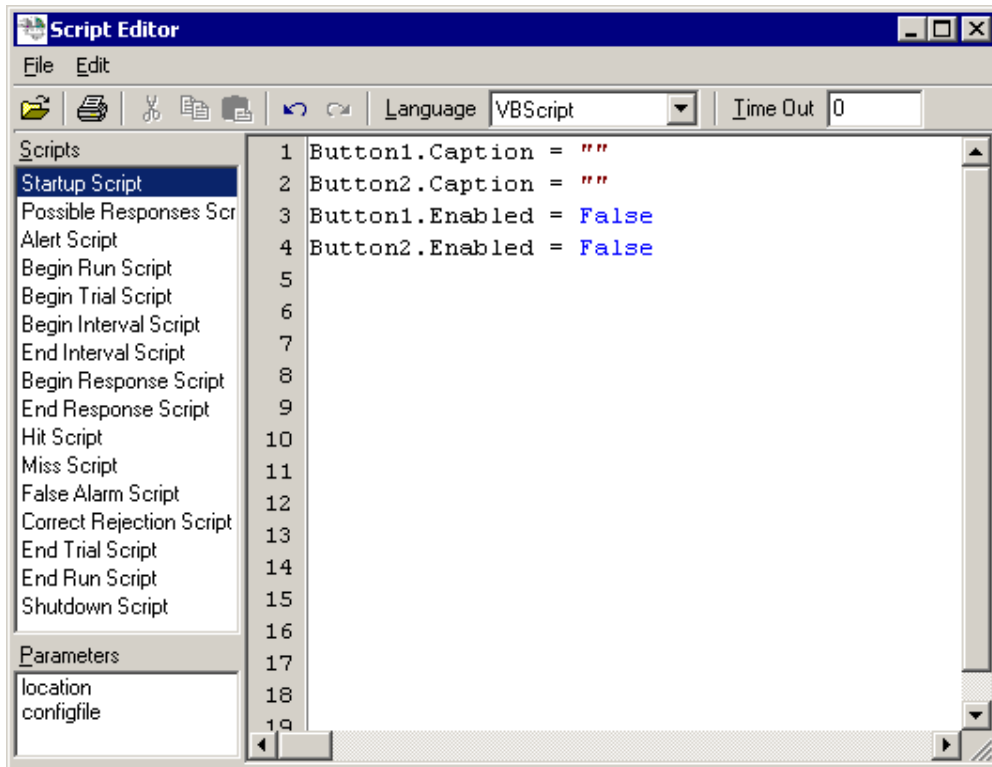
#### KVM Keyboard Video Mouse

This subject interface class is built upon the computer's keyboard, video, and mouse capabilities. To simplify the design of KVM subject interfaces, a KVM configuration utility is built into the SykofizX application. The KVM configuration utility (shown below) can be launched by selecting the Configuration parameter button from the Subject Interface Configuration wizard. Use of the KVM configuration utility is described in KVM, page 87.



### System II and System 3 Scripted Subject Interfaces

The subject interface plugins use custom scripts to interact with the System II PI2 parallel interface digital I/O lines or the System 3 device digital I/O lines. An infinite number of external hardware devices can be controlled with these TDT devices, however, many psychophysicists will use a simple button box with an LED display. In any case, the I/O lines can be controlled via user-defined scripts that may be executed at any point in time during data collection. The figure below illustrates the scripting environment.

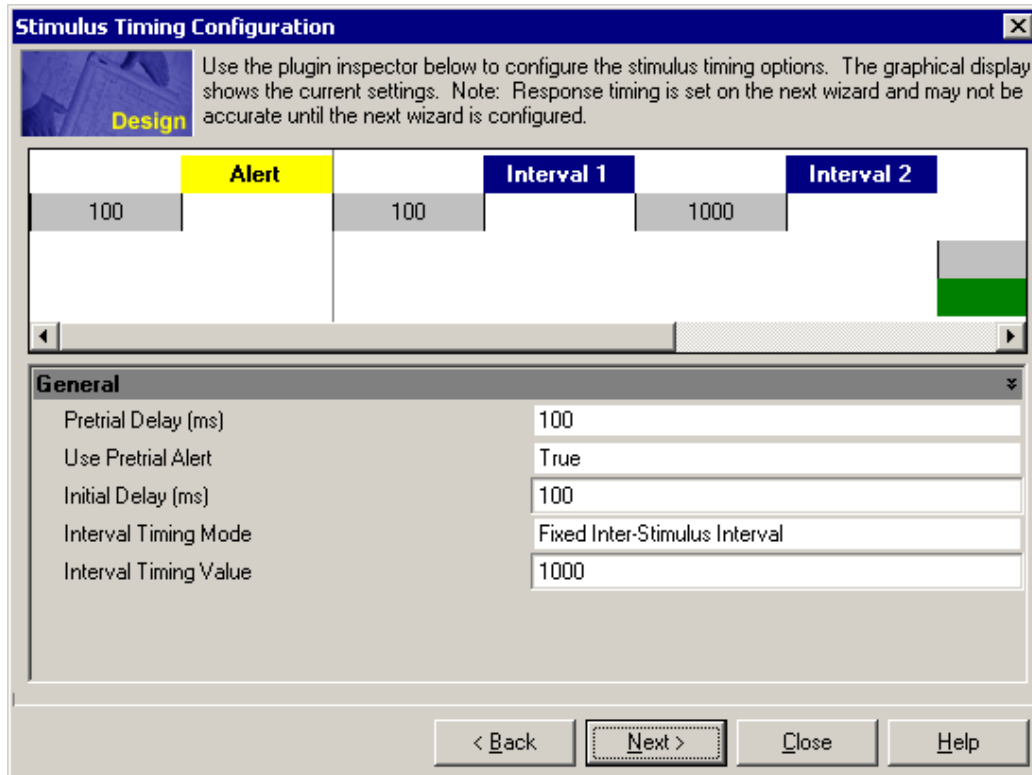


A script is just a simple sequence of commands that are executed in serial order. The SykofizX script editor makes use of the Microsoft Scripting Engine and is compatible with both the VBScript and JavaScript languages. The available scripts are shown in the panel on the left. Whenever a script is highlighted on the left, the script commands appear in the scripted editor panel on the right. Script files (.sc) may be exported from the experimental design as script libraries. A variety of sample subject interface script libraries are provided with the SykofizX application. In fact, the principal difference between the various System II scripted subject interface plugins is the specific script library associated with the plugin.

Note that a single script marks every possible event in the experimental sequence. Each script is executed at the onset of those events. Some scripts may be empty, in which case nothing will happen at those corresponding times. Also note that any parameters associated with the particular plugin are also exposed to the script editor for use in the scripts.

## Stimulus Timing Configuration

The stimulus presentation timing is controlled by the main application; however, many parameters related to stimulus timing are exposed during experimental design. The available options are listed in the parameter inspector of the Stimulus Timing Configuration wizard. At the top of that wizard is a time line depicting the trial structure and the default or current values of the timing parameters that are exposed. While the interval designator is dependent on the stimulus duration and controlled by the stimulus generation plugin, the remaining options are the same for all experimental designs, and thus are not inherently dependent on the settings in any other plugins. The horizontal scroll bar allows the user to see the entire experimental trial structure. Note that the Response Timing is set on the next wizard. The default values are displayed initially and may be modified from the next wizard.



### Pretrial Delay

This introduces a delay before the first trial in each trial series. The Pretrial Delay provides a means of setting an inter-trial interval.

**Use Pretrial Alert**

If a pretrial alert is required, then this should be set to True. Otherwise it should be set to false. There are many uses for pretrial alerts in various experimental designs. Examples might include alerting animations, sounds, or the activation of external hardware.

**Initial Delay**

Following a pretrial alert, a pretrial delay may be required to temporally separate the pretrial alert from the first observation interval. In the absence of a pretrial event, a pretrial delay could be used as a means of setting inter-trial intervals.

**Stimulus Timing Mode**

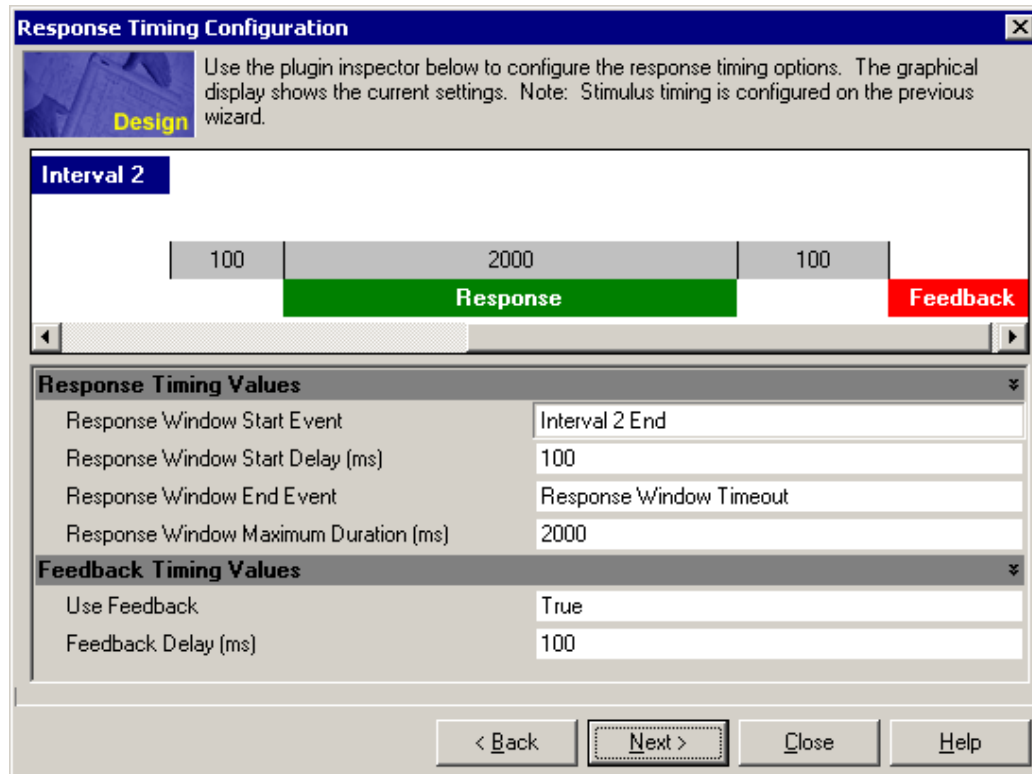
When multiple stimulus intervals are present, the stimulus timing is used to specify the time of one interval relative to another. The two choices are Fixed Inter-stimulus Intervals or Fixed Onset-to-Onset Intervals.

**Timing Value**

The desired Fixed Inter-stimulus Interval or Onset-to-Onset Interval must be set here (units of ms).

## Response Timing Configuration

The Response Timing Configuration wizard displays the available parameters used to control response timing and feedback. Similar to the Stimulus Timing Configuration wizard, a time line depicting the trial structure and the default or current values of the parameters related to response and feedback timing is shown at the top of the wizard window. These options are the same for all experimental designs, and thus are not inherently dependent on the settings in any other plugins. The horizontal scroll bar allows the user to see the entire experimental trial structure.



**Response Window Start Event**

Subject responses will only be collected during the Response Window. The response window begins with the initiation of the event specified by this parameter.



**Response Window Start Delay**

This delay determines any temporal offset from the starting event defined above to the onset of the response window.

**Response Window End Event**

Each response window must have a finite end. The end of the response window is defined relative to the Response Window End Event specified in this parameter.

**Response Window Maximum Duration**

The maximum duration of the response window is defined in ms. The response window will end as soon as a response is collected or when the maximum response duration is reached if no response has been recorded. The maximum duration serves as a "time out" window. If no time out window is desired, this parameter should be set to an unrealistically large value, such as 1,000,000,000.

**Use Feedback**

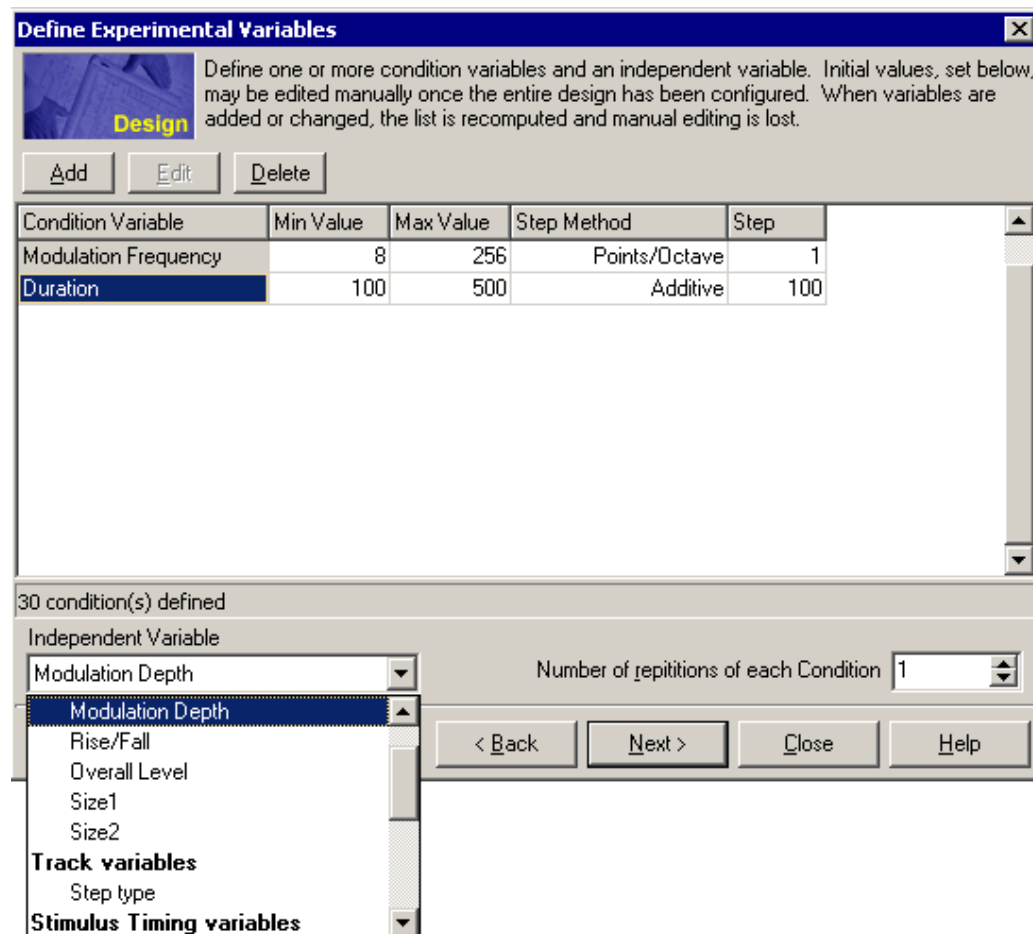
This parameter can be set to true or false, enabling or disabling the response feedback feature.

**Feedback Delay**

If feedback is enabled, then the duration of the feedback event is set by this parameter.

## Define Experimental Variables

This wizard allows the user to choose the independent variable to be manipulated and the various condition parameters used in the experiment.



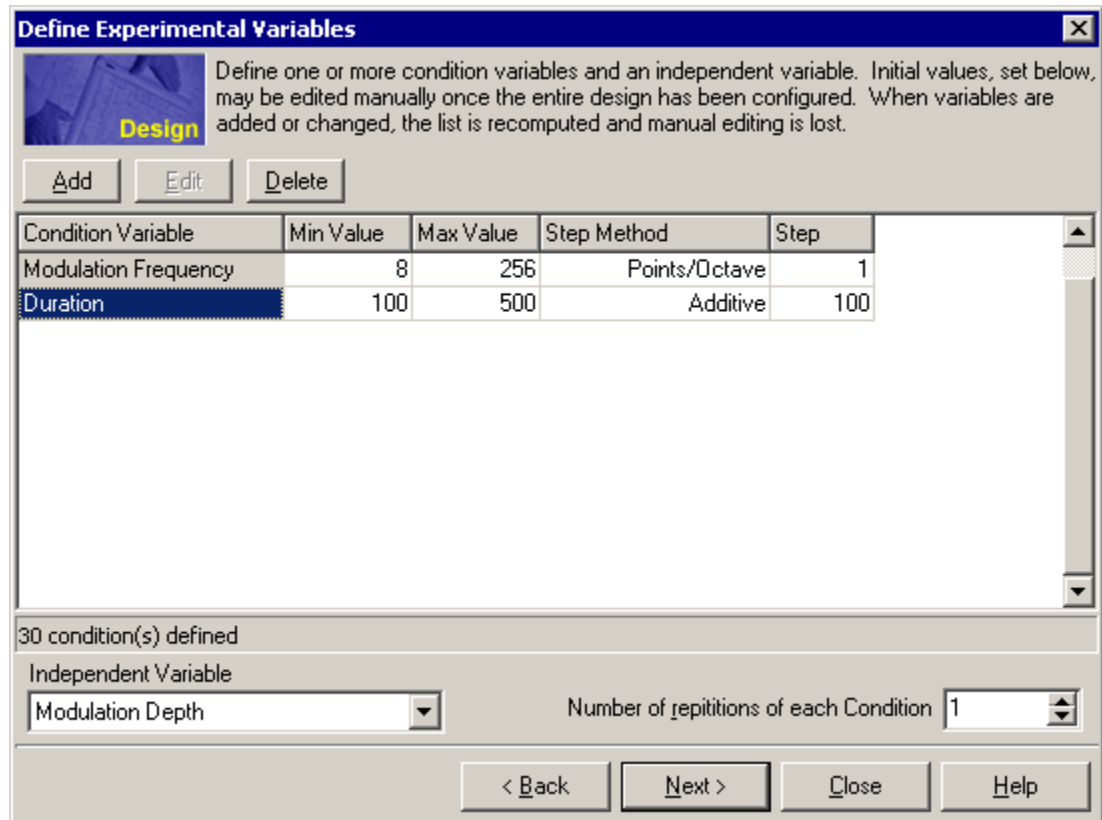
The independent variable must be selected from all of the possible independent variables. These include parameters exposed by the presentation, stimulus generation, and subject interface plugin as well as any custom parameters that may have been added and any parameters that the SykofizX application exposes.

Each experimental design consists of a number of experimental condition or dependent variables and those are configured on the Design Experimental Variables wizard as well. Each plugin specifies parameters that may serve as condition variables. In addition, user defined parameters that may serve as condition variables may be added to the Stimulus Generation Configuration, Presentation Paradigm, and the Independent Variable Configuration. Finally, a variety of timing options can serve as condition variables.

To add conditions to the list of possible conditions, select the Add button on the Define Condition Variables wizard, and then set the required parameters. In each case, the added condition variable will be chosen from the list of all possible condition variables. Next, minimum and maximum values for a particular condition variable must be chosen, followed by a step method and step size. The step method can either be additive, multiplicative, points/octave, or points/decade.

There is no limit to the number of condition variables that may be added. The number of repetitions (runs) of each condition is specified in the lower right corner of the Define Condition Variables wizard. The total number of conditions initially defined is listed in the scrollable portion of the wizard. The SykofizX application combines the condition variables specified and the values specified for each condition to form a matrix of conditions. This condition matrix may be modified once the entire design is complete.

Modifications include manually deleting one or more conditions, manually adding one or more conditions, or editing single or multiple conditions at once.

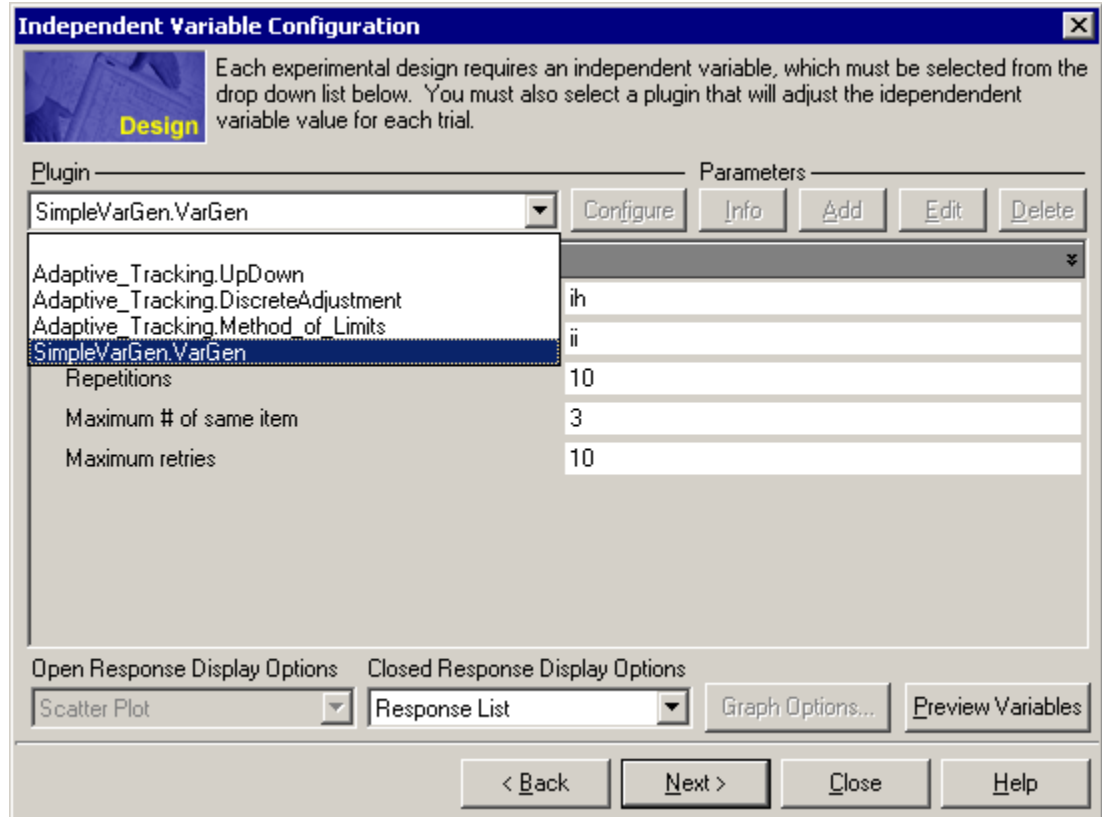


Note that Condition variables (and their parameters) and Independent Variables may be specified in any order. Furthermore, one can add and delete Condition Variables as they wish while in the design mode.

## Independent Variable Configuration

SykofizX supports four modes of independent variable manipulation that including adaptive tracking and fixed independent variables designs.

Adaptive tracking options allow the independent variable to vary during an experimental run according to a specified algorithm based on previous subject responses. The available adaptive tracking plugins are listed below. For a detailed description of each plugin, see Independent Variable, page 90.



In each case, the format for data display may be toggled between a line graph showing independent variable value versus trial number or a trial-by-trial data list that includes the independent variable value, possible response options, the actual response, etc.

### UpDown

On each trial, the independent variable changes according to the pattern of correct or incorrect responses on previous trials. At design time, the experimenter can specify a wide range of options to customize the adaptive tracking algorithm.

### Discrete Adjustment

On each trial, the subject must indicate whether the independent variable should be increased, decreased, or meets some criterion as per experimenter instructions. This plugin can be used to create both matching as well as a step-wise method of adjustment algorithms. By creating very short duration stimuli, and short inter-stimulus intervals, one can roughly approximate a continuous method of adjustment.

### Method of Limits

Similar to the UpDown tracking plugin, the Method of Limits plugin determines the independent variable value on the basis of previous subject responses. In this case, however, the direction of change (increase or decrease) of the independent variable is fixed during an experimental run. One

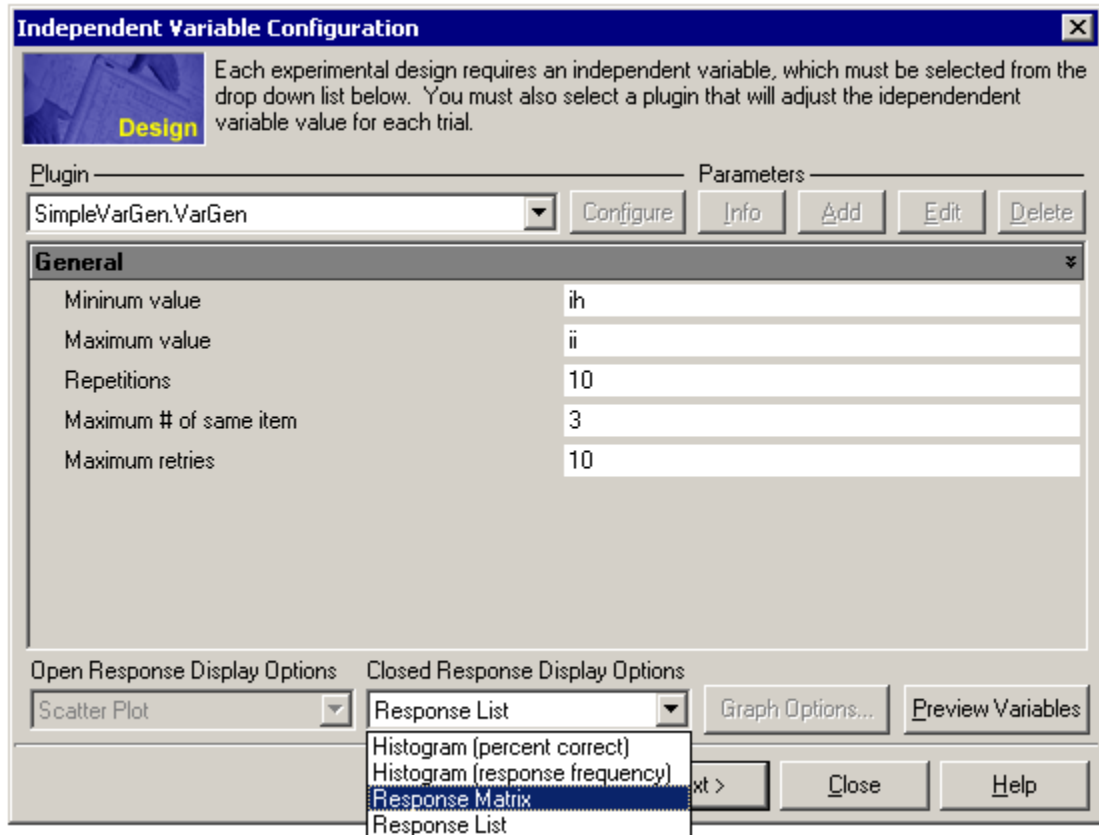
can specify ascending, descending, or alternating ascending/descending runs as well as the starting value for each of those runs.

**VarGen**

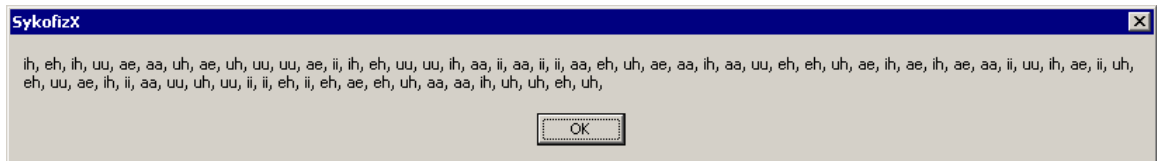
The VarGen plugin is used to generate a list of independent variable values and is used when independent variables values are to be selected from a fixed set rather than by an adaptive algorithm. The VarGen plugin is used with techniques such as the method of constant stimulus, identification, classification, and various others. The format for displaying data during data collection can be specified here as either a histogram (e.g., method of constant stimulus), a table (e.g., confusion matrix), or a list (classification or identification). In many cases, multiple display formats are available (see image below).

Default display types are set at runtime but may be changed during data collection or analysis from a window launched by pressing the right mouse button when the mouse cursor is hovering over the display itself.

Once the desired Independent Variable plugin has been selected, the parameter inspector displays the available tracking parameters and their default values.



One may also preview the fixed list of independent variables by selecting the Preview Variables button. A sample list is shown below.



Default Graph Options such as the axis labels and ranges may be modified by selecting the Graph Options button and setting the options on the subsequent display window, as shown below.

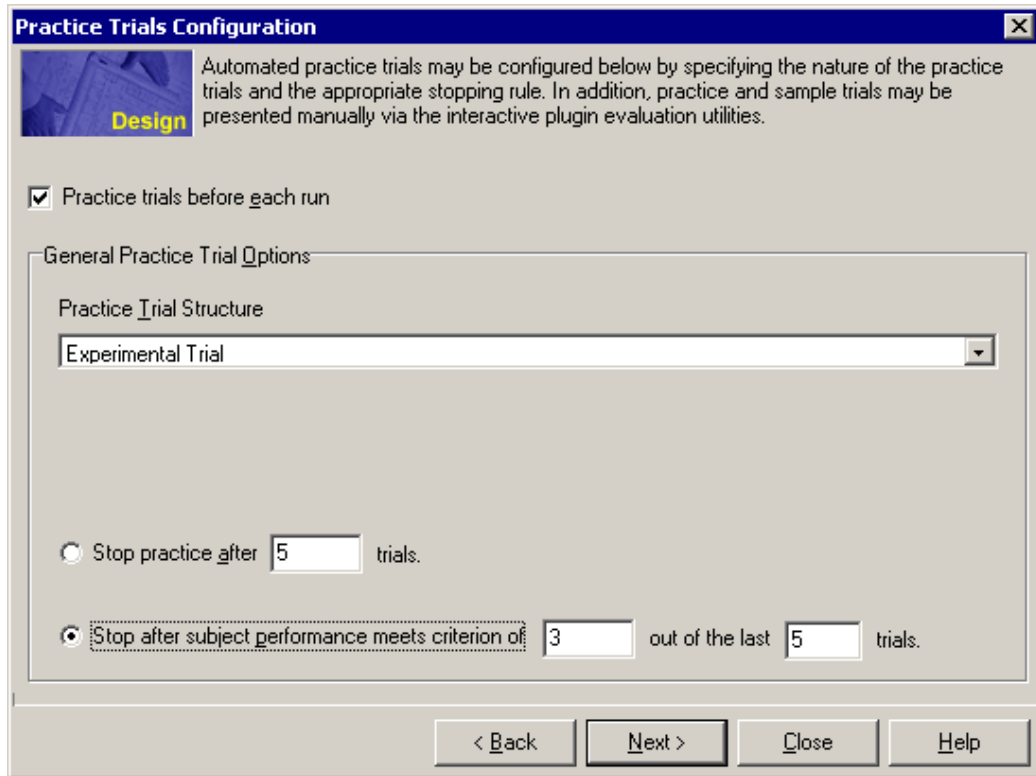
The image shows a 'Graph Options' dialog box with a title bar and a close button. It contains the following fields and controls:

- Title:** A text box containing 'Closed Response (Fixed)'.
- X Axis:**
  - Minimum: 0
  - Maximum: 6
  - Log Scaling:
  - Axis Label: 'Vowel Stimulus'
- Y Axis:**
  - Minimum: 0
  - Maximum: 100
  - Log Scaling:
  - Axis Label: 'Percent Response'
- Buttons:** 'Reset X axis', 'Reset Y axis', 'Reset Graph', 'OK', and 'Cancel'.

In some cases, there may be only one independent variable per run (block of trials), in which case a dummy variable may be specified. For example, in a Same/Different paradigm one might want to measure the percent correct same/different judgments for a single "target" value in a given run. In this case, one could create a bogus parameter and use that parameter as the independent variable to be manipulated. Because the bogus parameter is not referenced anywhere else in the design, its current, previous, or future values are irrelevant.

## Practice Trials Configuration

Practice trials are similar to experimental trials except that trial-by-trial data is not recorded or stored in the subject data file. With the Practice Trials Configuration wizard, General Practice Trial options and practice trial Stopping Rules can be defined. The general options include no practice trials, or practice trials before each run. To present practice trials occasionally, but less often than before each run, the Run-time Utilities may be used to present practice trials. The practice trial structure may be identical to the Experimental trial structure; however, alternate trial structures may be preferred. For example, in a Same-Different presentation paradigm, one may want to provide practice on AA, AB, BA, and BB trials separately. There are many other instances in which alternate trial structures may be useful during practice. The available trial structures are determined by the Presentation Paradigm plugin.

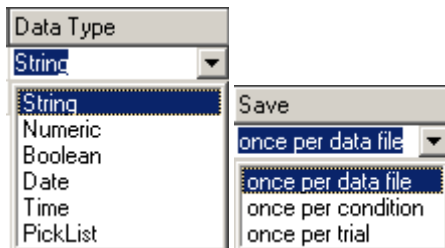


Stopping rules are used to terminate the practice trials. A fixed number of practice trials may be established by selecting the *Stop practice after \_\_\_ trials* option and setting the number of trials to be completed. Alternatively, one may require that some criterion performance be met before practice trials are stopped. The second stopping rule allows one to set the specific criterion for stopping the practice trials.

## Additional Values to Store in Data Files

Often it is useful to store additional information about the experiment or particular subject attributes along with the data collected at any given time. Additional attributes to be recorded may be specified on the Additional Attributes wizard - the final wizard encountered in design mode.

One may add, edit, or delete the additional values to be stored. Each additional value must be given an Attribute Name and the data type of that attribute must be specified. The additional values will appear during data collection and analysis when the display is in Response List mode. The additional values will be stored in the subject data file (.sfd) and are exported when the export format Response List is chosen.




Parameter	Usage, Description
Attribute Name	(String) - Name given to additional value attribute. Name is stored along with data value.
Data Type	(String) - User may specify string value of attribute at run time. (Numeric) - User may specify numeric value of attribute at run time. (Boolean) - User may specify Boolean (true/false) value of attribute at run time. (Date) - Appends current date as attribute. (Time) - Appends current time as attribute. (PickList) - Allows user to select attribute from pick list. Use "Edit" button to create and edit a list of choices.
Save	(Once Per Data File). Specifies when attributed is appended to data. (Once Per Condition). Specifies when attributed is appended to data. (Once Per Trial). Specifies when attributed is appended to data.

The Additional Values wizard also is used to specify how experimental conditions may be modified by the end user. The two options are:


- Allow experimenter to insert, update, or delete conditions in data collection mode: Some experimental designs have rigidly defined conditions, whereas for others, it may not be possible to define all of the required conditions at the beginning of the experiment. This feature allows conditions to be modified after the experimental design is completed.
- Allow experimenter to clear condition results in data collection mode. Disabling this feature maximizes data integrity.


## Condition Grid

Once the design wizards are all specified and the design wizards are closed, the experimental conditions specified on the Define Experimental and Independent Variable Configuration windows are displayed in the main Design window. At this point, conditions may be added, deleted, or edited manually.



**Adding conditions:** To add a new condition, use the **Insert Condition**  button to add a row to the condition grid. Then specify each condition parameter and move that condition to the correct location in the condition grid (if it is not already in the correct location) by selecting, dragging and dropping the row.

**Deleting conditions:** To delete an existing condition, select that condition and click the **Delete**

**Selected Condition**  button. The application prompts the user to confirm the delete operation before it is executed.

Finally, to save the new condition order, press the Freeze Order button  on the toolbar.

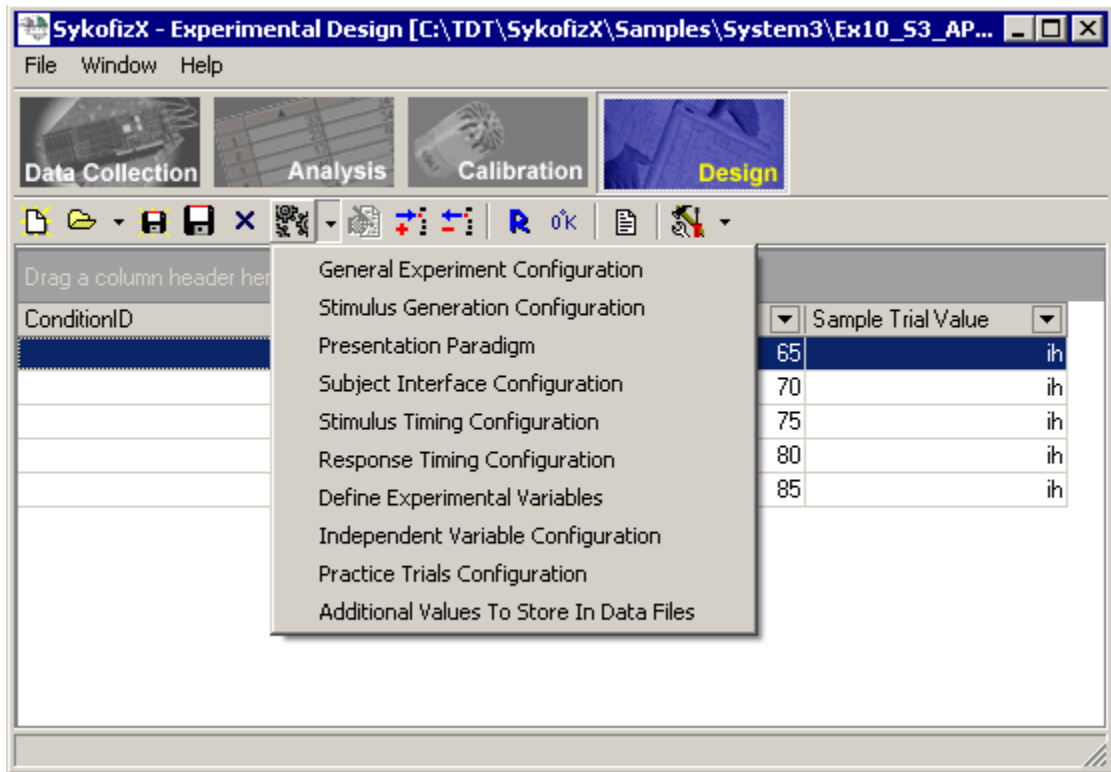
## Saving Design Files

To save a design file, use the Save  or Save As  buttons on the toolbar. It may be desirable to save the new or modified design in a particular file directory to maintain organization. Note that saving changes to an existing design will not influence subject data files that have already been associated with the original design.

Maintaining an association between the data stored in a subject data file and the actual experimental design used to collect that data is intended to maximize the integrity of subject data files. If the experimenter wishes to change the experimental design and merge data collected with two different designs, subject data files could be merged manually. In that case, create a new subject data file with the new design. Then cut the subject data from the original subject data file and paste it into the correct locations in the new subject data file. This manual editing of subject data files, and swapping experimental designs, may compromise the integrity of the subject data file.

## Evaluation Utilities

From the Design Mode, the user can quickly access individual design wizards by selecting the configure button on the toolbar and choosing a wizard from the drop-down list. Note that some wizards depend on the specification of previous wizards. Therefore, if all wizards have not been defined, it is possible that some wizards will not be available from the drop-down list.



Also available from Design mode are the equivalent several Evaluation Utilities. These allow the experimenter to test the stimulus generation, stimulus presentation, experimental trial structure, subject interface, and other plugin-related behavior while in the design mode. This is particularly useful for evaluating any custom scripting that the experimenter is developing. See Evaluation Utilities, page 19 for more information.



## Custom Scripting

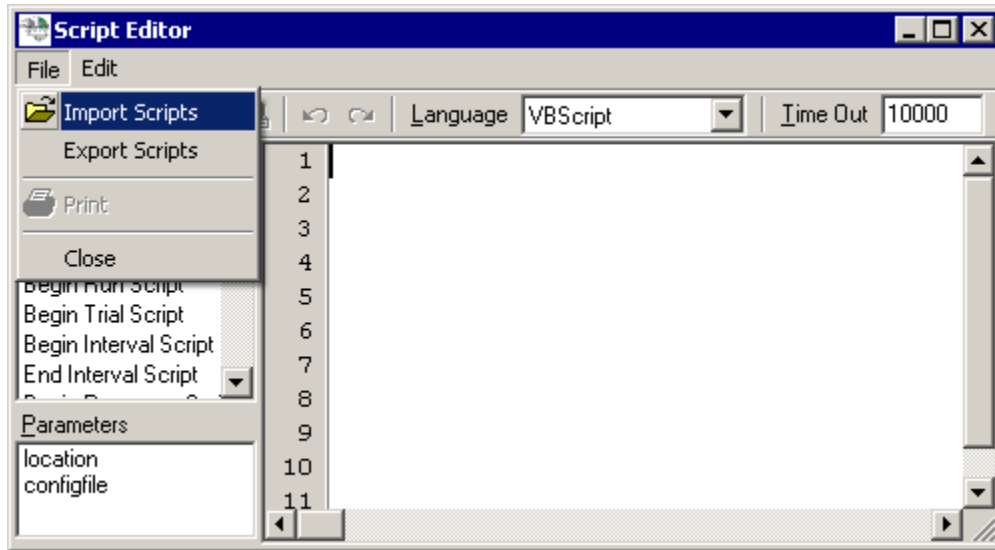
Apart from the modular design of the SykofizX application, the single feature that makes the application so powerful and flexible is the ability to use custom scripts to control *stimulus generation* and the *subject interface*. A script is a simple set of commands executed by the SykofizX application each time it is called by the application. The scripts make use of the commands made available by the SykofizX application, the user-defined parameters that are available to the scripts, and the native capabilities of the scripting language (VBScript or JavaScript). Like the plugin architecture, custom scripts can be mixed and matched to create novel experimental designs. Furthermore, many scripts designed for System II hardware and software function with System 3 hardware and software, and visa versa, without modification. Associated with each stimulus generation plugin (except the DemoStimulus plugin) is a set of scripts known as a script library. Likewise, a set of scripts or script library is associated with the subject interface plugin.

While users need to be somewhat familiar with either VBScript or JavaScript, users with minimal programming experience should find both scripting languages easy to learn and use. The major differences between scripting languages is the required syntax. Users can often master the syntax of a new scripting language simply by applying what they know about programming and examining sample scripts in the language of interest. The sample scripts provided with this application will go along way towards familiarizing new users of VBScript. VBScript guides found on the internet or in trade texts also can serve as very helpful references. An introduction to scripting with respect to stimulus generation and subject interface plugins and TDT System II and 3 hardware follows.

## Saving Custom Scripts

Each scripts library associated with an experimental design is stored in the Experimental Design File (.sfx). Furthermore, since a copy of the experimental design is stored within all Subject Data Files (.sfd), a copy of the relevant script libraries will be stored in each data file. Thus, one may access the scripts from either a design or a data file. It is never necessary to save a script or script library, since modifying and saving an experimental design will save any changes to the associated scripts.

However, it may be convenient to export a script library to a separate file. For convenience, the SykofizX script editor does include export and import features. Thus, a script library can be exported to an independent file with the .sc extension. Likewise, those libraries can be imported into a novel experimental design, avoiding the need to code a set of scripts from scratch or to cut and paste scripts from another design. Each time a script is modified, a prompt reminds the user that the scripts have been modified and asks the experimenter whether or not the scripts should be exported to a separate file. Note that the exported script libraries (.sc files) are not used by the SykofizX application, but provide a convenient way to exchange files among experimental designs as well as users.



## Stimulus Generation Scripts

### Stimulus Generation Scripts

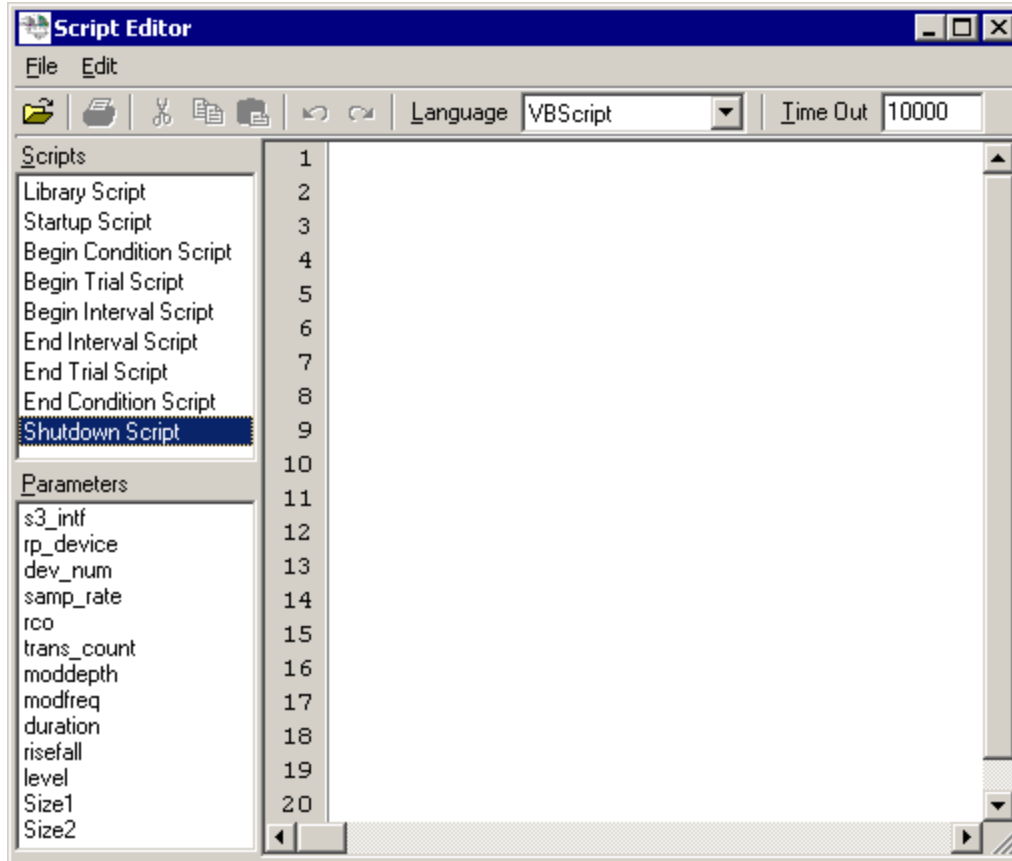
Scripting may be used to assist in stimulus generation in three ways.

- Using System II hardware, a script may specify APOS commands and the scripting logic necessary to generate an unlimited number of stimuli using the AP2 Array Processor. The script may also include commands to other System II hardware devices such as the PA4 programmable attenuator and the PI2 parallel interface.
- Using System 3 hardware, a script may provide a custom interface between the SykofizX application and parameter tags specified in an RCO file created in the RPvdsEx application. The script may also include commands to other System 3 hardware devices such as the PA5 programmable attenuator and the digital I/O.
- Using System 3 hardware, a script may specify APOS commands and the scripting logic necessary to generate an unlimited number of stimuli using the Array Processor Emulator (APE). The APE executes APOS function calls on the PC CPU and passes the final data buffer to a System 3 device. The script may also include commands to other System 3 hardware devices such as the PA5 programmable attenuator and digital I/O.

For information on the AP2 function calls supported by the APE and their usage see System II AP2 APOS Interface, page 72.

Individual scripts making up the stimulus generation scripting library are given names and are executed at times corresponding to events within a typical psychophysical experiment. As shown in the figure below, the nine possible scripts are listed in the top left panel of the script window under the heading *Scripts*.

Each script is executed at the appropriate time in the course of running an experiment. The execution of all scripts is obligatory; however, scripts may contain no commands, in which case, no commands are executed. In addition to eight event markers, a Library script allows users to write functions and procedures and to make declarations that can be executed by any other script, greatly simplifying some more repetitive scripts.



**Library Script:** Constants, variables, functions, and procedures declared in the Library script are available for use by any of the other scripts and are thus considered *global* declarations.

**Startup Script:** Executed each time a file is loaded in the data collection mode.

**Begin Condition Script:** Executed each time a new experimental condition is initiated.

**Begin Trial Script:** Executed each time a new experimental trial is initiated.

**Begin Interval Script:** Executed each time a new presentation/observation interval is initiated.

**End Interval Script:** Executed each time a presentation/observation interval is terminated.

**End Trial Script:** Executed each time an experimental trial is terminated.

**End Condition Script:** Executed each time an experimental condition is completed.

**Shutdown Script:** Executed each time data collection is terminated.

## System II Stimulus Generation Scripts

SykofizX scripting supports AP2 APOS function calls as well as XBus function calls for a variety of System II hardware devices. The following documentation assumes familiarity with System II hardware and the associated software function calls.

**XBus device support:** DAx, ADx, PD1x (A/D and D/A commands only), WG1/WG2, PA4, PI2, and SW1 hardware devices. The function calls corresponding to each of these devices may be found in the TDT System II manual.

**AP2 APOS support:** Most APOS function calls are supported by SykofizX. The usage of each of these calls can be found in the TDT APOS User's Guide.

**Note:** APOS calls that previously required specification of the sample period no longer require this parameter. SykofizX always knows the current sample period and corresponding sample frequency, and therefore automatically supplies it when needed.

The user is responsible for buffer management, including allotting DAMA buffers, pushing to, popping from, and clearing the AP2 memory stack, with the exception of the `qpopbuffer16` command.

**AP2 Locking:** The `SystemII.ScriptedStimulus` plugin automatically handles AP2 locking and unlocking for the user.

**Application Defined Functions and Parameters (Specific to System II):** A list and description of new System II functions and modifications of several previous functions is provided in the Standard Plugin section on page 69.

**AP2 APOS Function Calls Support:** With the exception of AP2 macros and plotting routines, all APOS function calls are supported by SykofizX. A list of supported calls is provided in the Standard Plugin section on page 69.

### System 3 Stimulus Generation Scripts

SykofizX scripting allows users to create an interface between application parameters and parameter tags defined in an RCO file, to communicate with various ZBus devices, and to specify APOS function calls to be interpreted by the Array Processor Emulator (APE). Example 5 (page 110) illustrates the use of scripts to interact with a user-defined RCO file using System 3 hardware. Examples 1 through 4 and 6 through 11 illustrate the use of scripts to interact with the APE.rco file using System 3 hardware.

**ZBus device support:** All RPx devices (RP2, RM1, RM2, RX5, RL2, RA4/16, RA8, RV8) including their digital I/O features, as well as the PA5 programmable attenuator. The function calls corresponding to each of these devices may be found in the TDT System 3 Manual and the RPvdsEx Manual.

#### Application Defined Functions and Parameters (Specific to System 3)

**System 3:** Boolean - Returns true if using System 3 hardware, false otherwise.

**qpopTag TagName:** String, **SizeTag:** String - Pops stack to a PC buffer and sends buffer to RCO file tag TagName. For APE.rco, TagName is either "Channel1" or "Channel2". SizeTag is a string representing the number of points in the stimulus buffer.

### Additional Commands Available for Use in Stimulus Generation Scripts

#### User Defined Parameters

Each stimulus generation plugin allows the user to add and define custom parameters to the parameter inspector. Typically these parameters are used during stimulus generation and they may be called from any of the stimulus generation scripts. The value of those parameters are either specified when the parameter is defined or the value is set by the application in the course of an experiment. For example, the user defined parameter "modfreq" in Example 1 (see Sample Designs, page 105) is used as a "Condition Variable" whereas the parameter "moddepth" is used as the "Independent Variable". The Experimental Variable Configuration wizard allows a parameter to be chosen either as a Condition Variable or the Independent Variable.

#### Application Defined Functions and Parameters

Below is a list of the documented functions and parameters that are made available by the application for use in scripts.

**StimulusClass:** String - Returns one of two possible stimulus classes (SIGNAL or STANDARD) specified by the presentation paradigm.

**ApplicationPath:** String - Returns the current application path to be used when referencing files located within that path.

**SampleFreq:** Float - Returns the current sample frequency in Hz.

**SamplePeriod:** Float - Returns the current sample frequency in ms.

**Sleep:** (ms integer) - Procedure pauses the stimulus timing thread specified duration in ms. Relies on Microsoft precision timer for accuracy < 1 ms.

**GetScalar:** (level: float, rms: float, transducernumber: integer, lowfreq: float, highfreq: float) - Using the calibration file specified in the stimulus generation plugin parameter inspector, GetScalar returns the real value necessary to scale a stimulus with the specified RMS value to achieve the specified overall level in dB. If lowfreq and highfreq are equal, GetScalar considers the transducer output at only that frequency. If a range of frequencies is specified by setting lowfreq to a value less than highfreq, GetScalar returns a real value based on the average transducer output across that range. GetScalar uses a cubic spline interpolation to determine the transducer output at frequencies not recorded in the calibration file. When lowfreq or highfreq is outside the range of frequencies recorded in the calibration file, frequencies outside that range are set to the nearest actual value recorded. If the correct calibration file (.bcf) is not specified, the application raises a "GetScalar" related error.

**GetAttenuation:** (level: float, rms: float, transducernumber: integer, lowfreq: float, highfreq: float) - Using the calibration file specified in the stimulus generation plugin parameter inspector, GetAttenuation returns the dB value necessary to attenuate a stimulus with the specified RMS value to achieve the specified overall level in dB. If lowfreq and highfreq are equal, GetAttenuation considers the transducer output at only that frequency. If a range of frequencies is specified by setting lowfreq to a value less than highfreq, GetAttenuation returns a real value based on the average transducer output across that range. GetAttenuation uses a cubic spline interpolation to determine the transducer output at frequencies not recorded in the calibration file. When lowfreq or highfreq is outside the range of frequencies recorded in the calibration file, frequencies outside that range are set to the nearest actual value recorded. If the correct calibration file (.bcf) is not specified, the application raises a "GetAttenuation" related error.

**Transducer.Item(Number).Name:** (Number: String) - Returns the Name of the transducer specified by Number as indicated in the calibration file. For example, if transducer 2 in a given calibration file is "ER-2 (SN 14001)" then the function would return "ER-2 (SN 14001)".

**Attenuator(Number).Attenuation:** (Number: Integer) - Sets programmable attenuator indexed "number" to specified value. Used with SystemII.ScriptedStimulus plugin. Example: Attenuator(0).Attenuation = 10.3.

**PA5x:** (ActiveX control) - See the ActiveX Manual for information about using ActiveX controls. Used with System3.RcoFile plugin. Example: PA5x.SetAtten 10.3.

**RPcoX:** (ActiveX control) - See the ActiveX Manual for information about using ActiveX controls. Used with System3.RcoFile plugin.

**Qpopbufferf:** (DamaBufferName: String) - Allocates DAMA buffer with index DamaBufferName and writes data on memory stack to DAMA buffer.

**qpopbuffer16:** (DamaBufferName: String, Channel: Integer) - Allocates DAMA buffer with index DamaBufferName and writes data on memory stack to DAMA buffer. Prepares for conversion via specified D/A Channel.

**Dama:** (DamaBufferName: String) - Reads data from specified DAMA buffer and pushes the buffer onto the memory stack.

**Uallot:** Implements APOS command \_Allot.

**Ufir:** Implements APOS command \_fir.

**Uiir:** Implements APOS command \_iir.

**Udeallot:** Implements APOS command \_deallot

## Sample Stimulus Generation Scripts

Below are the two stimulus generation scripts comprising the script library used in Example 1. Only the Library and Begin Interval scripts include commands to be executed. In the library script, note commands used to specify constants and the function used to create a random phase tone.

```

Script Editor
File Edit
Language VBScript Time Out 10000
Scripts
Library Script
Startup Script
Begin Condition Script
Begin Trial Script
Begin Interval Script
End Interval Script
End Trial Script
End Condition Script
Shutdown Script
Parameters
s3_intf
rp_device
dev_num
samp_rate
rco
trans_count
moddepth
modfreq
duration
risefall
level
Size1
Size2
22 Dim energy
23 buffersize = topsize
24 scale MAX_DIGITAL_AMP/maxmag
25 qdup
26 shift -1*average
27 square
28 energy = sum
29 RMS = sqr(energy/buffersize)
30 drop
31 End Function
32
33 Function RandPhaseTone(freq) 'Function creates a random phase tone
34 Dim sphase
35 Dim slope
36 randomize
37 sphase = rnd * 2 * pi
38 slope=2*pi*freq*SamplePeriod/1000000
39 fill sphase,slope
40 sine
41 End Function
42
43

```

In the Begin Interval Script, note the use of a statement conditional on the parameter StimulusClass. Four user-defined parameters include duration, modfreq, moddepth, and risefall. The default values for these parameters are set in the plugin's object inspector. Those values may also be specified as condition variables or as the independent variable. Also note the conditional statement used to determine the current hardware being used (System II or System 3) and the appropriate qpop command. By using this statement, the script may be used with System II or System 3 hardware without any modifications. Finally, note that a buffer of zeros is sent to channel two, since the APE.rco file requires that two parameter tags be specified.

```

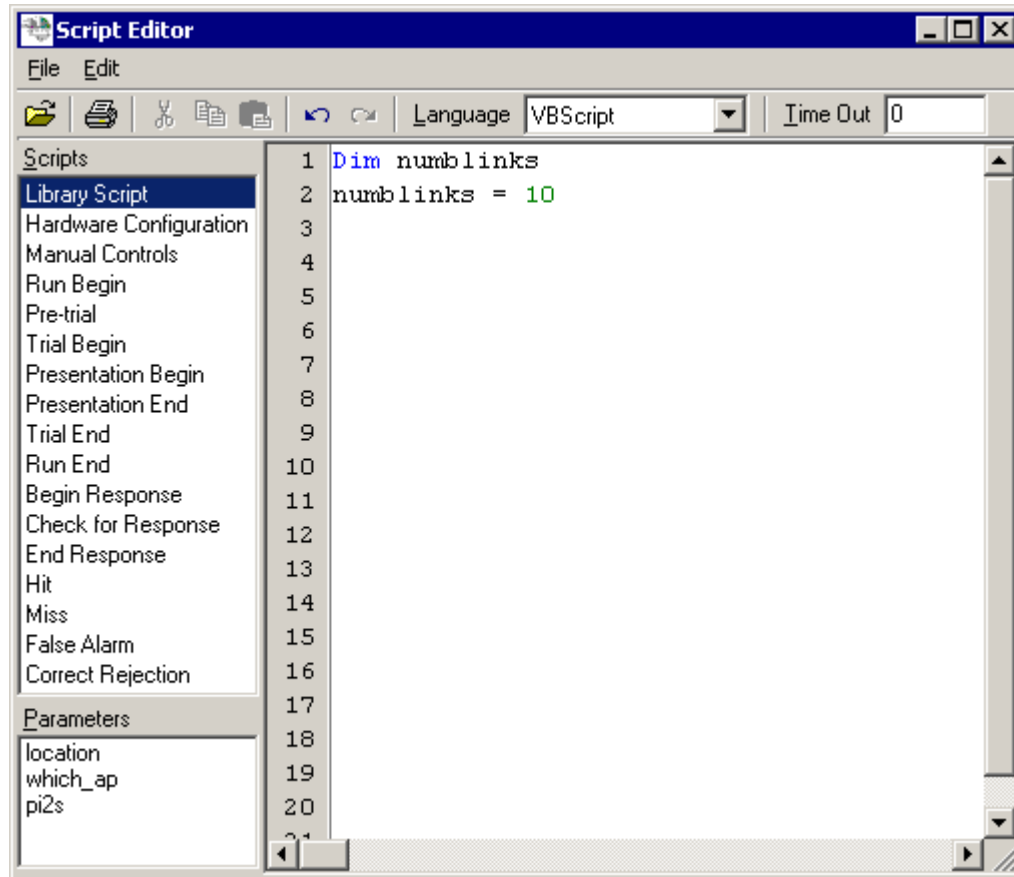
Script Editor
File Edit
Language VBScript Time Out 10000
Scripts
Begin Condition Script
Begin Trial Script
Begin Interval Script
End Interval Script
End Trial Script
Parameters
s3_intf
rp_device
dev_num
samp_rate
rco
trans_count
moddepth
modfreq
duration
risefall
level
Size1
Size2
7 'CREATE CHANNEL 1 BUFFER
8 dpushms duration 'Allocate memory on stack - uses SPER to d
9 RandPhaseTone(modfreq) 'calls function defined in Library script,
10 scaleval = 10^(moddepth/20) 'compute scale value based on user-defined
11 if StimulusClass = "SIGNAL" then 'computes signal or standard by scaling mo
12 scale scaleval
13 else
14 scale 0.0
15 end if
16 shift 1.0 'DC shift of modulator
17 dpushms duration 'allocate memory on stack - uses SPER to d
18 gauss 'fill with samples from Gaussian distribut:
19 mult 'Multiply modulator and carrier
20 qwind risefall 'Applies risefall specified as user-defin
21
22 'SCALE TO SPECIFIED LEVEL

```

## Subject Interface Scripts

### Subject Interface Scripts

Subject Interface scripts allow experimenters to customize interaction between the SykofizX application and the subject interface hardware. There are three different Subject Interface plugins and each one has an associated script library. Subject Interface scripts are defined and used in a manner similar to the Stimulus Generation scripts. The scripts make use of a host of commands made available by the SykofizX application as well as user-defined parameters and the native capabilities of the scripting language (VBScript or JavaScript). Custom scripts may be mixed and matched to create novel experimental designs. Associated with each stimulus generation plugin (except the DemoStimulus plugin) is a set of scripts known as a script library, as shown below.



For other general information about scripts, see Custom Scripting and Saving Custom Scripts on page 57. The goal of the subject interface script library is to specify the behavior of the subject interface conditional on the events in the experimental sequence. There are a total of 17 possible subject interface scripts comprising a single script library. Each of these scripts corresponds to a specific event in a typical experimental trial. In most cases, commands will be defined only for a few of these scripts, the others remaining blank. Design of the subject interface is most closely related to the presentation paradigm.

**Example:** Consider the subject interface scripts required for a simple two-interval, forced-choice presentation paradigm without feedback and a simple two-button box with two LEDs for the subject interface. One might define events within 4 of the 17 scripts as follows.

**Begin Run Script:** Flash the LEDs as a warning signal that the trial is about to begin. Disable the response buttons.

**Begin Interval Script:** Turn on LED corresponding to current presentation/observation interval to mark that interval.

**End Interval Script:** Turn off LED corresponding to current presentation/observation interval.

**Begin Response Script:** Enable the response buttons and scan until an appropriate input is provided by the subject.

The Subject Interface scripts and corresponding plugins are designed to function either with the digital I/O lines available from System II (e.g., PI2) and System 3 (e.g., RPx) hardware or with a Keyboard-Video-Mouse (KVM) interface. Across laboratories and individual scientists, specific subject interface hardware and event behavior can differ widely. For a given laboratory or scientist, that behavior typically will not differ widely. Thus, once a general subject interface is designed, it can be used over and over again, often with only a few, if any, modifications.

For studies of human auditory perception, the two most common subject interfaces are a simple button box with LEDs and a KVM display (using one or a combination of the keyboard, video, and mouse devices). For speech perception, the KVM interface is used most often. For animal psychophysics, the actual subject interface hardware varies greatly from experiment to experiment. In most cases, however, digital I/O lines under application control may be used to control the subject interface hardware, such as lights, buttons, stimulators, feeders, etc.

The available subject interface plugins are listed below along with examples of scripts for each. These plugins are described in detail in the Standard Plugins section of the User's Guide.

There are six potential sources of commands that may be used in Subject Interface scripts:

1. SykofizX application
2. User defined within script
3. System 3 RPx digital I/O commands
4. System II XBus/PI2 commands
5. Objects associated with a KVM configuration file
6. Native (VBScript or JavaScript) scripting commands

The commands available from the SykofizX application are listed below.

**Interval Integer:** Value of current presentation interval.

**ResponseValue Integer:** Value of current subject response.

**CorrectResponse Integer:** Value of current expected (correct) response as determined by the presentation plugin on an interval-by-interval basis.

**OpenResponseValue String, Float, or Integer:** Value of current subject response.

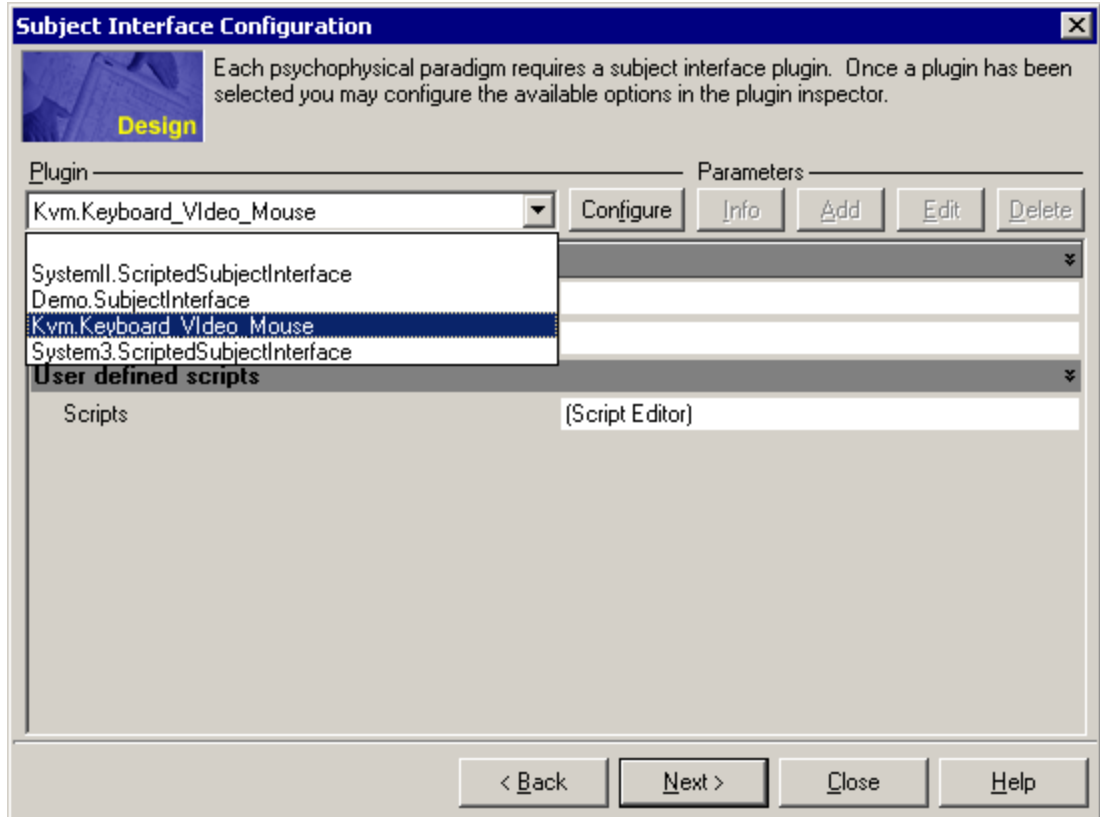
**PossibleResponses.count Integer:** Number of possible responses.

**PossibleResponses.Item(#).Name String:** Returns the name of the possible response of the specified item (0 to n-1 items). Possible response names are specified in design wizard memo (e.g., foils for CSIL) or parameter inspector (e.g., Forced Choice or Same/Different).

**PossibleResponses.Item(#).Value Value:** Returns the value of the possible responses of the specified item (0 to n-1 items). Values are determined by order of items in the design wizard memo list (e.g., CSIL) or by the parameter inspector (e.g., Forced Choice or Same/Different).

The sample scripts shown below and those provided with various sample designs illustrate how to combine these commands to achieve the desired subject interface.



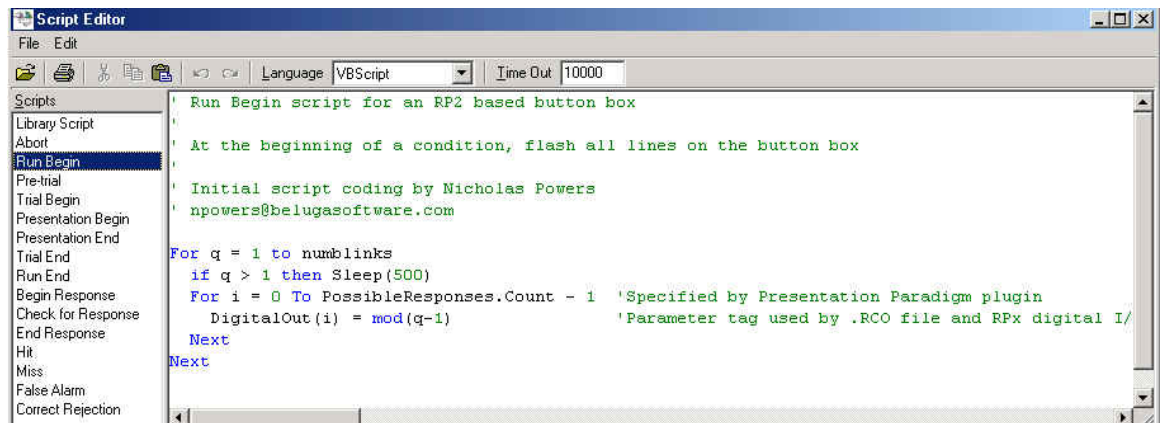


## System 3 Subject Interface Scripts

**Plugin:** System3.ScriptedSubjectInterface

The System3.ScriptedSubjectInterface plugin allows users to write scripts that control System 3 hardware devices, which in turn serve as or control the subject interface device(s). The most common use of this plugin/script combination is to control a simple button box with LEDs.

The sample script below is executed at the beginning of each experimental run. The script relies on the DigitalOut(i) parameter tag, specified in the associated RCO file, to communicate with the digital I/O lines on the RPx and RMx devices. The PossibleResponses.Count function, made available to the scripts by the SykofizX application, returns the number of possible responses, as determined by the Presentation Paradigm plugin. The constant Numblinks was defined in the Library Script.



The Check for Response script below illustrates the loop used to poll the RPx digital input lines via the DigitalIn(i) RCO parameter tag to determine when a response is recorded and passes that response to the application using the application command ResponseValue.

The screenshot shows a 'Script Editor' window with a menu bar (File, Edit), a toolbar, and a 'Language' dropdown set to 'VBScript'. The 'Time Out' is set to '10000'. A 'Scripts' list on the left includes 'Check for Response', which is selected. The main text area contains the following code:

```

Run Begin script for an RP2 based button box

' read all the input lines for a response. Set ResponseValue when a response is detected

' Initial script coding by Nicholas Powers

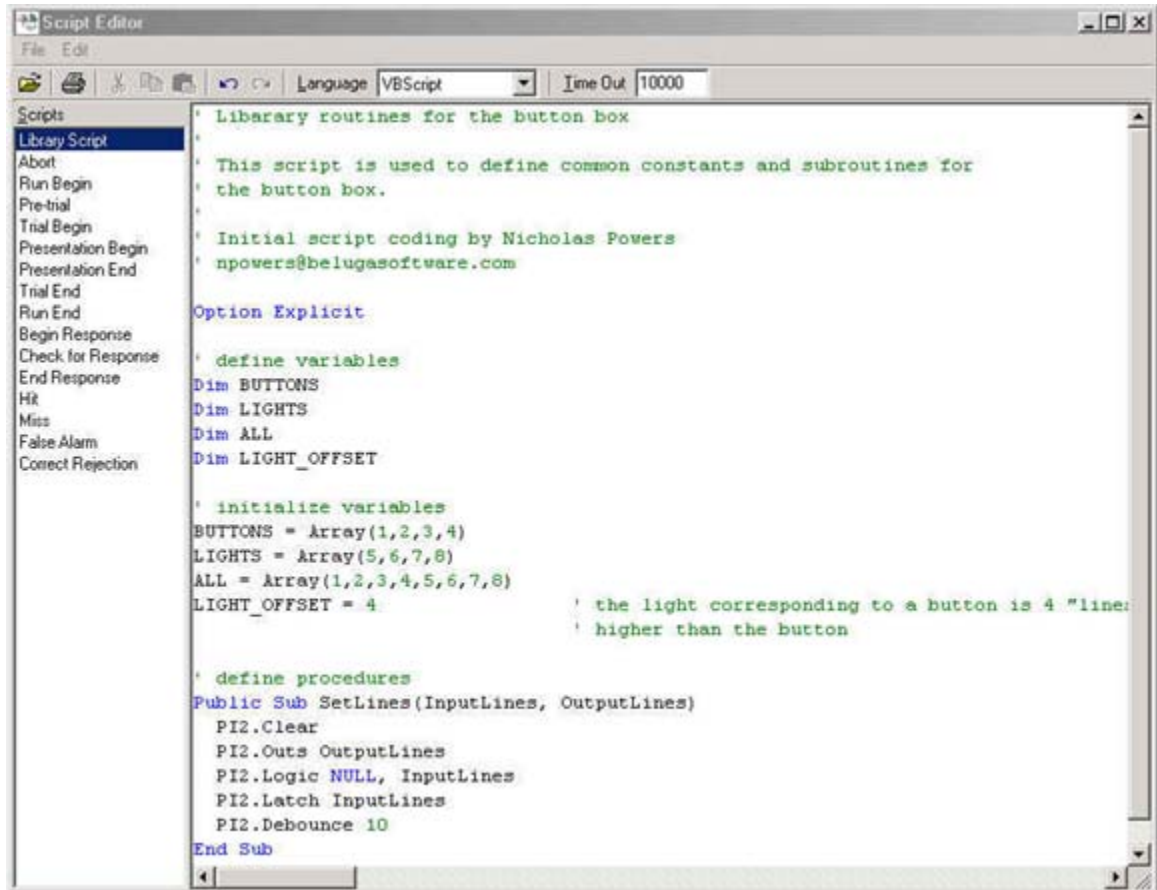
For i = 0 To PossibleResponses.Count - 1 'Determined by the Presentation Paradigm plugin
  If DigitalIn(i) = False Then 'Parameter tag used by .RCO file and RPx digital I/O
    ResponseValue = i + 1 'Returns "ResponseValue" to application
  Exit For
End If
Next
    
```

Although the scripts used to control a button box with LEDs are relatively simple, one may require much more elaborate scripts to control a variety of external hardware devices, as often encountered in an animal behavior lab. Scripts might be used to trigger events associated with trial initiation, or even measure behavior that might lead to the initiation of a trial. Other scripts might be used to trigger events associated with stimulus presentation, and still others to measure responses and to provide positive or negative feedback.

## System II Subject Interface Scripts

**Plugin:** SystemII.ScriptedSubjectInterface

System II subject interface scripts are very similar to those used with the System 3 hardware. They typically rely on the PI2 parallel interface. Because control of the PI2 digital I/O is a bit more elaborate than control of the RPx digital I/O, System II scripts may require a bit more structure at the outset. The script below defines common constants and subroutines for the PI2/button box interface. This makes scripting specific PI2/button box events in subsequent scripts much easier to code and to read.



The screenshot shows the Script Editor window with the following content:

```

Script Editor
File Edit
Language VBScript Time Out 10000

Scripts
Library Script
Abort
Run Begin
Pre-trial
Trial Begin
Presentation Begin
Presentation End
Trial End
Run End
Begin Response
Check for Response
End Response
Hit
Miss
False Alarm
Correct Rejection

' Library routines for the button box
'
' This script is used to define common constants and subroutines for
' the button box.
'
' Initial script coding by Nicholas Powers
' npowers@belugasoftware.com

Option Explicit

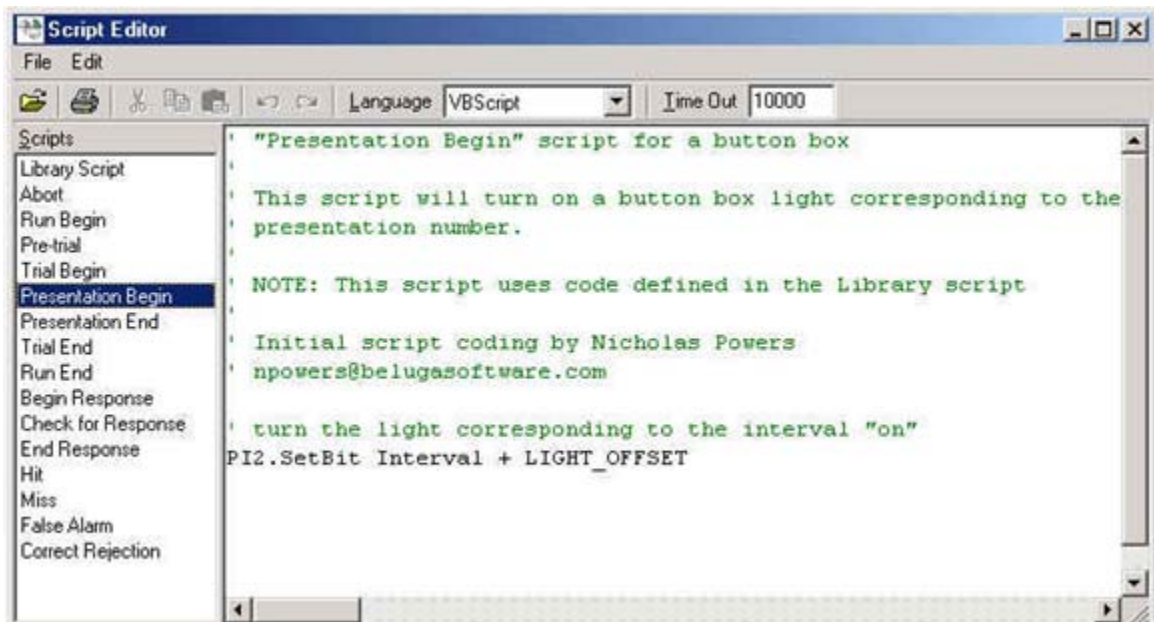
' define variables
Dim BUTTONS
Dim LIGHTS
Dim ALL
Dim LIGHT_OFFSET

' initialize variables
BUTTONS = Array(1,2,3,4)
LIGHTS = Array(5,6,7,8)
ALL = Array(1,2,3,4,5,6,7,8)
LIGHT_OFFSET = 4           ' the light corresponding to a button is 4 "lines
                           ' higher than the button

' define procedures
Public Sub SetLines(InputLines, OutputLines)
    PI2.Clear
    PI2.Outs OutputLines
    PI2.Logic NULL, InputLines
    PI2.Latch InputLines
    PI2.Debounce 10
End Sub

```

The script below uses the System II PI2 digital I/O device to turn on an LED interval marker on a button box at the beginning of a presentation/observation interval. The script uses the XBus command `PI2.SetBit` as well as the parameter `Interval`, returned by `SykofizX`, and `LIGHT_OFFSET`, a constant declared in the Library Script. Like System 3 subject interface scripts, System II hardware and scripting may be used to control a wide variety of subject interface systems.



The screenshot shows the Script Editor window with the following content:

```

Script Editor
File Edit
Language VBScript Time Out 10000

Scripts
Library Script
Abort
Run Begin
Pre-trial
Trial Begin
Presentation Begin
Presentation End
Trial End
Run End
Begin Response
Check for Response
End Response
Hit
Miss
False Alarm
Correct Rejection

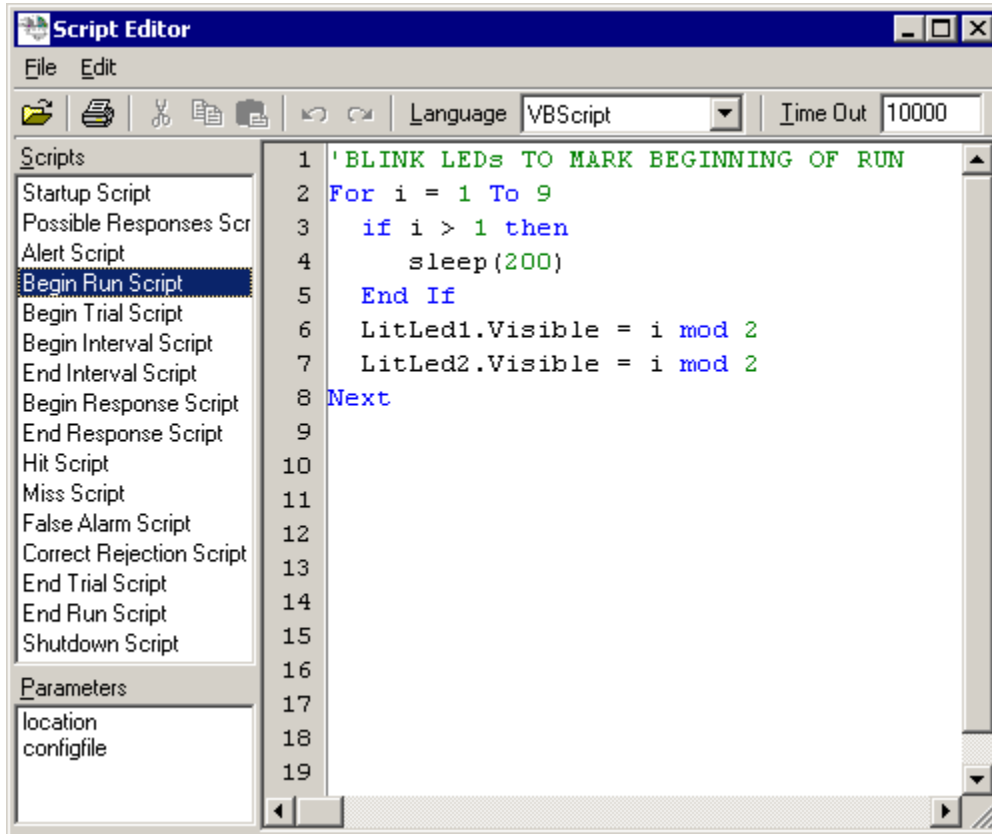
' "Presentation Begin" script for a button box
'
' This script will turn on a button box light corresponding to the
' presentation number.
'
' NOTE: This script uses code defined in the Library script
'
' Initial script coding by Nicholas Powers
' npowers@belugasoftware.com
'
' turn the light corresponding to the interval "on"
PI2.SetBit Interval + LIGHT_OFFSET

```

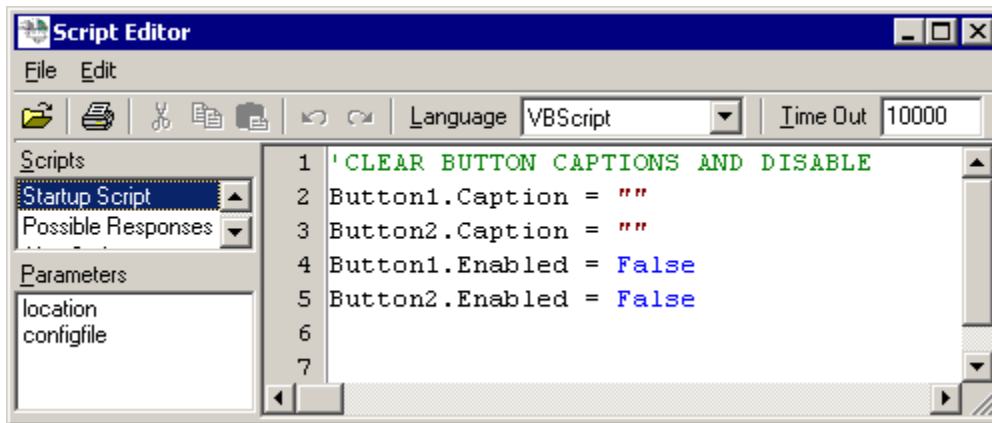
## KVM Subject Interface Scripts

**Plugin:** KVM.KeyboardVideoMouse

The KeyboardVideoMouse plugin uses a graphical design saved in a configuration file. Control over elements of that design is achieved through scripting. The script below illustrates flashing images designed to look like LEDs on a button box. The LED images in the configuration file are named LitLED1 and LitLed2. The script changes the visible property from true to false, effectively flashing the LEDs on and off. Similarly, scripts are used to enable and disable buttons or text boxes, providing a means for user input.



A KVM configuration might also include buttons with labels. The script below illustrates the default state of such buttons at the beginning of data collection. In this case, labels are not displayed and buttons are inactive.



Note that there is a single script marked by every possible event in the experimental sequence. Each script is executed at the onset of those events. Some scripts may be empty, in which case nothing will happen at those corresponding times. Also note that any parameters associated with the particular plugin are also exposed to the script editor for use in the scripts.

In addition to the commands available to all subject interface plugins, the following are used exclusively with the KVM plugin:

**objectname.visible**

Boolean. Toggles visibility property of referenced object.

**objectname.enabled**

Boolean. Toggles enable property of referenced object.

**objectname.caption**

String. Sets caption of referenced object.

**objectname.ResponseValue**

Integer. Codes response value of referenced object.

## Standard Plugins

A *plugin* is simply a collection of instructions within a single (DLL) file that dictates the behavior of the application when that plugin is referenced. All plugins have standard I/O parameters that allow them to communicate with the main application. Individual experimental designs may share a great deal of common functionality but differ substantially in one or more aspects of the design. Exchanging plugins relevant to specific parts of the design allows the user to reconfigure the experimental design as needed without having to re-create an entirely new design from scratch. In the experimental Design mode, plugins are selected from a list of available plugins to create an experimental or calibration design. The application then executes the design, including the commands specified in a particular plugin.

Critical in the experimental design and calibration processes is the selection of appropriate plugins used for:

- Stimulus Generation
- Stimulus Presentation
- Subject Interface
- Independent Variable Configuration

Together the presentation paradigm and independent variable configuration plugins specify the nature of the psychophysical procedure to be used. These include a variety of standard procedures such as same-different, yes-no, forced-choice, rating, identification, categorization, magnitude estimation, and magnitude production. These procedures may be used in combination with fixed and adaptive independent variable control. The application supports simultaneous collection of both closed and open-set subject responses as well.

The stimulus generation and subject interface plugins are compatible with TDT System 3 as well as System II hardware. Using System 3 hardware, a combination of RCO files created in the TDT RPvdsEx application and custom scripts created within the SykofizX application allow the user to generate and control stimuli and the digital I/O necessary for specific subject interface requirements. Using System II hardware, extensive customization can be achieved using standard AP2 APOS commands written in a simple scripting environment.

For added convenience, the same custom scripts used with System II hardware may be used with System 3 hardware in association with an AP2 emulator (APE) and the APE.rco RCO file. In this case, the commands normally executed on the AP2 are actually executed on the CPU of the PC and a final stimulus buffer is passed to the RP2 device. In each case, the script routines can be mixed, matched, and modified by the user to create novel designs.

In addition to digital I/O via the TDT System 3 and System II platforms, subject interface routines support a variety of KVM (keyboard, video, and mouse) configurations. The KVM subject interface plugin also includes a configuration utility that allows the user to design a variety of KVM I/O configurations using combinations of static images, animations, movies, and audio clips. Subject interface script routines are easily customized to work with other hardware I/O systems such as button boxes, switches, feeders, and lights.

## Parameter Tools

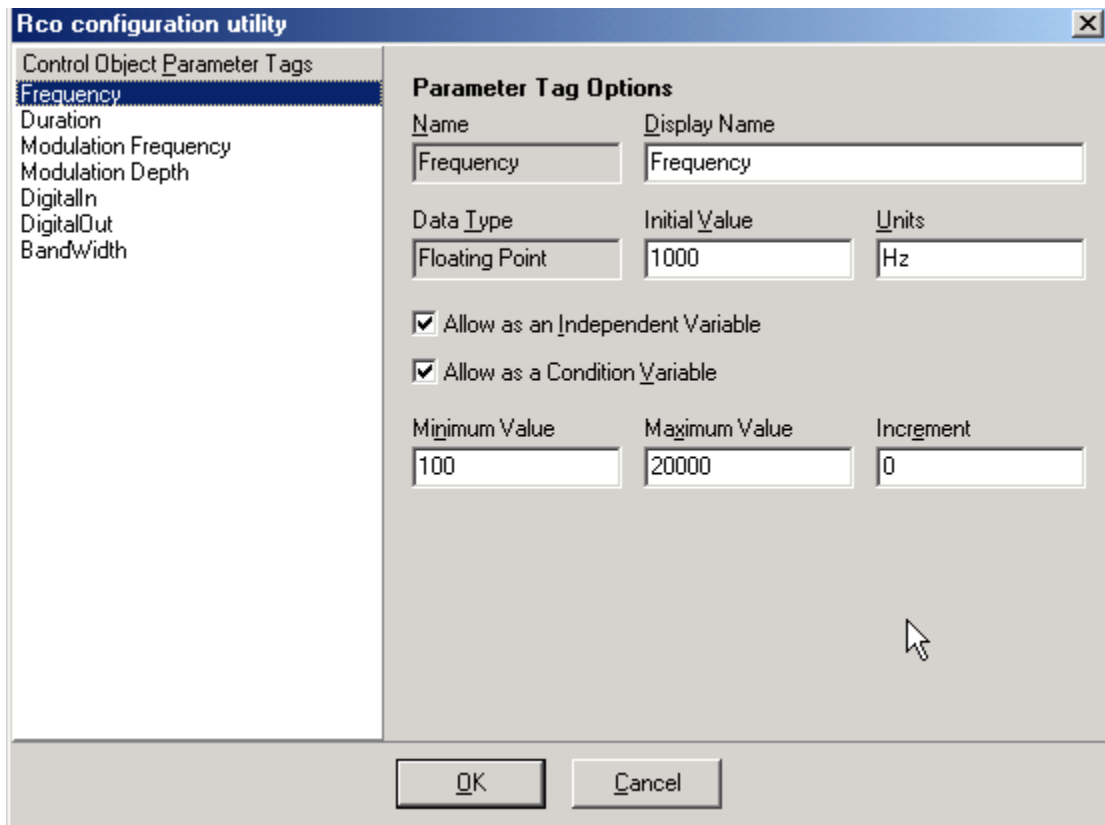
Many plugins permit the user to add new parameters to the plugin from the plugin parameter inspector. If the plugin permits user-defined parameters, the Add button will be enabled. These additional parameters are defined and modified using the following parameter tools: Configure, Info, Add, Edit, and Delete. Examples of such parameters might include *Stimulus Level* controlled by programmable attenuators, vowel categories used as conditions (e.g., /eh/, /oh/, /ih/, /ii/, /ae/, /ah/), etc.

**Note:** It is often the case that during the experimental design process, one may realize the need for user defined stimulus parameters later in the design process (e.g., when selecting condition parameters). When this occurs, simply go back to the Stimulus Generation wizard and add the necessary parameters. The examples below illustrate the use of custom parameters.



### Configure

Some plugins provide configuration utilities that are launched using the Configure button on the wizard. For example, the System3.RcoFile stimulus generation plugin allows the user to configure names, ranges, and default values for the parameter tags in a specified RCO file. In this case, clicking the Configure button launches the RCO configuration utility shown below.



The parameter tags made available by the System3.RcoFile plugin that were listed in the parameter inspector of the Stimulus Generation Configuration wizard, are listed again in the upper left of the configuration utility. By selecting any one of the parameter tags listed, the default behavior can be specified and/or modified. Furthermore, the possible functionality of these parameters can be specified, allowing the parameter to serve either as an Independent Variable or a Condition Variable or both. Later in the design process, parameters allowed to serve as independent or condition variables will be listed among the other parameters that could be chosen as independent or condition variables.

### Info

Once a parameter in the parameter inspector is highlighted, the Info button will be activated and a click of that button will display the details of that parameter.

### Add

Parameters not specified by the plugin may be added at design time. These parameters may not be associated with the plugin itself, but may be important in stimulus generation or may serve as important parameters in choosing the Independent Variable or Condition Variable values. Note that once the parameter is defined and added, it appears in the main parameter inspector as a User Defined parameter. The Info button provides a display of the parameter specifications. In this example, the parameter Atten is added and represents attenuation values that may be fed to a programmable attenuator such as the PA5 to control stimulus level.

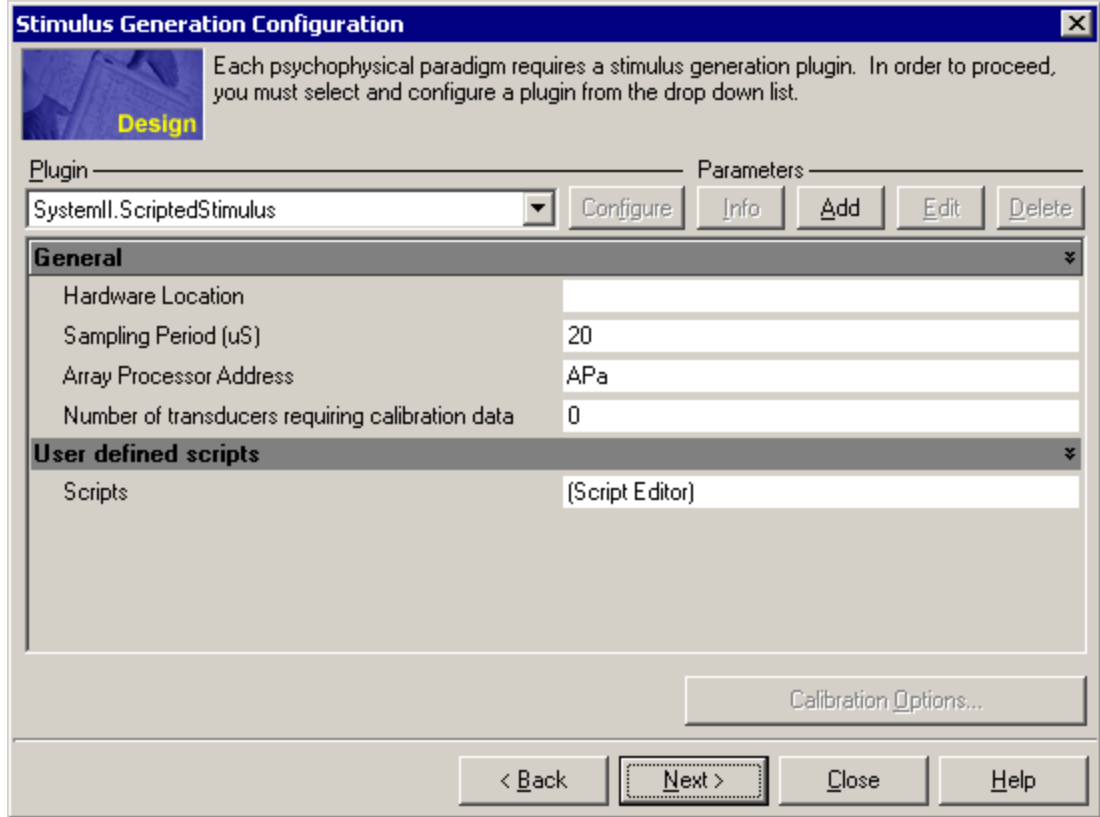
### Delete

The Delete button allows a user defined parameter to be deleted.

## Stimulus Generation

### SystemII.ScriptedStimulus

This plugin allows users to control TDT System II programmable hardware devices such as the AP2 and other XBus devices via the available parameters in the parameter inspector and via custom scripts. Following the traditional TDT System II approach, the AP2 Array Processor and various programmable XBus devices are used to generate and control stimuli. In addition to supporting nearly all of the APOS commands, the SykofizX application provides enhanced support for several APOS commands and provides optional automation for much of the APOS buffer management and System II D/A conversion.



Parameter	Description/Usage
Hardware Location	Default: None - Specify location of remote PC and hardware in a Local Area Network (LAN)
Sampling Period ( $\mu$ s)	Default: 20 - Sampling period in microseconds.
Array Processor Address	Default: APa - Choices are APa and APb. See System II support documentation for further information about AP addresses.
Number of transducers requiring calibration data	Default: 0 - When number is greater than 0, an additional parameter is added to the inspector requiring the specification of a transducer calibration file (.bcf).
User Defined Parameters	Default: None - The user can add parameters that may serve as condition variables or the independent variable.
User Defined Scripts	Default: None - Load and access user defined scripts.

### System II AP2 APOS Interface

See AP2 APOS Function Calls Support for a list of all APOS commands supported by the SykofizX SystemII.ScriptedStimulus plugin. Each of these commands is documented in the APOSHelp.pdf file found online at: <http://www.tdt.com/T2Download/manuals/aposref.pdf>

Six of the APOS commands have been modified slightly to simplify their usage and/or render them compatible with VBScript and JScript. For the tone and qwind commands, the original APOS usage required that two parameters be specified, the second of which was the sampling



period in microseconds. In the SykofizX application, the sampling period is always known and therefore need not be specified for these commands. For the remaining four commands (`_allotf`, `_allot16`, `_fir`, and `_iir`), the initial underscore in the command names are not supported by the scripting languages. Therefore, these commands have been renamed.

Below both the old and new usage is given:

Old Command	New Command/Usage
<code>tone freq, sper</code>	<code>tone freq</code> where <i>freq</i> is the tone frequency in Hertz
<code>qwind dur, sper</code>	<code>qwind dur</code> where <i>dur</i> is the window duration in ms
<code>_allotf npts</code>	<code>Uallotf npts</code> where <i>npts</i> is the length of the memory buffer to be allotted
<code>_allot16 npts</code>	<code>Uallot16 npts</code> where <i>npts</i> is the length of the memory buffer to be allotted
<code>_fir</code>	<code>Ufir</code>
<code>_iir</code>	<code>Uiir</code>

Additional SystemII.ScriptedStimulus plugin commands are listed and described below.

Command/Usage	Description
<code>dpushms duration</code>	Allocates stack space corresponding to the stimulus duration "duration". Analogous to the <code>dpush</code> command, <code>dpushms</code> uses knowledge of the sampling period to push the appropriate number of points onto the AP2 stack, rounded to the nearest integer number of points.
<code>dama</code> "damabuffername"	Returns the integer dama buffer number corresponding to the specified dama buffer name.
<code>qpopbufferf</code> "damabuffername"	Allocates floating point dama buffer and moves data from the stack top to that buffer. Allocates a dama buffer named "damabuffername" using the APOS <code>_allot</code> command, moves floating point data from the AP2 memory stack to a dama buffer named "damabuffername", and determines the time at which the application will automatically de-allocate the dama buffer using the APOS command <code>deallot</code> . De-allocation occurs at a time in the condition structure that mirrors the time of allotment. For example, if allotment of the dama buffer memory corresponds to "begin trial", then <code>deallot</code> is called when "end trial" occurs.  Alternatively, users may use the <code>allot</code> , <code>qpopf</code> , and <code>deallot</code> commands directly.
<code>qpopbuffer16</code> "damabuffername", channel#	Allocates floating point dama buffer, moves data from the stack top to that buffer, sets up sequence play list for D/A conversion. Allocates a dama buffer named "damabuffername" using the APOS <code>_allot</code> command, moves floating point data from the AP2 memory stack to a dama buffer named "damabuffername", and determines the time at which the application will automatically de-allocate the dama buffer using the APOS command <code>deallot</code> . De-allocation occurs at a time in the condition structure that

	<p>mirrors the time of allotment. For example, if allotment of the dama buffer memory corresponds to "begin trial", then deallot is called when "end trial" occurs. The channel# parameter determines the D/A channel to which the dama buffer will be sent. SykofizX automatically sets up a playlist and sequence list for use by the APOS command seqplay, assuming only one repetition per buffer.</p> <p>Alternatively, users may use the Uallot, qpopf, and deallot commands directly and may set up a playlist and sequence list manually as described in the APOS User's Guide.</p>
StimulusClass	Returns the current stimulus class under focus of the SykofizX application.
SampleFreq	Returns the current sampling frequency in Hertz.
SamplePeriod	Returns the current sampling period in microseconds.
ApplicationPath	Returns the current application path.
System II	Returns true if the current stimulus generation plugin is specific to System II hardware, otherwise returns false.
System 3	Returns true if the current stimulus generation plugin is specific to System 3 hardware, otherwise returns false.
Sleep ms	Pauses the current timing thread for the specified duration in ms. Uses the Windows precision timer for sub-ms accuracy.
SeqList	Returns the dama buffer number associated with the current sequence list created by the SykofizX application.
PlayList	Returns the dama buffer number associated with the current play list created by the SykofizX application.

### System II XBus Hardware Interface

The following XBus Hardware devices are supported by the SykofizX SystemII.ScriptedStimulus plugin. Support for each of these devices is identical to that described in the TDT XBDrv documentation.

- DA1
- DA3-1, DA3-2, DA3-4, DA3-8
- DD1
- AD1
- AD2
- PD1 (A/D and D/A commands only)
- PA4
- PI2
- SW2

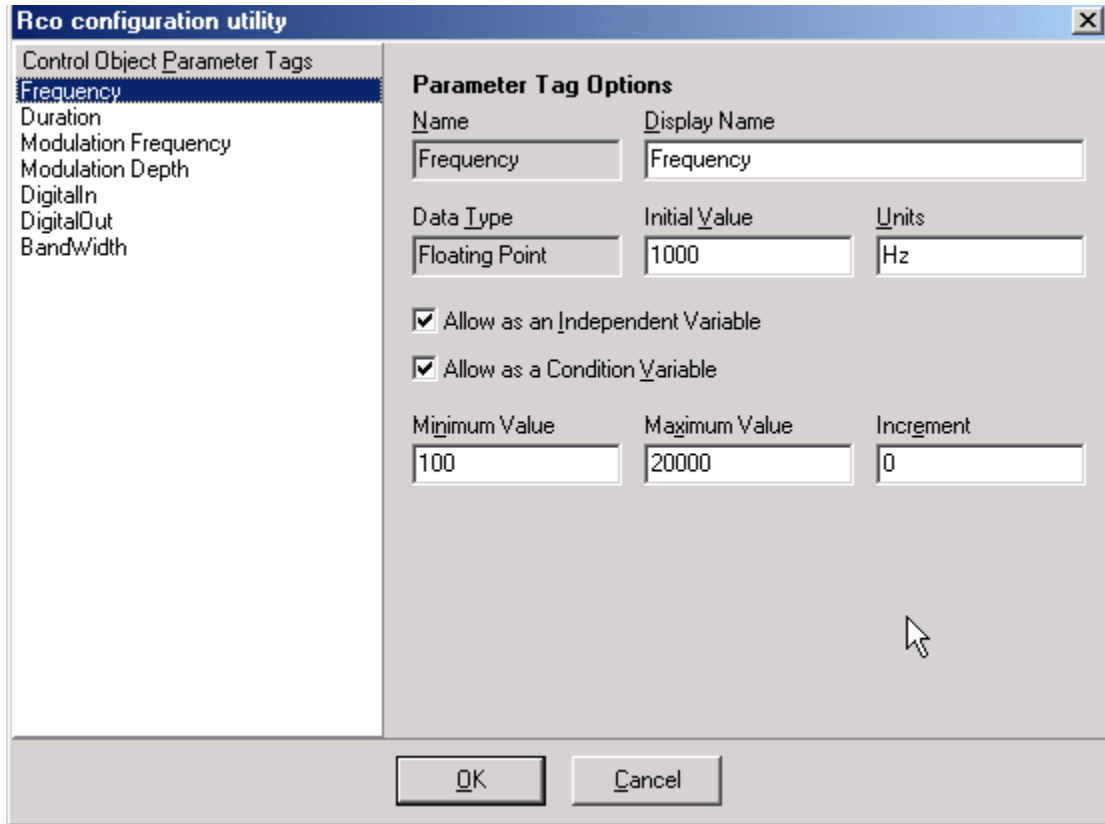
## System3.RcoFile

This plugin allows users to control TDT System 3 programmable hardware devices using RCO files created in RPvdsEx and custom scripts. The RCO file designed in the RPvdsEx serves as an interface between the SykofizX application and the System 3 hardware devices. This is similar to other TDT System 3-compatible software applications. Several Sample RCO files are provided with the SykofizX application. These RCO files are not fundamentally different from any other RCO files. They each use parameter tags to allow the parent application to send information to the hardware about the required stimulus parameter values (such as tone frequency, noise bandwidth, etc.). The System3.RcoFile plugin has the added requirement that each RCO file have output tags that allow the SykofizX application to determine what stimulus classes are supported by the RCO file, which software triggers are used to control stimulus I/O, and to monitor the activity of a certain stimulus class. The sample RCO files installed with SykofizX include files designed to load waveform data files from disk and files that interface with the APE (Array Processor Emulator).

Parameter	Description/Usage
ZBus interface	Default: USB - USB and Gigabit interfaces are supported. The default interface may be set at design time, however, that interface can be changed at run time as needed.
Device type	Default: RP2 - The plugin supports the Rx processing family including the RP2, RM1, RM2, RL2, RV8, RA16, and RXn Realtime Processor devices.
Device number	Default: 1 - Specifies the device number as registered by TDT System 3 drivers.
Sample rate	Default: 12 kHz. Optional rates of 6, 25, 50, 100, and 200 kHz are available - By convention, the actual sample rates differ from these nominal rates. See the RPvdsEx Manual for more information.
Rco File	Default: None - Choose the desired Rco file with to associate with the design.
Use Calibration File	Default: False - Specify whether a .bcf calibration file is to be used by the stimulus generation plugin.
System II Emulation	Default: False - Enable the Array Processor Emulator (APE) support.
Rco File Parameter Tags	Default: Tags in associated RCO file - Any single valued numeric tag (integer, float, boolean) in RCO file appears and can be set to a specific numeric value here.
User Defined Parameters	Default: None - The user can add parameters that may serve as condition variables or the independent variable.
User Defined Scripts	Default: None - Load and access user defined scripts.

### Configure

Some plugins provide configuration utilities that are launched using the Configure button on the wizard. For example, the System3.RcoFile stimulus generation plugin allows the user to configure names, ranges, and default values for the parameter tags in a specified RCO file. In this case, clicking the Configure button launches a RCO configuration utility, as shown below.



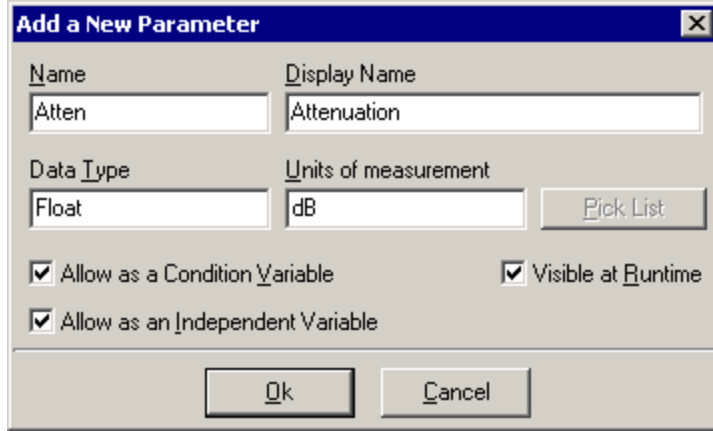
The parameter tags made available by the System3.RcoFile plugin that were listed in the parameter inspector of the Stimulus Generation Configuration wizard, are listed again in the upper left of the configuration utility. By selecting any one of the parameter tags listed, the default behavior can be specified and/or modified. Furthermore, the possible functionality of these parameters can be specified, allowing the parameter to serve as either an Independent Variable or a Condition Variable.

### Info

Once a parameter in the parameter inspector is highlighted, the Info button will be activated and clicking that button will display the details of that parameter.

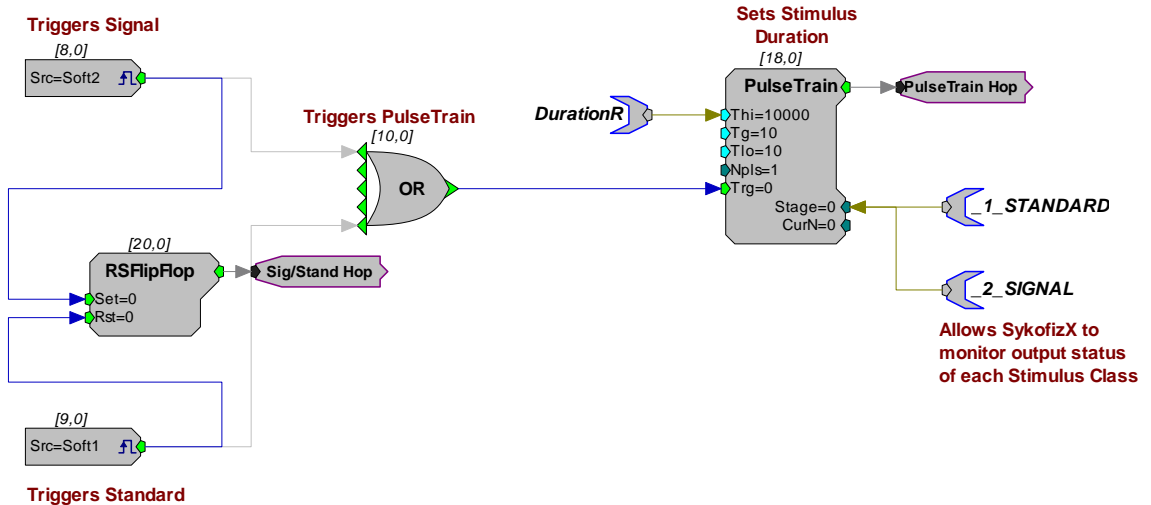
### Add

Parameters not specified by the plugin may be added at design time. These parameters may not be associated with the plugin itself, but may be important in stimulus generation or may serve as important parameters in choosing the Independent Variable or Condition Variable values. Note that once the parameter is defined and added, it appears in the main parameter inspector as a User Defined parameter. The Info button provides a display of the parameter specifications. In this example, the parameter Atten is added and represents attenuation values that may be fed to a programmable attenuator such as the PA5 to control stimulus.



### Using Parameter Tags in RpvdsEx

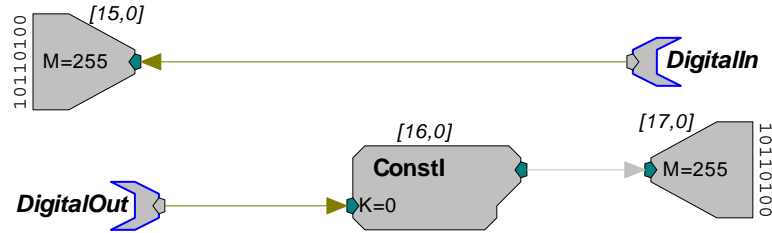
There are three basic categories of parameter tags used to link RpvdsEx circuits to the SykofizX application. Input tags allow the SykofizX application to send variable values and file names to the RpvdsEx circuit. The application scans the RCO file for any parameter tags, making those parameters available in the plugin parameter inspector and as potential condition or independent variables. Example input parameter tag DurationR is shown in the image below taken from the AMNoise\_Template.rpx file.



Output tags are a second category of parameter tags used by SykofizX. The SykofizX application requires a specific format for output tags. The tag name must begin with an underscore, followed by the D/A channel number, followed by an underscore, and finally the name of the stimulus class associated with the specified D/A channel. These output tags are considered stimulus classes by the SykofizX application. The example above specifies two stimulus classes (STANDARD and SIGNAL), however, a user may add additional stimulus classes by creating .rpx files with additional formatted output tags. The third category of parameter tags is reserved for digital I/O using System 3 devices.

**These 5 components control the digital I/O lines via RPx and RMx devices. All logic for the bit masks is set in stimulus generation scripts. Simply specify the logic for "DigitalIn" or "DigitalOut" lines in the scripts.**

**DO NOT MODIFY**



The application communicates with the RPx digital I/O lines via the DigitalIn and DigitalOut parameter tags. These tags called from custom scripts when using the System 3 Subject Interface plugin.

### Demo.StimulusSound

The Demo.StimulusSound plugin requires no external hardware and relies on the PC sound card to generate a selection of "Windows Sounds." This allows users to use the Design mode of the application independent of stimulus generation hardware or specific parameters associated with stimulus generation.

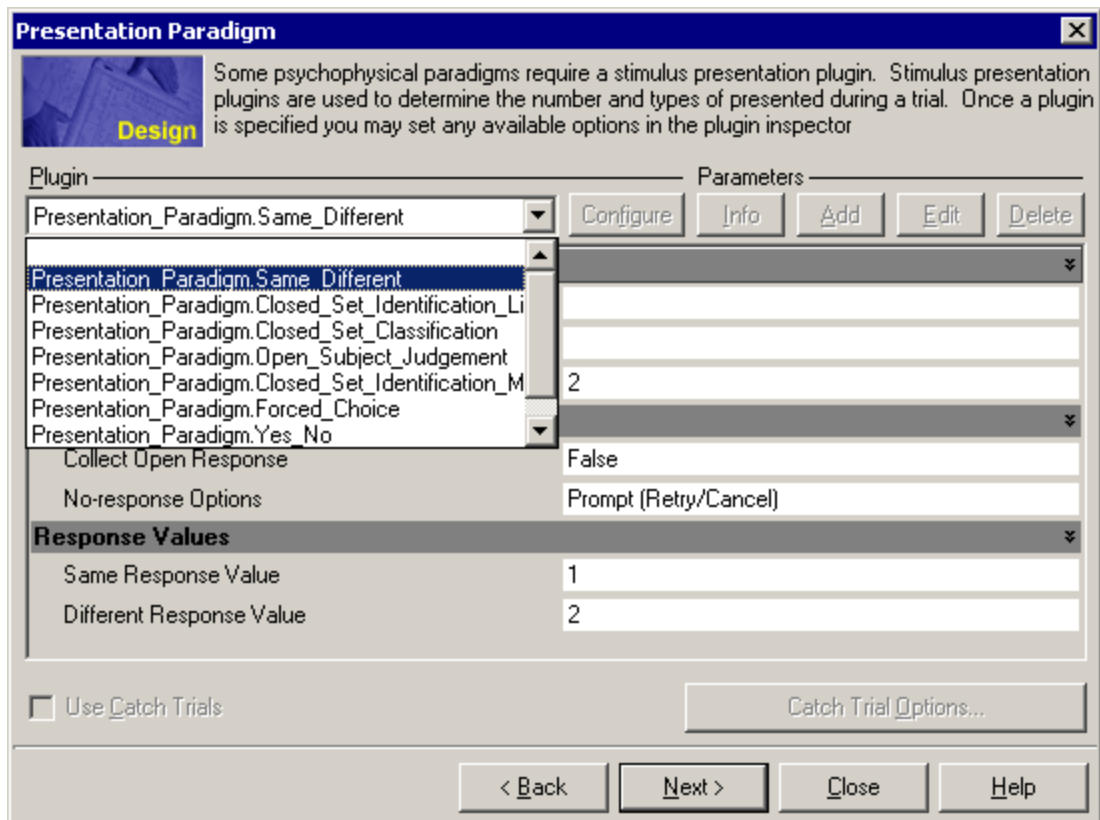
## Presentation Paradigm

Each experiment requires a presentation paradigm defined in a Presentation plugin. Stimulus presentation plugins are used to determine the number and types of stimuli to be presented during an experimental trial. Once a plugin is chosen, the user may set any available options in the plugin parameter inspector. The available parameter options depend on the chosen plugin. All standard plugins have Stimulus Classes parameter options. When supported, individual plugins may also have parameter options related to Intervals, Open Response, and Response Values. Those presentation plugins with a variable number of observation intervals provide an option to set the number of observation intervals. In addition, some plugins support the concepts of pre-trial or post-trial reminder intervals. Reminder intervals may be associated with either the signal or the standard stimulus class, and stimuli may be customized for reminder intervals via scripting.

Furthermore, the subject interface may have different behavior for reminder intervals than for regular observation intervals. A pre-interval reminder is assigned the interval number zero (0) by default whereas a post-interval reminder is assigned the interval number  $n+1$  where  $n$  is the number of regular observation intervals. These interval numbers are helpful when writing scripts associated with reminder intervals. The Open Response parameter permits the collection of open set responses from the subject, in the form of text or numeric entries, slider position, etc. If the open response parameter is set to true, then the subject interface plugin will be required to handle an open response.

### Plugins

The desired presentation plugin must be selected from the list of plugins available in the drop-down menu below the Plugin label.



Presentation plugins are often thought of as the psychophysical method used in an experiment. Below is a list of the available presentation plugins along with a description of each.

### Yes No

The typical yes/no paradigm may include a single or multiple observation intervals and requires a binary response (e.g., Yes/No). This plugin supports optional reminder and catch trial intervals. Reminder intervals may be used to create a type presentation paradigm. Note that the response categories need not be restricted to the responses Yes and No. Any binary response may fit into this presentation paradigm and responses may be coded as desired.

Parameter	Description/Usage
Target Stimulus Class	Corresponds to the "Yes" response. Choose the Signal or Standard Stimulus Class.
Reminder Stimulus Class	Choose either the Signal or Standard Stimulus Class.
Catch Trial Stimulus Class	Choose either the Signal or Standard Stimulus Class - See Catch Trials below.
Pre-Interval Reminder	Boolean: True invokes a reminder interval at the beginning of each trial.
Number of Trials	Number of trials (excluding reminder intervals). The possible number of trials is unlimited.

Post-Interval Reminder	Boolean: True invokes a reminder interval at the end of each trial.
Collect Open Response	Boolean: In addition to the Yes/No response, an optional open response may be recorded on an interval-by-interval basis.
Yes Response	Default = 1. Allows the application to link the presentation interval with the subject response to determine and record subject performance.
No Response	Default = 0. Allows the application to link the presentation interval with the subject response to determine and record subject performance.
Done Response	Default = -1. Some designs require the concept of done, allowing the subject to indicate that a trial or a run is done. By default, this parameter is not used and has a value of -1.

### Forced Choice

Subjects are presented with  $n$  observation intervals and are "forced" to choose the target interval from the  $n$  choices. The end of an experimental trial occurs when a listener responds or when the established time-out occurs. Reminder intervals, catch trials, and open responses are optional parameters.

Parameter	Description/Usage
Target Stimulus Class	Choose the Signal or Standard Stimulus Class as the Target Stimulus.
Standard Stimulus Class	Choose the Signal or Standard Stimulus Class as the Standard Stimulus.
Reminder Stimulus Class	Choose either the Signal or Standard Stimulus Class.
Catch Trial Stimulus Class	Choose either the Signal or Standard Stimulus Class - See Catch Trials below.
Pre-Interval Reminder	Boolean: True invokes a reminder interval at the beginning of each trial.
Number of Trials	Number of trials (excluding reminder intervals). The possible number of trials is unlimited.
Post-Interval Reminder	Boolean: True invokes a reminder interval at the end of each trial.
Collect Open Response	Boolean: In addition to the single closed response, an optional open response may be recorded on an interval-by-interval basis.
Interval Label	Default = Interval. Used in data storage as well as KVM displays to mark different observation intervals.



## Same Different

Subjects are presented with two observation intervals and must determine whether they were the "same" or "different". Reminder intervals are optional. A go/no-go type procedure can be created using a pre-interval reminder constructed of multiple standard stimuli.

Parameter	Description/Usage
Stimulus Class A	Choose the Signal or Standard Stimulus Class for element A in a Same/Different paradigm (e.g., AA, AB, BA, BB).
Stimulus Class B	Choose the Signal or Standard Stimulus Class for element B in a Same/Different paradigm.
Number of Intervals	Integer: Number of presentation intervals. Minimum is 2. If the number of intervals, n, is greater than 2, the stimulus selected for the first interval is repeated n-2 times. This allows one to set up a Go/No Go procedure.
Collect Open Response	Boolean: In addition to the single closed response, an optional open response may be recorded on an interval-by-interval basis.
Same Response Value	Default = 1. Allows the application to link the presentation interval with the subject response to determine and record subject performance.
Different Response Value	Default = 2. Allows the application to link the presentation interval with the subject response to determine and record subject performance.

## Subject Judgment

Subjects are required to judge some attribute of the stimulus or stimuli presented. Several different paradigms can be constructed using this plugin, simply by selecting appropriate options and by using specific subject instructions. Examples include stimulus rating, matching, etc.

For example, a matching task might include a reminder interval with some standard stimulus followed by a target interval. In the target interval, a signal is presented and its independent variable may be varied on the basis of subject responses. Instructing the subject to adjust the stimulus in the target interval to match that in the reminder interval yields a matching task.

Parameter	Description/Usage
Target Stimulus Class	Corresponds to the "Yes" response. Choose the Signal or Standard Stimulus Class.
Reminder Stimulus Class	Choose either the Signal or Standard Stimulus Class.
Pre-Interval Reminder	Boolean: True invokes a reminder interval at the beginning of each trial.
Number of Intervals	Number of Intervals (excluding reminder intervals). The possible number of intervals is unlimited.

Post-Interval Reminder	Boolean: True invokes a reminder interval at the end of each trial.
Collect Open Response	Boolean: In addition to the Yes/No response, an optional open response may be recorded on an interval-by-interval basis.
Correct Response Value	Default = 1. Allows the application to link the presentation interval with the subject response to determine and record subject performance.
Correct Response Label	Default = Yes.
Incorrect Response Value	Default = 2. Allows the application to link the presentation interval with the subject response to determine and record subject performance.
Incorrect Response Label	Default = No.
Done Response Value	Some designs require the concept of done; allowing the subject to indicate that a trial, a track, or a sub-track is done. By default, this parameter is not used and has a value of -1.
Done Response Label	Default = Done.

## Open Subject Judgment

Subjects are required to judge some attribute of the stimulus or stimuli presented. Several different paradigms can be constructed using this plugin, simply by selecting appropriate options and by using specific subject instructions. Examples include magnitude estimation, open set rating, description, etc.

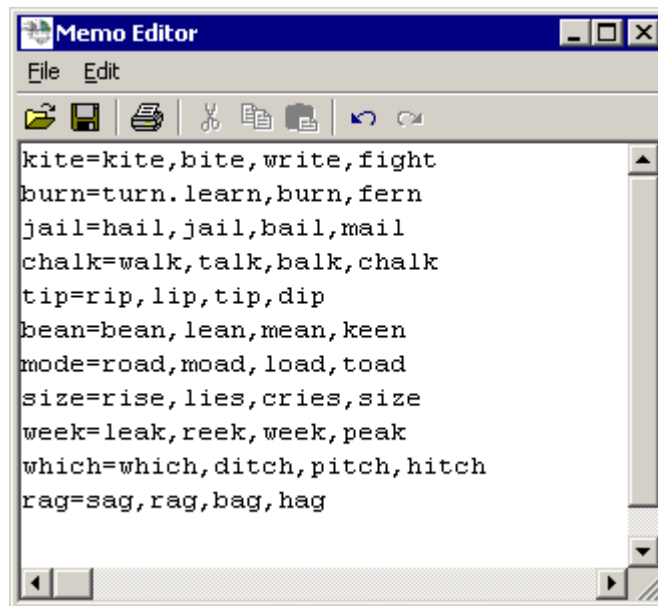
Parameter	Description/Usage
Target Stimulus Class	Select the Target Stimulus Class.
Reminder Stimulus Class	Select the Reminder Stimulus Class.
Pre-Interval Reminder	Boolean: True invokes a reminder interval at the beginning of each trial.
Number of Intervals	Integer: Number of target stimulus intervals.
Post-Interval Reminder	Boolean: True invokes a reminder interval at the end of each trial.
Open Response Type	String, Integer, Float. Choose the appropriate response type.
Open Response Minimum	Default = 0. Set the minimum open response value. This parameter does not apply to "String" response types.

Open Response Maximum	Default = 0. Set the maximum open response value. This parameter does not apply to "String" response types.
-----------------------	---

### Closed Set Identification - List

On each trial, the subject must identify the stimulus by choosing the appropriate response alternative from a fixed set of alternatives related to the stimulus token. Within a given condition, the number of stimulus possibilities does not need to correspond to the number of response alternatives. For example, a given condition might include 25 different stimuli and each stimulus might have four stimulus-specific response alternatives. Thus, the task is closed-set identification; however, the stimuli do not form a simple matrix and thus must be displayed as a list of stimulus values and responses. This is analogous to a multiple-choice experiment.

Parameter	Description/Usage
Stimulus Class 1	Select the Signal or Standard Stimulus Class for Stimulus Class 1.
Collect Open Response	Boolean: In addition to the single closed response, an optional open response may be recorded on an interval-by-interval basis.
Foils	Memo: Use the associated memo pad to specify link the possible response strings to the target item (see figure below).



Linked with a KVM subject interface, responses (foils) might be displayed as text on buttons or images on panels within the display, as in design Example 9 on page 114.

### Closed Set Identification - Matrix

On each trial, the subject must choose the response alternative from a fixed set of alternatives corresponding to the stimulus tokens. Within a given condition, the number of stimulus possibilities and response alternatives are identical, creating a square stimulus-response matrix. This type of design is sometimes called a confusion matrix.

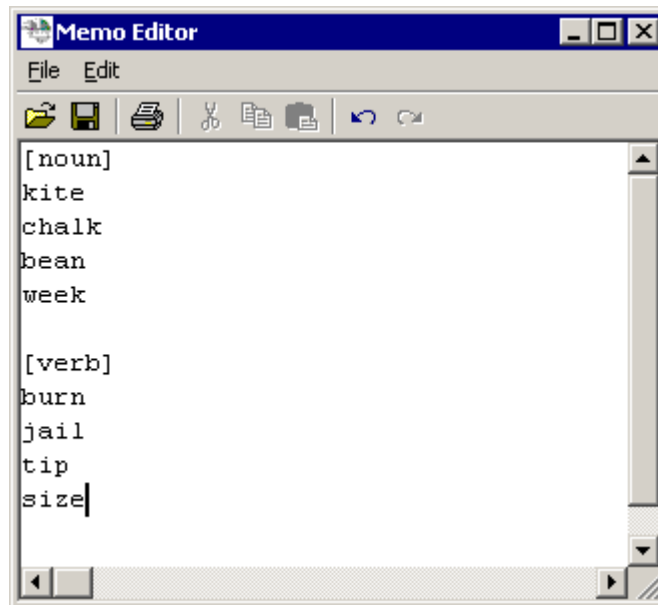
Parameter	Description/Usage
Stimulus Class	Select the Signal or Standard Stimulus Class for Stimulus Class.
Collect Open Response	Boolean: In addition to the single closed response, an optional open response may be recorded on an interval-by-interval basis.

## Closed Set Classification

On each trial, the subject must assign the stimulus to one of a set of possible classes or categories. For example, an experiment might require classification of monosyllabic words as either nouns or verbs. In that case, the stimuli are specified in the stimulus generation plugin and the corresponding categories are specified in the Closed Set Classification plugin using a memo pad.

Parameter	Description/Usage
Stimulus Class	Select the Signal or Standard Stimulus Class for Stimulus Class.
Collect Open Response	Boolean: In addition to the single closed response, an optional open response may be recorded on an interval-by-interval basis.

Categories and items are specified in a memo list, as shown below.



## Subject Interface

Each experiment requires that a Subject Interface plugin be chosen and configured. The subject interface controls all information that the subject either gives or receives, with the exception of the actual experimental stimulus. In standard human psychophysical experiments, that might include warning markers, interval markers, feedback, and subject responses such as button presses or changes in slider position. SykofizX provides the software required to interact with external hardware.

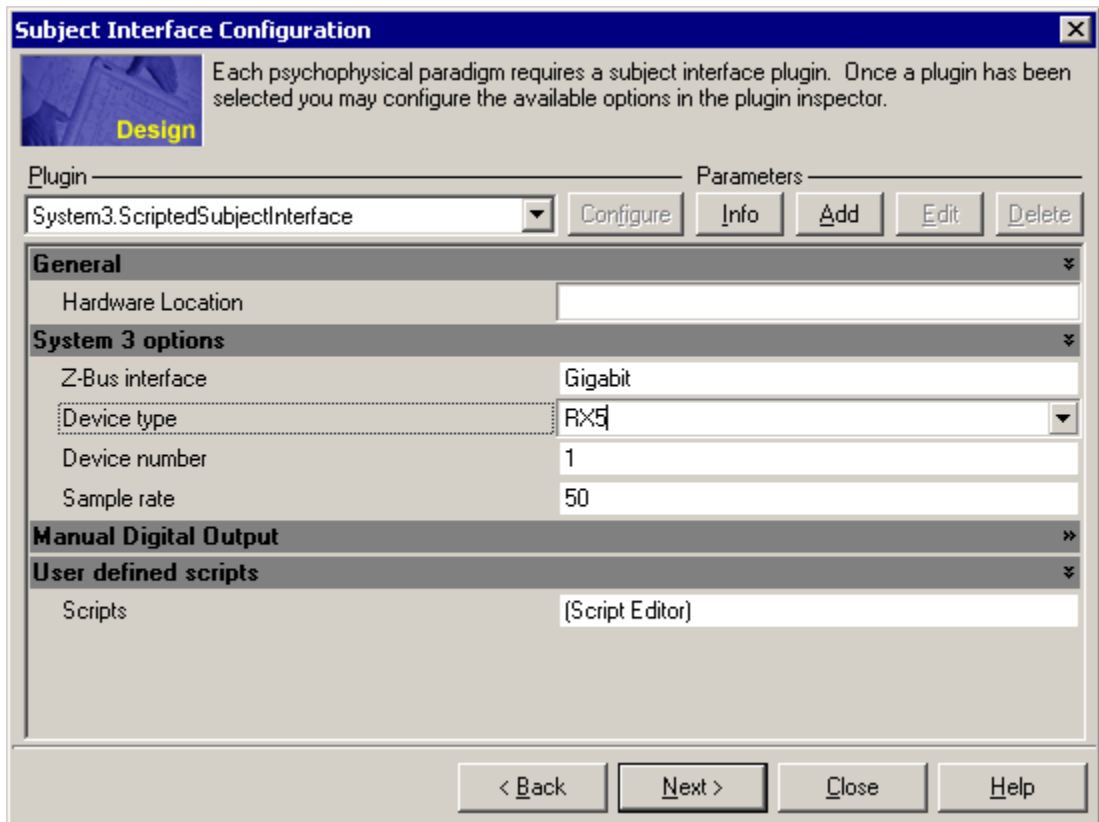
As with other plugins, once selected, a parameter inspector is loaded into the main window. Each option available from the plugin appears in the parameter inspector and can be customized by the user. The available parameters are grouped by common functionality. The parameters and parameter groups differ across plugins.

There are three Subject Interface plugins. One is associated with System 3, three with System II, and one is associated with the PC's KVM hardware rather than TDT hardware.

The System 3 and System II Scripted Subject Interfaces use custom scripts to interact with the System 3 digital I/O lines or the System II PI2 parallel interface digital I/O lines. The scripts may also be used to interact with other aspects of the System 3 ZBus and System II XBus hardware devices. An infinite number of external hardware devices can be controlled with these TDT devices, however, many psychophysicists will use a simple button box with an LED display. In any case, the I/O lines can be controlled via user-defined scripts that may be executed at any point in time during data collection. See Subject Interface Scripts, page 63 for more information.

### System 3

**Plugin:** System3.ScriptedSubjectInterface

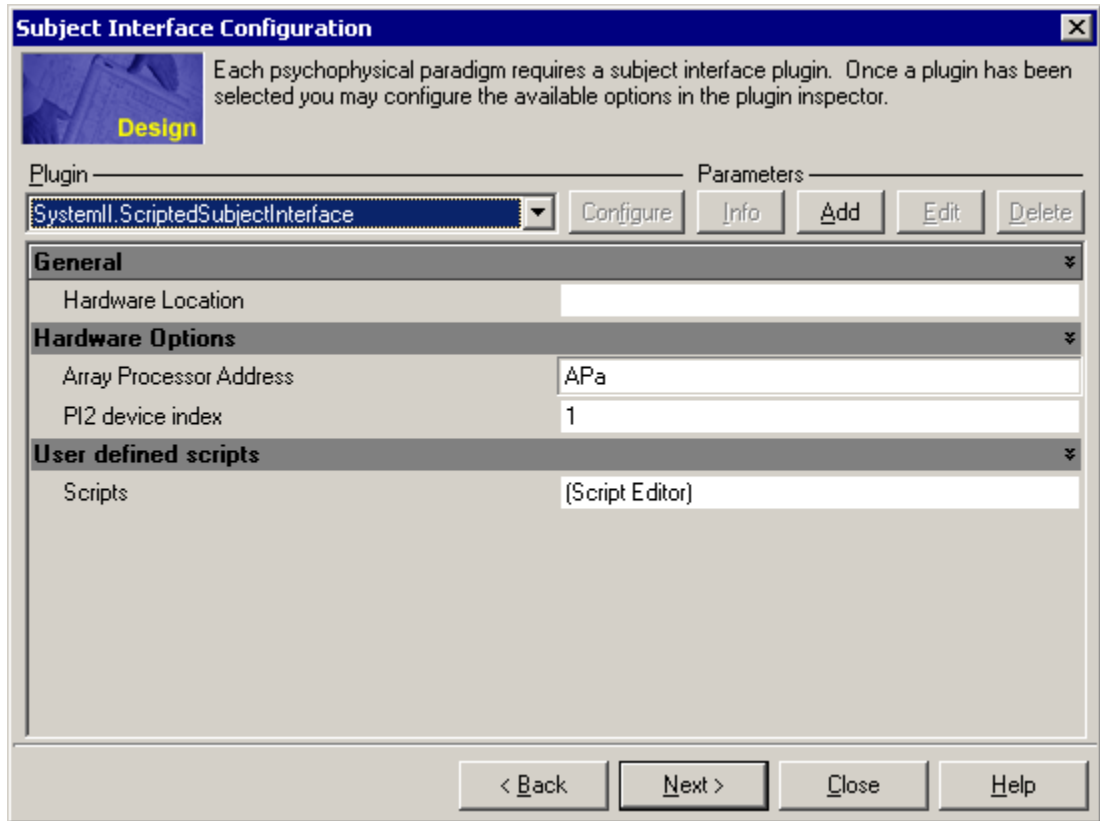


Parameter	Description/Usage
Hardware Location	Specify location of remote PC and hardware in a Local Area Network (LAN).
ZBus Interface	USB or GigaBit.
Device type	Select RPx, RMx, or RXn device type.

Device Number	Select device number. Dependent on number of like devices in hardware system.
Sample Rate	Sample rate in microseconds using RPx/RMx/RXn convention. Note that labeled sample rate is not actual sample rate: See the RPvdsEx Manual for more information.
Manual Digital Output	Not fully supported or documented. Enable and configure a manual control toolbar for overriding automated control of subject interface via digital I/O lines.
Scripts	Access current script library or load a script library that has been saved to an external file.

## System II

**Plugin:** SystemII.ScriptedSubjectInterface



The System II subject interface is similar to the System 3 subject interface; however, in System II the digital I/O lines of the PI2 are often used as the primary hardware interface with peripheral subject interface hardware.

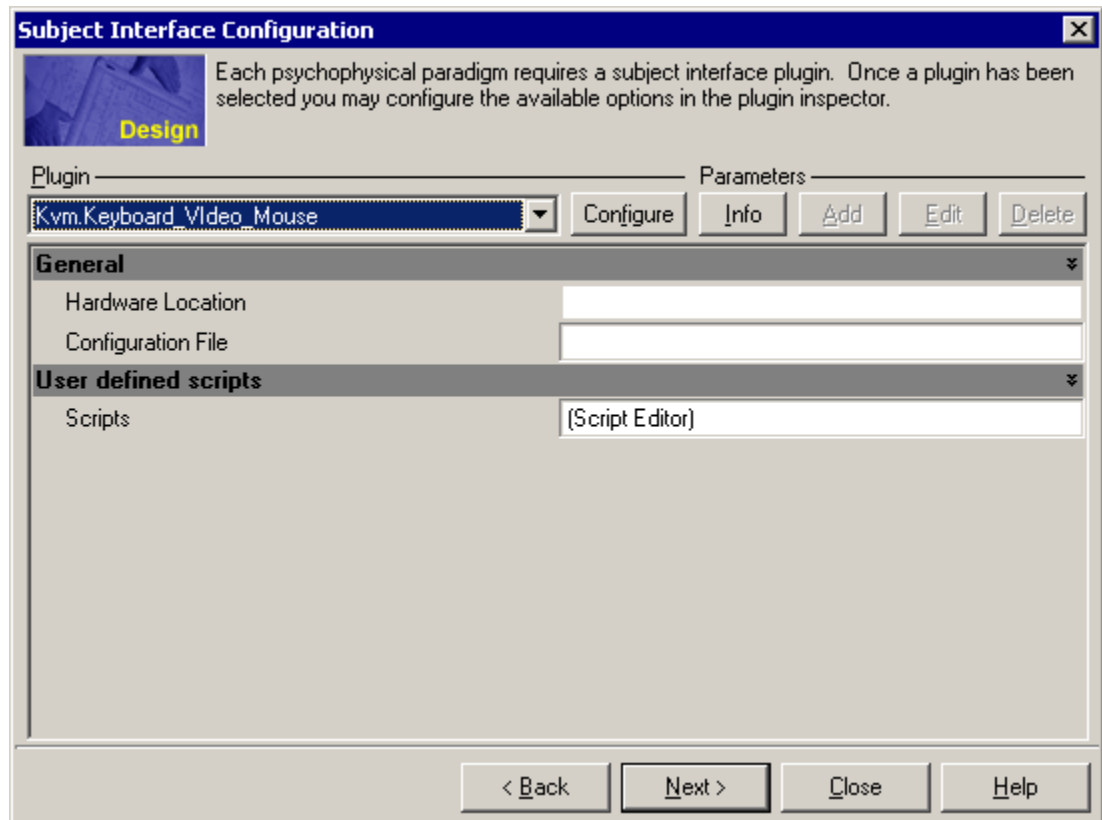
## KVM

**Plugin:** KVM.KeyboardVideoMouse

This subject interface relies upon the computer's keyboard, video, and mouse capabilities to implement a subject interface. Users may configure use either the mouse, a keyboard, or a touch screen configured with a video monitor as input devices to collect subject responses. The video monitor may be used as an output device to provide the subject information.

To add a KVM subject interface to a design:

1. Specify remote hardware location (if applicable)
2. Select **KVM.Keyboard\_Video\_Mouse** plugin
3. Select a **KVM configuration file** (.kvm)
  - a. To create a new configuration file select **Configure**
  - b. Use the KVM configuration utility to design the interface canvas then save the file
  - c. Select the new file as the KVM configuration file to be used
4. Set up KVM scripts. Create new script library or import existing script library



<b>Parameter</b>	<b>Description/Usage</b>
Hardware Location	Specify location of remote PC and hardware in a Local Area Network (LAN).
Configuration File	Specify a KVM configuration file created with the Configuration Utility. The Configuration Utility is accessed via the Configure button at the top of the window.
Scripts	Access current script library or load a script library that has been saved to an external file.

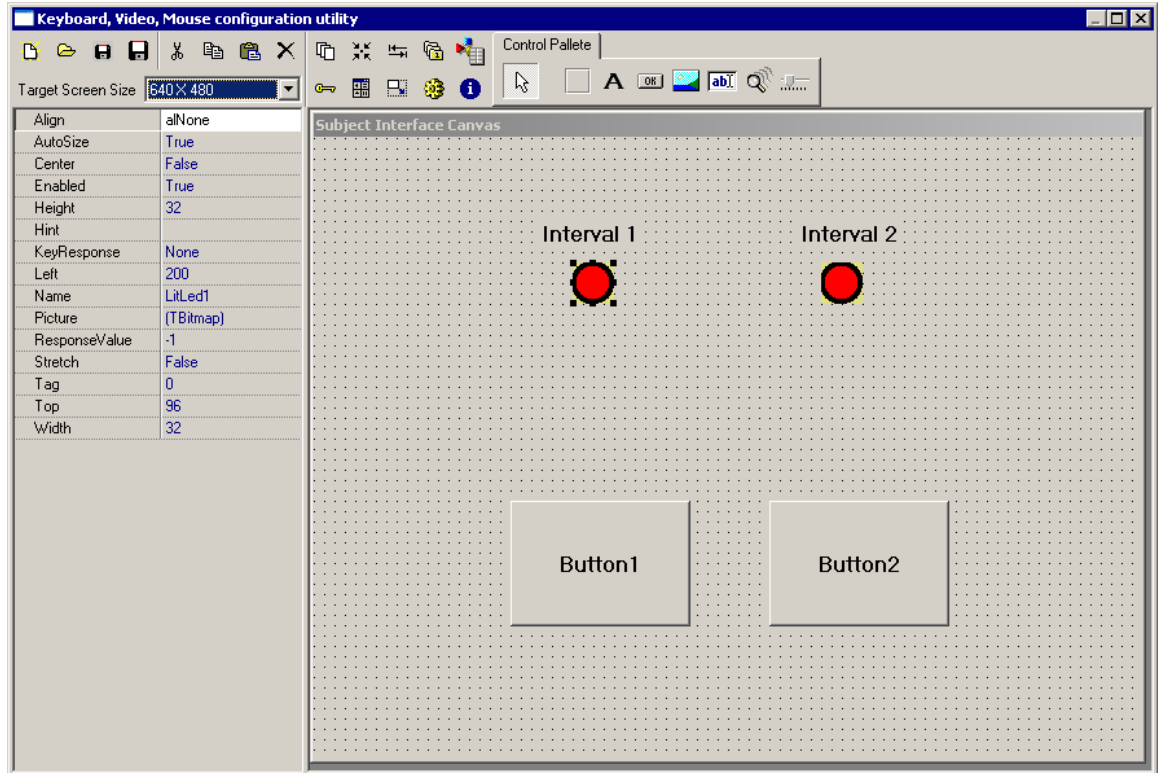
To simplify the design of KVM subject interfaces, a KVM configuration utility is built into the SykofizX application. The KVM configuration utility (shown below) can be launched by selecting the Configuration parameter button from the Subject Interface Configuration wizard. Begin by selecting items from the Control Palette and dropping them on the Subject Interface Canvas.

Clicking an object on the canvas selects that object and opens the parameter inspector for that object to the left of the Subject Interface Canvas. One can modify a variety of parameters associated with individual canvas objects. When editing the parameter inspector, note that most options require the Enter key to be pressed before the change takes effect. Most of the parameter options are intuitive and are not described here.

Exceptions include the following:

<b>Parameter</b>	<b>Description/Usage</b>
KeyResponse	Select a keyboard key to link to a response object. This allows the user to respond via key press as well as mouse click.
ResponseValue	Associates a response value with a given response. Allows the application to link the presentation interval with the subject response to determine and record subject performance.
OpenResponse	Not used.





The following list describes the toolbar icons.

**New:** Create a New Canvas/Configuration File.

**Open:** Open an existing Canvas/Configuration File.

**SaveAs:** Save current Canvas/Configuration File with a new file name.

**Save:** Save current Canvas/Configuration File.

**Cut:** Cut selected item.

**Copy:** Copy selected item.

**Paste:** Paste item on clipboard.

**Delete:** Delete selected item.

**Target Screen Size:** Set target screen size (video display size).

**Note:** Selection of a screen size smaller than the video resolution will partially fill the available screen.

**Select All:** Select all objects on the canvas.

**Align to Grid:** Align items on the canvas to a grid.

**Align:** Align selected items. Launches alignment window with options.

**Tab Order:** Set the tab order. Launches tab order window with options.

**Property Inspector:** Toggles Property Inspector to the left of the canvas on/off.

**Lock/Unlock:** Lock/unlock toggle to protect objects on canvas.

**Show Alignment Palate:** Launches alignment palate tool for convenient access to alignment options.

**Size:** Launches window for customizing the size of objects.

**Designer Properties:** Allows user to customize the Configuration Utility Properties.

**About:** Displays information about SykofizX KVM Configuration Utility.

**Pointer:** Select and activate pointer.

**Panel:** Select Panel object type. Drop on canvas by moving cursor to desired location and pressing left mouse button.

**Label:** Select Label object type. Drop on canvas by moving cursor to desired location and pressing left mouse button.

**Button:** Select Button object type. Drop on canvas by moving cursor to desired location and pressing left mouse button.

**Image:** Select Image object type. Drop on canvas by moving cursor to desired location and pressing left mouse button.

**Note:** To load a new image, select **Picture** from the parameter inspector, click in the box to the right, click **...**, select **Load** from the new window, navigate to and select desired file, choose **Open**, press **OK**.

**Edit Box:** Select Edit Box object type. Drop on canvas by moving cursor to desired location and pressing left mouse button.

**Animation:** Select Animation object type. Drop on canvas by moving cursor to desired location and pressing left mouse button.

**Slider:** Select Slider object type. Drop on canvas by moving cursor to desired location and pressing left mouse button.

## Implementation

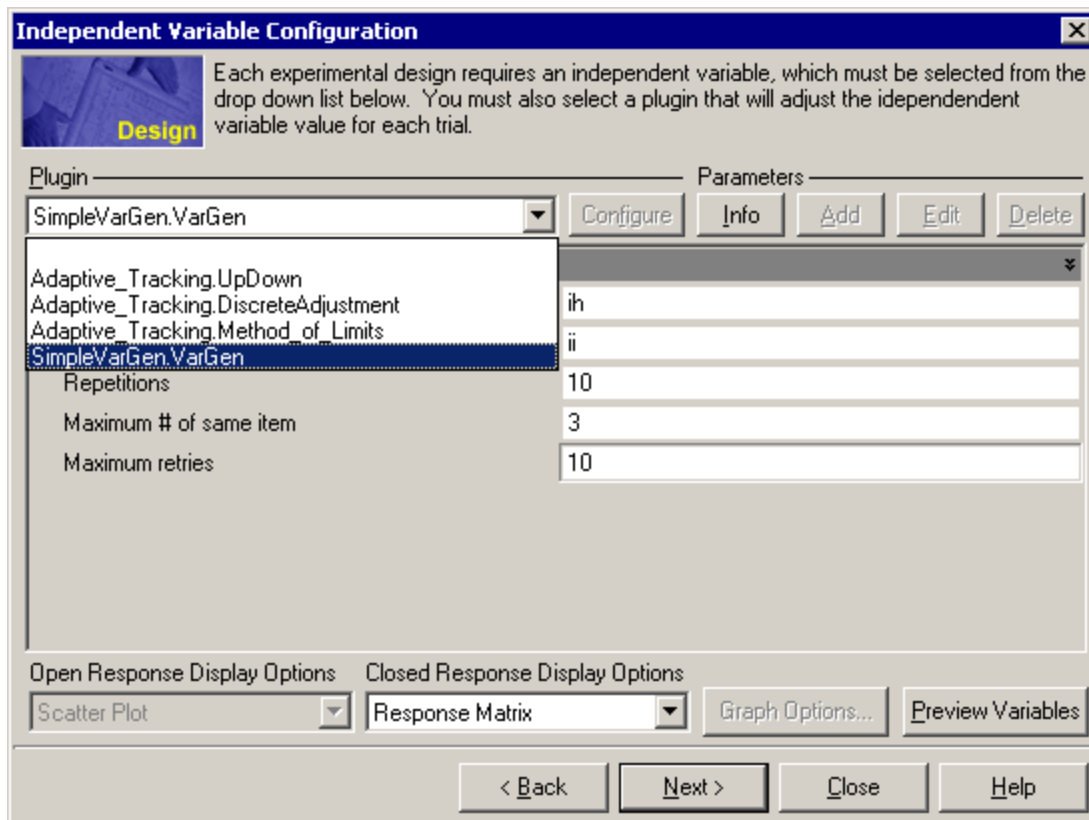
There are many ways to implement a KVM subject interface. For many experimenters, a single PC, keyboard, video monitor, and mouse may be used by both the experimenter and subject. In this case, they take turns using the PC. For others, it may be useful to have one PC and multiple KVM systems. This may be accomplished crudely by simply plugging in multiple devices to a PC (e.g., dual monitors, dual keyboards via PS2/USB combinations, dual mice via PS2/USB combinations) or elegantly via KVM switching devices. Using the built in SykofizX KVM server, one may also connect to a remote PC, using a host PC to run the experiment and the remote PC to run the KVM server and provide the KVM interface.

Standard monitors may easily be converted to touch-screen monitors with third-party add-on devices. These devices typically interpret touches as mouse clicks and allow the user to interact with the video display simply by touching a location on the display. Dedicated touch-screen monitors are also available and work in a similar fashion.

## Independent Variable Configuration

SykofizX supports five modes of independent variable manipulation that can be separated into adaptive tracking and fixed independent variable manipulation. Adaptive tracking options allow the independent variable to vary during an experimental run according to a specified algorithm based on previous subject responses or under control of the subject (i.e., matching or adjustment experiments).

Available adaptive tracking plugins are shown in the image and described in the topics that follow.



## Adaptive\_Tracking.UpDown

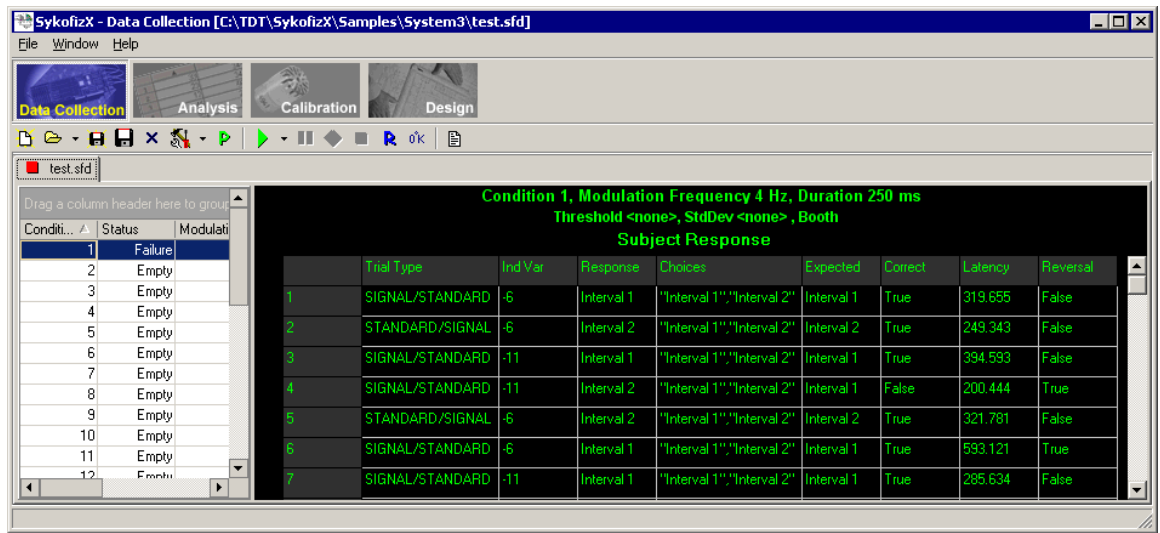
On each trial, the independent variable changes according to the sequence of correct or incorrect responses on previous trials. At design time, the experimenter can specify a wide range of options to customize the adaptive tracking algorithm as summarized below. The average and standard deviation is computed on the basis of the last even number of reversals in independent variable value excluding the discarded reversals.

For example, if the number of discarded reversals is set to 3 and a total of 10 reversals were recorded in a given run, threshold would be based on the last 6 reversals (discarding the first three plus the fourth since it would lead to an odd number of reversals).

Parameter	Description/Usage
Stopping rule	Default: Fixed number of trials - Determines the end of the adaptive track (Fixed number of trials or Fixed number of reversals).
Total Trials	Default: 20 - Required number of trials for Fixed number of trials stopping rule. Minimum number of trials for Fixed number of reversals stopping rule. <b>Note:</b> The application gives priority to total trials over minimum required reversals. If the minimum required reversals are not recorded before the total trials, the condition is recorded as a "failure".
Minimum required reversals	Default: 4 - Required number of reversals for Fixed number of reversals stopping rule. Minimum number of reversals for Fixed number of trials stopping rule.

Number of discarded reversals	Default: 3 - Number of reversals ignored in threshold/convergence computation.
Number of big step reversals	Default: 3 - Number of reversals using a large step size (0 - n).
Step type	Default: Additive - Manner in which independent variable step is implemented: Additive, Multiplicative, Points per octave, Points per decade.
Big step	Default: 5 - Size of big independent variable increment or decrement required in adaptive algorithm.
Small step	Default: 1 - Size of big independent variable increment or decrement required in adaptive algorithm.
Up rule	Default: 1 - Number of incorrect trials required for an increase in independent variable value.
Down rule	Default: 1 - Number of incorrect trials required for a decrease in independent variable value.
Initial value	Default: 0 - Initial independent variable value.
Minimum value	Default: 0 - Minimum independent variable value allowed.
Maximum value	Default: 100 - Maximum independent variable value allowed.

The application stores a detailed trial-by-trial history of the adaptive track along with the final average and standard deviation of the independent values corresponding to reversals. The display in data collection illustrates the summary and trial history available at run-time, stored in the data file, and available for export to standard file formats.



Trial-by-trial data includes:

- Trial number
- Trial Type: Stimulus class by presentation interval
- Ind Var: Independent variable value
- Response: Subject response
- Choices: Available response choices
- Expected: Expected (correct) subject response
- Correct: True for correct and false for incorrect responses
- Latency: Response Latency (ms) based on response timing thread using Microsoft precision timer (< 1 ms accuracy)
- Reversal: True if a reversal occurred, false otherwise

### Adaptive\_Tracking.DiscreteAdjustment

On each trial, the subject must indicate whether the independent variable should be increased, decreased, or meets some criterion as per experimenter instructions. This plugin can be used to create both matching as well as a step-wise method of adjustment algorithms. By creating very short duration stimuli, and short inter-stimulus intervals, one can roughly approximate a continuous method of adjustment. Trial-by-trial and summary data are identical to the data displayed and recorded in the Adaptive\_Tracking.UpDown plugin.

Parameter	Description/Usage
Stopping rule	Default: Fixed number of trials - Determines the end of the adaptive track (Fixed number of trials or Fixed number of reversals). Choosing Fixed number of reversals adds the Number of sub-tracks parameter.
Total trials	Default: 20 - Required number of trials for Fixed number of trials stopping rule. Minimum number of trials for Fixed number of reversals stopping rule. <b>Note:</b> The application gives priority to total trials over minimum required reversals. If the minimum required reversals are not recorded before the total trials, the condition is recorded as a "failure".
Number of sub-tracks	Default: 4 - Number of sub-tracks required when fixed number of reversals is chosen.
Adjust IV based on open response	Default: False - When true, allows the subject to specify the value of the step size for the next trial.
Step type	Default: Additive - Choose Additive, Multiplicative, Points per octave, Points per decade. Determines manner in which IV will be changed on the basis of correct/incorrect responses.
Step size	Default: 5 - Size of independent variable increment or decrement required in adaptive algorithm. This option is not available when the "Adjust IV based on open response" option is True.

Initial value	Default: 0 - Initial independent variable value.
Minimum value	Default: 0 - Minimum independent variable value allowed.
Maximum value	Default: 100 - Maximum independent variable value allowed.

### **Adaptive\_Tracking.Method\_of\_Limits**

Similar to the UpDown tracking plugin, the Method of Limits plugin determines the independent variable value on the basis of previous subject responses. In this case, however, the direction of change (increase or decrease) of the independent variable is fixed during an experimental run. Experimental runs alternate between ascending and descending (1/2 runs are ascending, 1/2 runs are descending). The initial independent variable value for each run is set to the minimum value for ascending runs and the maximum value for descending runs. Trial-by-trial and summary data are identical to the data displayed and recorded in the Adaptive\_Tracking.UpDown plugin.

<b>Parameter</b>	<b>Description/Usage</b>
Stopping rule	Default: Fixed number of series - Determines the end of the adaptive track: either fixed number of trials or fixed number of (ascending and descending) series.
Maximum number of trials	Default: 20 - Required number of trials for Fixed number of trials stopping rule. Minimum number of trials for Fixed number of reversals stopping rule. <b>Note:</b> The application gives priority to total trials over minimum required reversals. If the minimum required reversals are not recorded before the total trials, the condition is recorded as a "failure".
Number of series	Default: 4 - Parameter is available when "Fixed number of (ascending and descending) series" option is chosen. Total number of series including ascending and descending series (1/2 series are ascending and 1/2 are descending).
Step type	Default: Additive - Choose Additive, Multiplicative, Points per octave, or Points per decade. Determines manner in which IV will be changed on the basis of correct/incorrect responses.
Step size	Default: 5 - Size of independent variable increment or decrement required in adaptive algorithm.
Minimum value	Default: 0 - Minimum independent variable value allowed.
Maximum value	Default: 100 - Maximum independent variable value allowed.

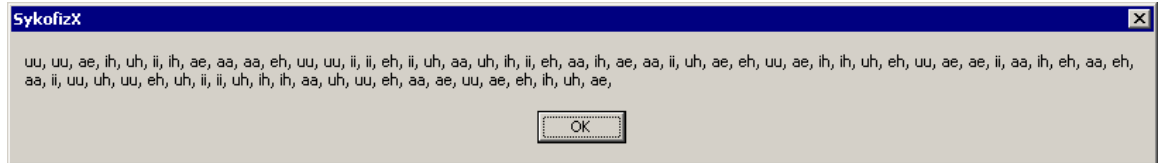
## SimpleVarGen.VarGen

The VarGen plugin is used to generate a list of independent variable values and is used when independent variables values are to be selected from a fixed set rather than by an adaptive algorithm. The VarGen plugin is used with techniques such as the method of constant stimulus, identification, classification, and various others. The format for displaying data during data collection can be specified here as either a histogram (e.g., method of constant stimulus), a table (e.g., confusion matrix), or a list (classification or identification). In many cases, multiple display formats are available (see image below). Default display types are set at runtime but may be changed during data collection or analysis from a window launched by pressing the right mouse button when the mouse cursor is hovering over the display itself.

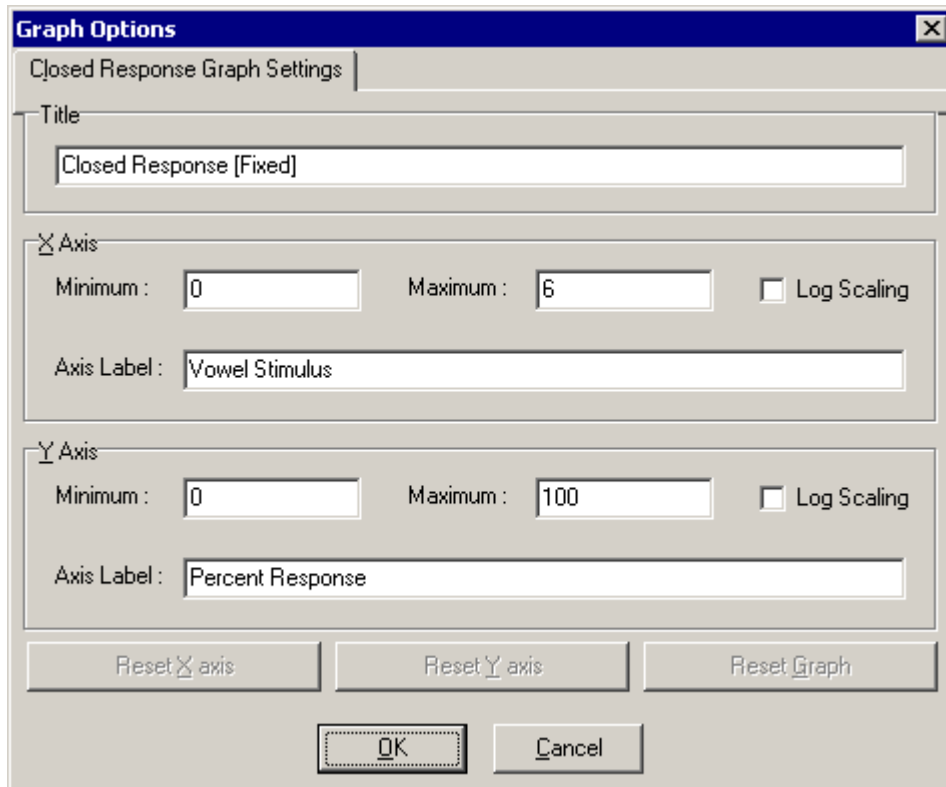
Parameter	Description/Usage
Minimum value	Numeric Default: 0 - String Default: first element in the parameter list corresponding to the independent variable (defined in the Stimulus Configuration wizard). Minimum independent variable value allowed.
Maximum value	Numeric Default: 0 - String Default: last element in the parameter list corresponding to the independent variable (defined in the Stimulus Configuration wizard). Maximum independent variable value allowed.
Step type	Default: Additive - Choose Additive, Multiplicative, Points per octave, or Points per decade. Determines manner in which IV will be changed on the basis of correct/incorrect responses.
Step size	Default: 5 - Size of independent variable increment or decrement required in adaptive algorithm.

Repetitions	Default: 10 - Number of times each independent variable is repeated during a single block of trials.
Maximum # of same item	Default: 3 - Maximum number of times the same item can be presented successively within a single block of trials. The sequence of independent variables to be presented is determined at the beginning of each block of trials.  This parameter is used, in concert with the Maximum retries parameter (see below), to restrict the number of times the same independent variable is presented successively.
Maximum retries	Default: 10 - Maximum number of times the application will attempt to create a new list of independent variable sequences to meet the criterion maximum number of same items allowed. If the criterion is not achieved, the last sequence is used for the current block of trials.

The Preview Variables button allows one to view random a list of independent variables based on the current values set in the parameter inspector. A sample list is shown below.



The Graph Options, such as the axis labels and ranges, may be modified by selecting the Graph Options button and setting the options on the subsequent display window, as shown below. This button is only active when the display options include histograms or scatter plots.



**Note:** In some experiments, it may be desirable to use only one independent variable per run (block of trials). In these cases, a dummy independent variable must be specified.



For example, in a Same/Different paradigm one might want to measure the percent correct same/different judgments for a single "target" value in a given run. In this case, one could create a bogus parameter and use that parameter as the independent variable to be manipulated. Because the bogus parameter is not referenced anywhere else in the design, its current, previous, or future values are irrelevant.

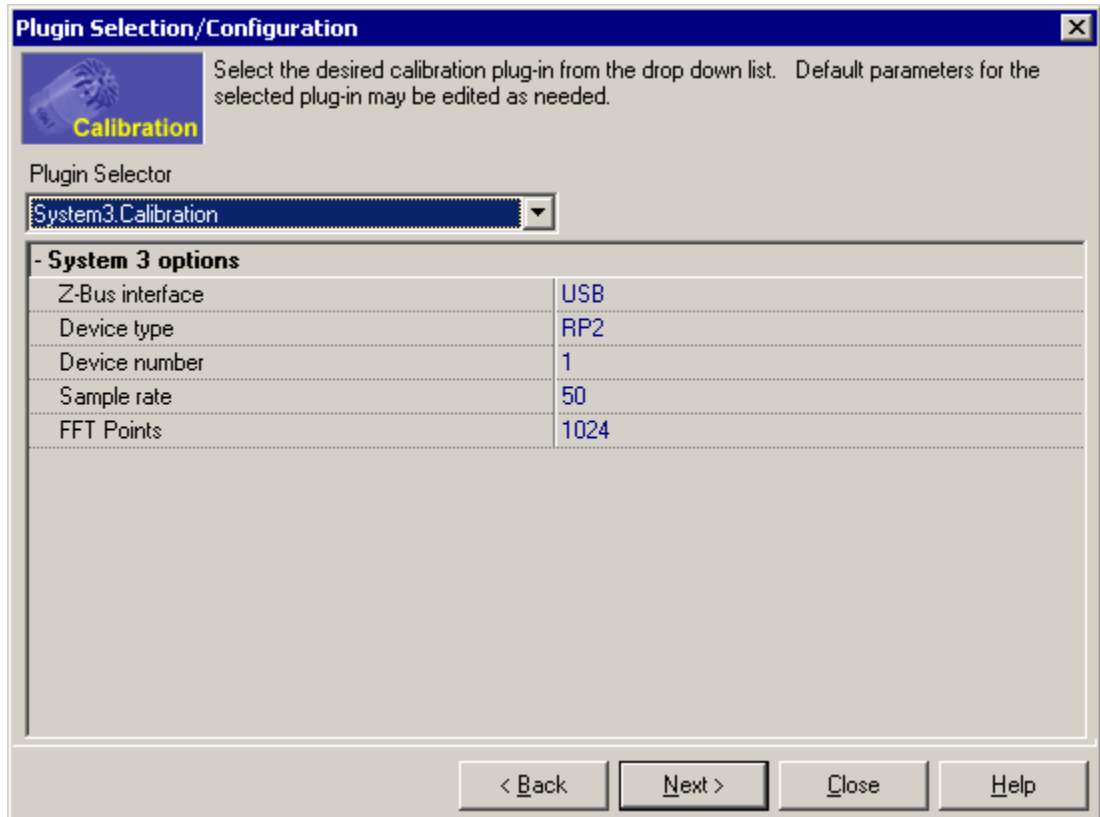
## Calibration

The Calibration Mode provides the SykofizX application with a powerful yet elegant way to calibrate transducers, to store and retrieve calibration data, and to use the calibration data during stimulus generation to ensure that each stimulus presented has the desired characteristics. A detailed description of the calibration process and ways to use the calibration data are described in the Calibration Mode section, page 29. Below is a description of the two available calibration plugins.

### System 3

**Plugin:** System3.Calibration

The System3.Calibration plugin is used when calibrating transducers using TDT System 3 hardware such as the RP<sub>x</sub>, RM<sub>x</sub>, and RX<sub>n</sub> devices. Once the correct plugin is chosen, one may modify the default parameter settings displayed in the parameter inspector.

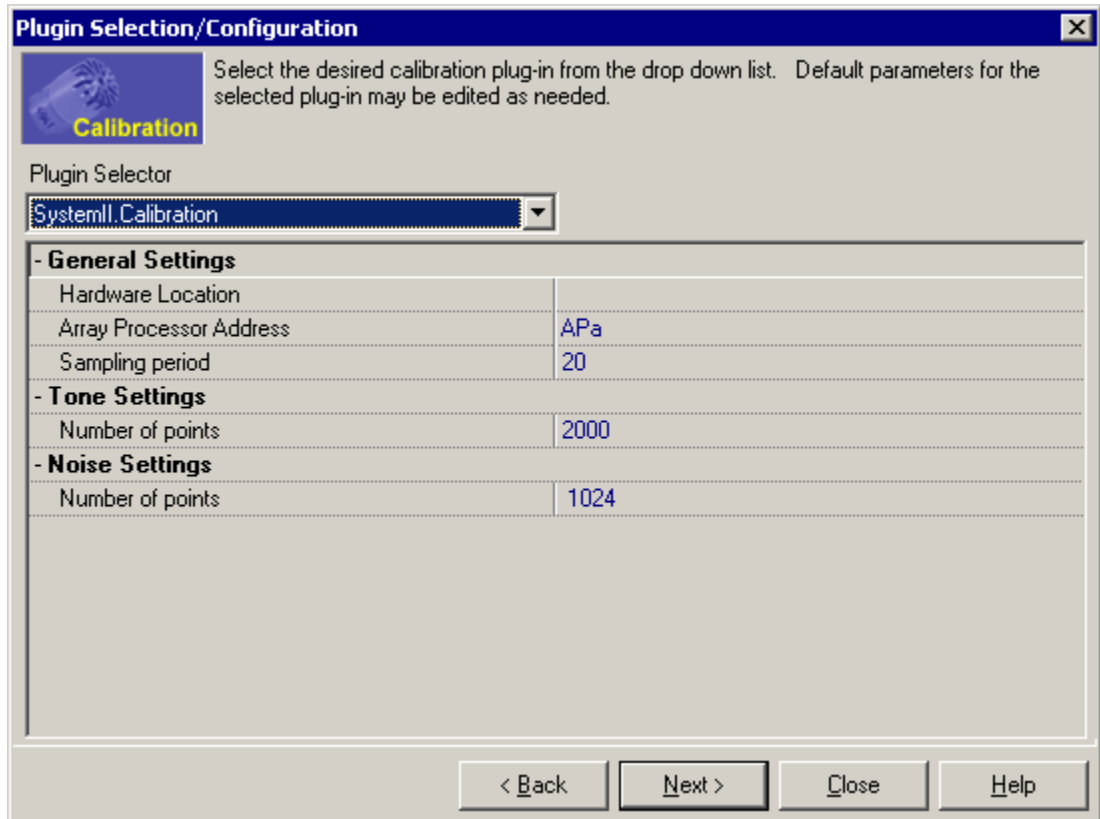


Parameter	Description/Usage
ZBus interface	Default: USB - Both USB and Gigabit interfaces are supported. The default interface may be set at design time, however, that interface can be changed at run time as needed.
Device type	Default: RP2 - The plugin supports the Rx processing family including the RP2, RM1, RM2, RL2, RV8, and RA16 Realtime Processor devices.
Device number	Default: 1 - Specifies the device number as registered by TDT System 3 drivers.
Sample rate	Defaults to 50 kHz with optional rates of 6, 12, 25, 100, and 200 kHz - By convention, the actual sample rates differ from these nominal rates. See the RpvdsEx Manual for more information.
FFT Points	Default: 1024 - Number of points in FFT if broadband computations are used.

## System II

**Plugin:** SystemII.Calibration

The SystemII.Calibration plugin is used when calibrating transducers using TDT System II hardware such as the AP2 Array Processor and associated XBus devices. Once the correct plugin is chosen, one may modify the default parameter settings displayed in the parameter inspector.




Parameter	Description/Usage
Hardware Location	Default: None - Specify location of remote PC and hardware in a Local Area Network (LAN).
Array Processor Address	Default: APa - Choices are APa and APb. See System II support documentation for further information about AP addresses.
Sampling Period	Default: 20 - Sampling period in microseconds.
Tone: Number of points	Default: 2000 - With the sampling period determines duration of tonal stimuli.
Noise: Number of points	Default: 1024 - With the sampling period determines duration of noise stimuli. Also represents number of points in FFT and thus must be a radix-2 number.

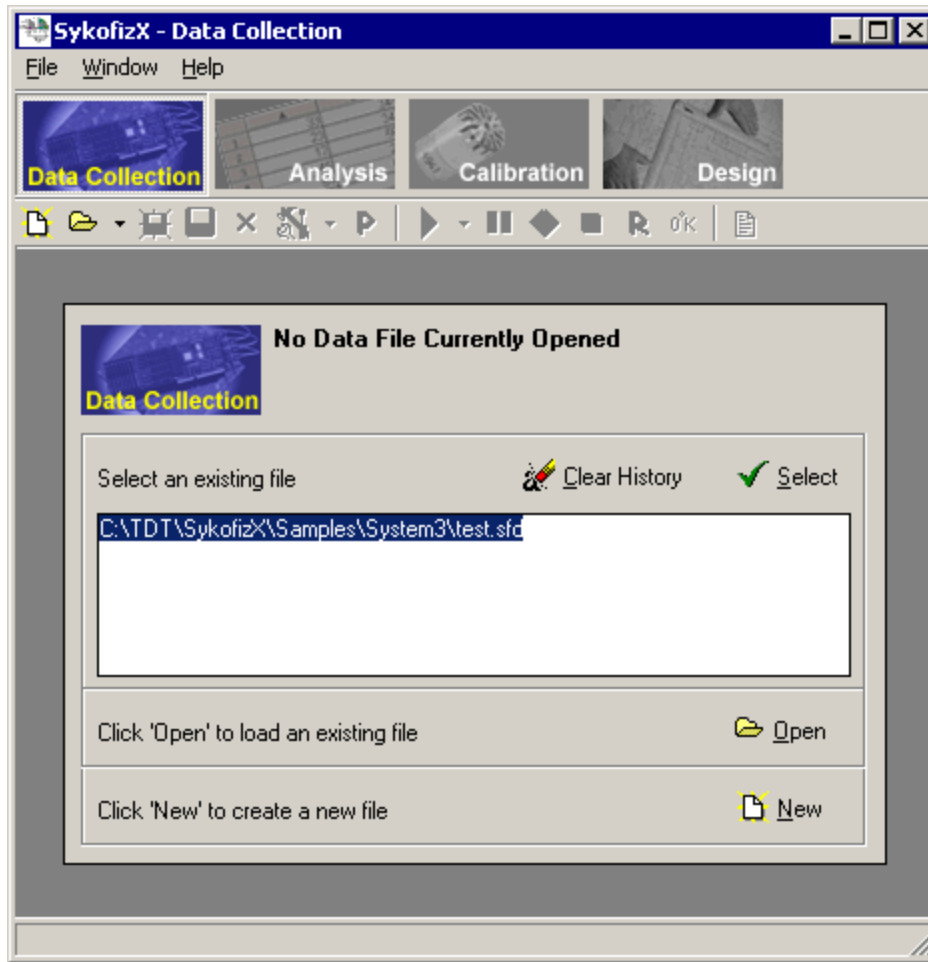
## Data Analysis

The Data Analysis mode of SykofizX provides a convenient environment for reviewing subject data and exporting that data for use by other applications. Data for all conditions in a subject data file (.sfd) may be viewed in much the same way it is viewed in Data Collection mode. Furthermore, the data may be exported as one of several file structures and saved in one of three file formats (.txt, .xls, and .xml).

### Loading a Subject Data File

An existing subject data file (.sfd) may be selected in one of three ways

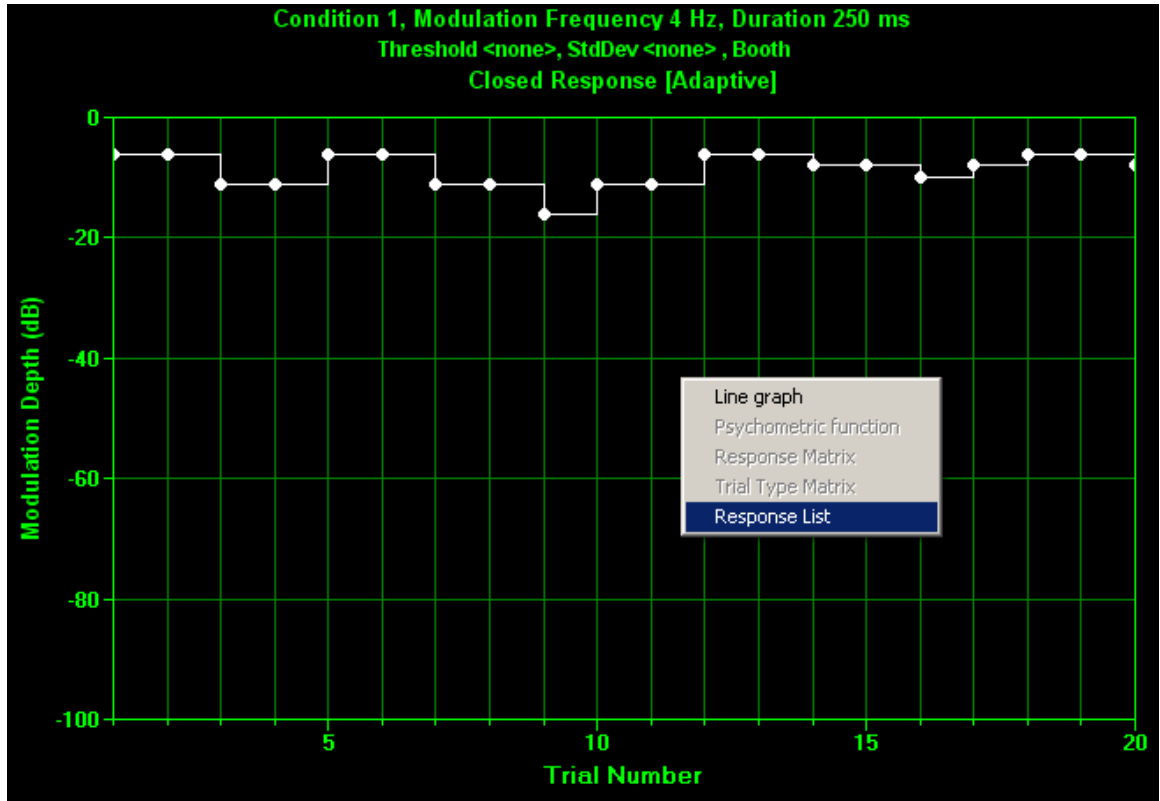
- From the window in the center of the main application window, choose **Open** and navigate to the desired file.
- From the window in the center of the main application window, choose a file from the **Select an existing file** list.
- From the main application window, click the **Open** button  on the control toolbar and navigate to the desired file.



## Navigating a Subject Data File

Use the cursor or keypad to navigate the conditions listed in the condition grid. Conditions may be grouped for more convenient analysis and export as shown below. Right-clicking the data display launches a list of display options as in the Data Collection mode. Simply choose a different display option from the list to change the current display.

ConditionID	Modulation Frequency	Duration	Run_Number	Sample Trial Value
Status : Empty				
Status : Success				
2	4	500	1	-6
Status : Failure				
1	4	250	1	-6



## Exporting Data from a Subject Data File

Data may be exported from a SykofizX Data File (.sfd) in a number of different formats.

### Selecting data one or more conditions for export

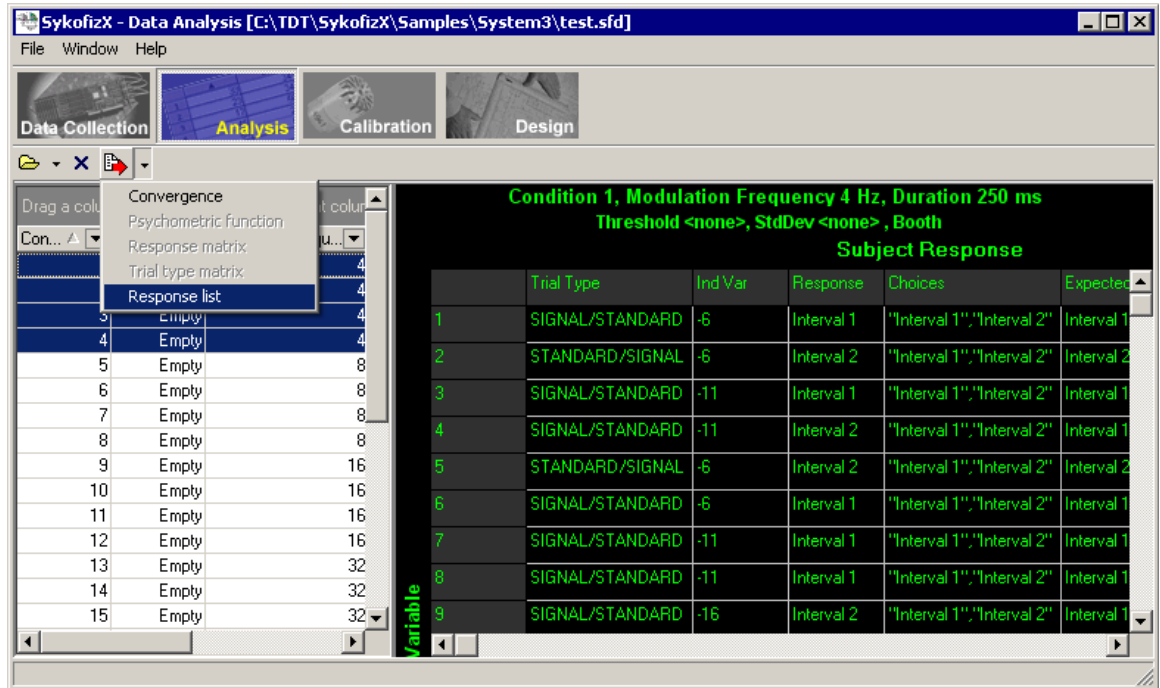
Simply use the cursor to highlight the conditions in the condition grid for which data is to be exported. A range of conditions may be highlighted by selecting the first condition in the range, pressing and holding the shift key, and selecting the last condition in the range. Non-contiguous conditions may be selected by selecting the first condition, pressing and holding the control key, and selecting subsequent conditions.

### Selecting the export data structure

There are five possible data structures, however, only the data structures that are appropriate for a given design are enabled.

Data Structure	Description
Convergence	Exports mean and standard deviation of adaptive track only. For each condition, condition information is exported along with summary data for each condition (e.g., mean and standard deviation of the threshold for an adaptive track).
Psychometric function	Exports percent correct for each independent variable value. For each condition, condition information as well as a table of values displaying percent correct corresponding to each independent variable value is exported.

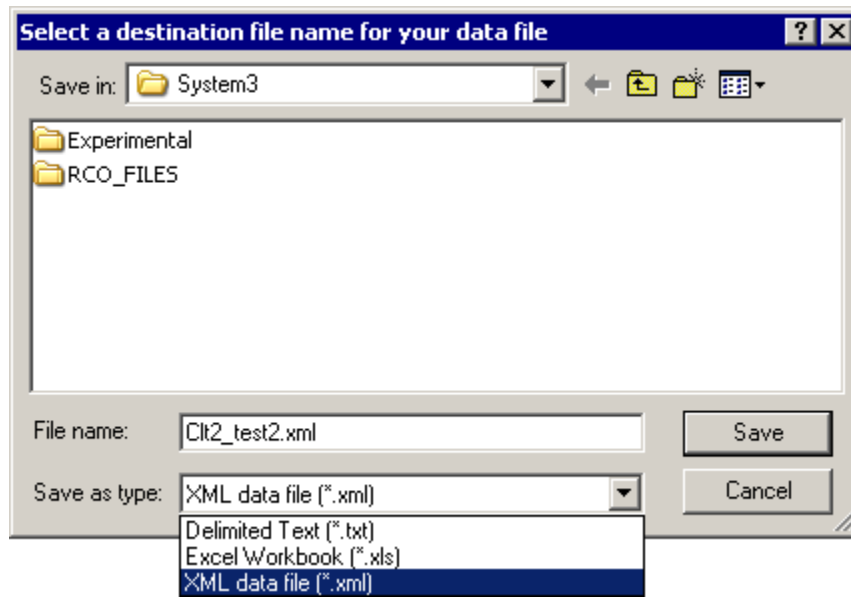
Response matrix	Exports frequency count for each stimulus/response type. For each condition, condition information as well as a table of values displaying frequency count corresponding to each stimulus/response type is exported.
Trial type matrix	Exports frequency count for each trial type. For each condition, condition information as well as a table of values displaying frequency count corresponding to each trial type is exported.
Response list	Exports trial-by-trial information for each condition. For each condition, condition information as well as a complete record of all of the trial-by-trial data shown in the response list as displayed in the Analysis or Data Collection modes is exported.



## Selecting a File Type

Three different file types are available for exporting.

- Comma delimited text (.txt) file. This file contains is organized according to the chosen data structure. File and condition information is arranged in a header and actual data is displayed below. String items are surrounded by double quotation marks. Data for multiple conditions are identified by condition number.
- Excel (.xls) workbook. Data are exported to a Microsoft Excel (.xls) workbook. For the convergence structure, data for different conditions are stored on different rows in the workbook. For all other structures, data for different conditions are stored on different worksheets labeled with the appropriate condition number.
- Extensible Markup Language (.xml). Data are exported in .xml file format, convenient for importing to other databases as well as for viewing in an internet browser. Data are organized in much the same way the data are stored in the subject data file.



~



# Sample Designs

Several sample designs are included with SykofizX to illustrate the various application features. These samples also can be used as templates for creating similar experimental designs. The stimulus scripts and subject interface scripts for each example have been exported for easy use by other designs.



**Important!** All System 3 examples are designed by default to work with an RM1 and USB interface.

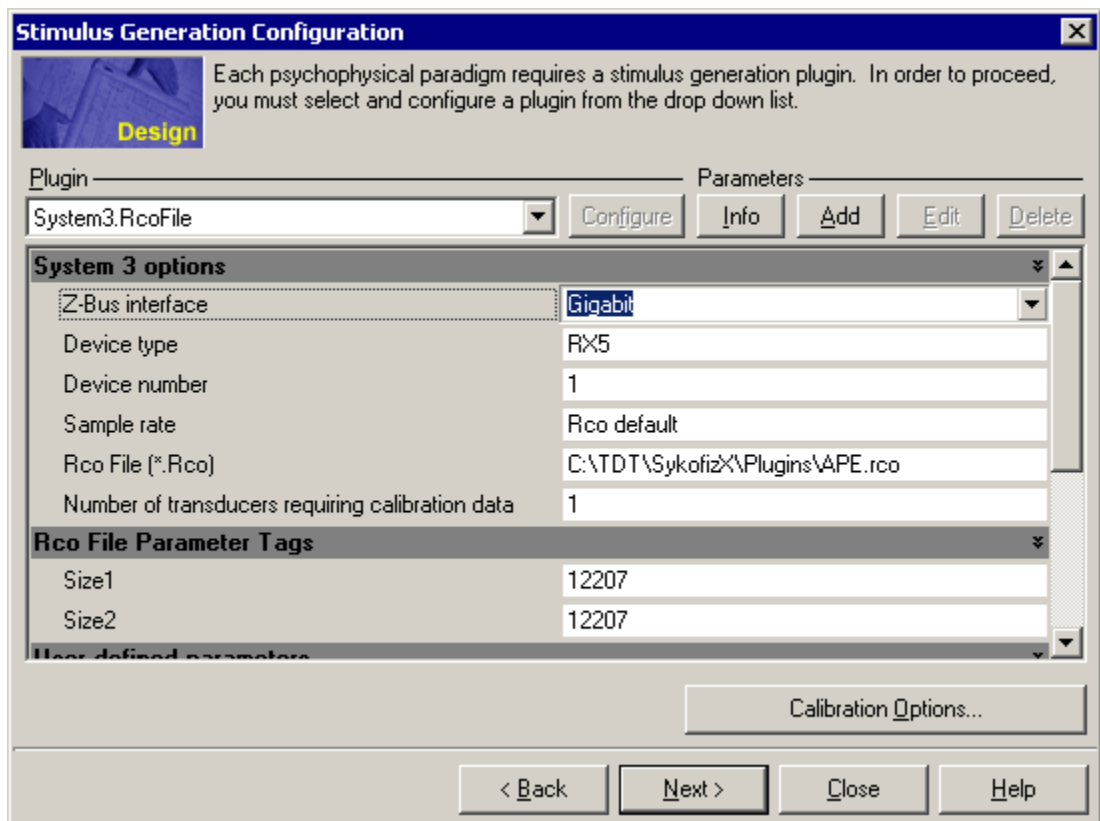
## Hardware Settings

SykofizX supports a wide variety of System II and System 3 hardware. Before attempting to run an experiment, ensure that your hardware is correctly configured within the Stimulus Generation Configuration window.

### Configuring Optibit and Gigabit Interfaces

Both Optibit and Gigabit interfaces are supported by SykofizX. To configure hardware for these interfaces:

1. Click the arrow and select **Stimulus Generation Configuration** or **Stimulation Generation Utilities** from the **Configure** or **Evaluation** buttons (  or  ).
2. Select **Gigabit** from the Z-Bus interface option in the Stimulus Generation Configuration dialog box as shown below.



In addition to the interface configuration are options for selecting the device type and number.

## System 3

### Example 1 - Amplitude Modulation Detection Using a Forced Choice Paradigm and Keyboard-Video-Mouse

**File Name**

Ex01\_S3\_Ape\_AMD\_comp\_FC\_UD\_KVM.sfx

**Description**

Sinusoidal amplitude modulation detection using a broadband noise carrier. Presented with a two-interval, forced-choice adaptive tracking paradigm. Subject response via a keyboard-video-mouse (KVM) subject interface designed to mimic a two-button response box.

The listener hears two sounds, an unmodulated noise and a sinusoidally amplitude modulated noise, presented in random order and separated by a silent period. The listener must indicate, by button press, which interval was amplitude modulated. Presentation intervals are marked by LEDs and feedback is provided indicating correct/incorrect responses.

**RPvdsEx Circuit:** APE.rco

**Stimulus Scripts:** Ex01stim.sc

**Calibration file:** DE\_RE\_AD\_ER2\_test1.bcf

**Subject Interface Scripts:** TwoButtonKVM.sc

**Subject Interface Configuration:** TwoButtonKvm.kvm

**Stimulus Generation Configuration**

Stimulus generation is based on APOS commands and the APE (Array Processor Emulator). The stimulus generation scripts used in this design were written in VBScript using the SykofizX scripting interface. Necessary parameters to define: Rise/Fall time and Overall Level.

**Presentation Paradigm**

A forced choice presentation with two intervals (signal and standard presented).

**Subject Interface Configuration**

Subject response via a keyboard-video-mouse (KVM) interface designed to mimic a two-button response box. The subject interface plugin uses a configuration file created using the KVM configuration utility available whenever the KVM.KeyboardVideoMouse plugin is selected. Properties of the KVM display elements are set in the object inspector within the configuration utility. Actions associated with these elements are defined in the TwoButtonKVM.sc script library.

**Stimulus Presentation Timing**

2000 ms Initial Delay used as Inter-Trial Interval, 400 ms Inter-Stimulus Interval.

**Subject Response Timing**

10000 ms response window with feedback after the response. Response time begins at end of 2nd interval.

**Define Experimental Variables**

Independent variable: modulation depth. Condition variables: modulation frequency and stimulus duration.

**Independent Variable Configuration**

Adaptive Tracking Up/Down with 2-Down and 1-Up adaptive rule. Fixed number of trials set to 20.

## Example 2 - Amplitude Modulation Detection Using a Forced Choice Paradigm and Lowpass Filtering

### File Name

Ex02\_S3\_Ape\_AMD\_Comp\_LPF\_FC\_UD\_KVM.sfx

### Description

Sinusoidal amplitude modulation detection using a broadband noise. Presented with two-interval, forced-choice adaptive tracking paradigm. Subject response via a keyboard-video-mouse (KVM) interface designed to mimic a two-button response box.

The listener will hear two sounds, an unmodulated noise and a sinusoidally amplitude modulated noise, presented in random order and separated by a silent period. The listener must indicate, by button press, which interval was amplitude modulated. Presentation intervals are marked by LEDs and feedback is provided indicating correct/incorrect responses.

Example 2 differs from Example 1 in terms of the stimulus generation and experimental variables. In Example 2, the modulated broadband noise is filtered by one of several lowpass filters specified by filter coefficients read from data files. Details regarding filtering using the FIR function may be found in the APOS Reference available at [www.TDT.com](http://www.TDT.com). The four lowpass filter parameters are used as condition variables in the experimental variables wizard.

**RPvdsEx Circuit:** APE.rco

**Stimulus Scripts:** Ex02stim.sc

**Calibration file:** DE\_RE\_AD\_ER2\_test1.bcf

**Subject Interface Scripts:** TwoButtonKVM.sc

**Subject Interface Configuration:** TwoButtonKvm.kvm

### Stimulus Generation Configuration

Sinusoidal amplitude modulation detection using a broadband noise. Stimulus generation is based on APOS commands and the APE (Array Processor Emulator). The stimulus generation scripts used in this design were written in VBScript using the SykofizX scripting interface. Necessary parameters to define: Rise/Fall time and Overall Level.

### Presentation Paradigm

A forced choice presentation with two intervals (signal and standard presented).

### Subject Interface Configuration

Subject response via a keyboard-video-mouse (KVM) interface designed to mimic a two-button response box. The subject interface plugin uses a configuration file created using the KVM configuration utility available whenever the KVM.KeyboardVideoMouse plugin is selected. Properties of the KVM display elements are set in the object inspector within the configuration utility. Actions associated with these elements are defined in the TwoButtonKVM.sc script library which has been exported for easy use by other designs.

### Stimulus Presentation Timing

2000 ms Initial Delay used as Inter-Trial Interval, 500 ms Inter-Stimulus Interval.

### Subject Response Timing

10000 ms response window with feedback after response.

### Define Experimental Variables

Independent variable: modulation depth. Condition variables: modulation frequency, stimulus duration, and low pass filter.

### Independent Variable Configuration

Adaptive Tracking Up/Down with 2-Down and 1-Up adaptive rule. Fixed number of trials set to 20.

## Example 3 - Amplitude Modulation Detection Using a Forced Choice Paradigm and Response Box

### File Name

Ex03\_S3\_Ape\_AMD\_comp\_FC\_UD\_RBox.sfx

### Description

Sinusoidal amplitude modulation detection using a broadband noise carrier. Presented with a two-interval, forced-choice adaptive tracking paradigm. Subject response via a two-button subject response box.

The listener hears two sounds, an unmodulated noise and a sinusoidally amplitude modulated noise, presented in random order and separated by a silent period. The listener must indicate, by button press, which interval was amplitude modulated. Presentation intervals are marked by LEDs and feedback is provided indicating correct/incorrect responses.

Example 3 is the same as Example 1 except that Example 3 uses the System3.ScriptedSubjectInterface plugin with an actual response box rather than the KVM.KeyboardVideoMouse plugin.

**RPvdsEx Circuit:** APE.rco

**Stimulus Scripts:** Ex01stim.sc

**Calibration file:** DE\_RE\_AD\_ER2\_test1.bcf

**Subject Interface Scripts:** S3\_Rpx\_RBox\_nButton.sc

### Stimulus Generation Configuration

Sinusoidal amplitude modulation detection using a broadband noise carrier. Stimulus generation is based on APOS commands and the APE (Array Processor Emulator). The stimulus generation scripts used in this design were written in VBScript using the SykofizX scripting interface.

Necessary parameters to define: Rise/Fall time and Overall Level.

### Presentation Paradigm

A forced choice presentation with two intervals (signal and standard presented).

### Subject Interface Configuration

Subject response via a two-button subject response box. Actions associated with the response box are defined in the Scripts S3\_Rpx\_RBox\_nButton.sc. The response box is a function of the System 3 Scripted Subject Interface and the scripts specify the RP2 digital I/O logic necessary to control the standard TDT two-button response box with LED interval markers (or the first two buttons on a response box with more than two sets of buttons and LEDs).

### Stimulus Presentation Timing

2000 ms Initial Delay used as Inter-Trial Interval, 400 ms Inter-Stimulus Interval.

### Subject Response Timing

10000 ms response window with feedback after response.

### Define Experimental Variables

Independent variable: modulation depth. Condition variables: modulation frequency and stimulus duration.

### Independent Variable Configuration

Adaptive Tracking Up/Down with 2-Down and 1-Up adaptive rule. Fixed number of trials set to 20.

## Example 4 - Amplitude Modulation Detection Using a Same-Different Paradigm and Response Box

### File Name

Ex04\_S3\_Ape\_AMD\_comp\_SD\_VG\_RBox.sfx

### Description

Sinusoidal amplitude modulation detection using a broadband noise carrier. Presented with same-different response paradigm, the method of constant stimuli. Subject response via a two-button subject response box.

The listener will hear two sounds, an unmodulated noise (A) or a sinusoidally amplitude modulated noise (B). The two sounds on a given trial will take the form AA, AB, BA, or BB. The listener must indicate, by button press, whether the two sounds were the Same or Different. Presentation intervals are marked by LEDs and feedback is provided indicating correct/incorrect responses.

Example 4 is the same as Example 3 except that Example 4 is based on the same-different presentation paradigm and the SimpleVarGen.VarGen independent variable configuration plugin. The VarGen plugin presents fixed set of independent variable values within a specified range to be presented n times, resulting in a percent correct score for each independent variable value.

**RPvdsEx Circuit:** APE.rco

**Stimulus Scripts:** Ex01stim.sc

**Calibration file:** DE\_RE\_AD\_ER2\_test1.bcf

**Subject Interface Scripts:** S3\_Rpx\_RBox\_nButton.sc

### Stimulus Generation Configuration

Sinusoidal amplitude modulation detection using a broadband noise carrier. Stimulus generation is based on APOS commands and the APE (Array Processor Emulator). The stimulus generation scripts used in this design were written in VBScript using the SykofizX scripting interface.

Necessary parameters to define: Rise/Fall time and Overall Level.

### Presentation Paradigm

A same-different presentation paradigm with two intervals.

### Subject Interface Configuration

Subject response via a two-button subject response box. Actions associated with the response box are defined in the Scripts S3\_Rpx\_RBox\_nButton.sc. The response box is a function of the System 3 Scripted Subject Interface and the scripts specify the RP2 digital I/O logic necessary to control the standard TDT same-different response box with LED interval markers (or the first two buttons on a response box with more than two sets of buttons and LEDs).

### Stimulus Presentation Timing

2000 ms Initial Delay used as Inter-Trial Interval, 400 ms Inter-Stimulus Interval.

### Subject Response Timing

10000 ms response window with feedback after response.

### Define Experimental Variables

Independent variable: modulation depth. Condition variables: modulation frequency and stimulus duration.

### Independent Variable Configuration

Simple variable generation via the SimpleVarGen.VarGen Configuration. The VarGen plugin presents fixed set of independent variable values within a specified range to be presented n times, resulting in a percent correct score for each independent variable value.

## Example 5 - Amplitude Modulation Detection Using an RCO File

### File Name

Ex05\_S3\_RCO\_AMD\_comp\_FC\_UD\_KVM.sfx

### Description

Sinusoidal amplitude modulation detection using a broadband noise carrier. Presented with a two-interval, forced-choice adaptive tracking paradigm. Subject response via a keyboard-video-mouse (KVM) subject interface designed to mimic a two-button response box.

The listener hears two sounds, an unmodulated noise and a sinusoidally amplitude modulated noise, presented in random order and separated by a silent period. The listener must indicate, by button press, which interval was amplitude modulated. Presentation intervals are marked by LEDs and feedback is provided indicating correct/incorrect responses.

The only difference between Example 5 and Example 1 is the use of a dedicated RCO file rather than the Array Processor Emulator (APE.rco) in conjunction with scripting containing APOS commands.

**RPvdsEx Circuit:** AMNoise\_Template.rco

**Stimulus Scripts:** Ex05stim.sc

**Calibration file:** DE\_RE\_AD\_ER2\_test1.bcf

**Subject Interface Scripts:** TwoButtonKVM.sc

**Subject Interface Configuration:** TwoButtonKvm.kvm

### Stimulus Generation Configuration

Stimulus generation is based on an RCO circuit design in RPvdsEx . The circuit generates the unmodulated and amplitude modulated signals and includes parameter tags for specifying the stimulus parameters set by the SykofizX application and parameter tags for triggering the appropriate D/A channels. The stimulus generation scripts used in this design were written in VBScript using the SykofizX scripting interface. Necessary parameters to define: CosRFTIMEr, DA\_Out, ModDepPerR, Stim1Scalar, and Overall Level.

### Presentation Paradigm

A forced choice presentation with two intervals (signal and standard presented).

### Subject Interface Configuration

Subject response via a keyboard-video-mouse (KVM) interface designed to mimic a two-button response box. The subject interface plugin uses a configuration file created using the KVM configuration utility available whenever the KVM.KeyboardVideoMouse plugin is selected.

Properties of the KVM display elements are set in the object inspector within the configuration utility. Actions associated with these elements are defined in the TwoButtonKVM.sc script library.

### Stimulus Presentation Timing

1000 ms plus Initial Delay plus DurationR used as Inter-Trial Interval, 500 ms Inter-Stimulus Interval.

### Subject Response Timing

10000 ms response window with feedback after the response. Response time begins at end of 2nd interval.

### Define Experimental Variables

Independent variable: modulation depth. Condition variables: modulation frequency and stimulus duration.

#### **Independent Variable Configuration**

Adaptive Tracking Up/Down with 2-Down and 1-Up adaptive rule. Fixed number of trials set to 50.

## **Example 6 - Amplitude Modulation Detection Using a Same-Different Paradigm and Keyboard-Video-Mouse**

#### **File Name**

Ex06\_S3\_Ape\_AMD\_comp\_SD\_VG\_KVM.sfx

#### **Description**

Sinusoidal amplitude modulation detection using a broadband noise carrier. Presented with same-different response paradigm, the method of constant stimuli. Subject response via a keyboard-video-mouse (KVM) subject interface designed to mimic a two-button response box with the labels SAME and DIFFERENT.

The listener will hear two sounds, an unmodulated noise (A) or a sinusoidally amplitude modulated noise (B). The two sounds on a given trial will take the form AA, AB, BA, or BB. The listener must indicate, by button press, whether the two sounds were the Same or Different. Presentation intervals are marked by LEDs and feedback is provided indicating correct/incorrect responses.

Example 6 is similar to Example 4, using the Presentation\_Paradigm.Same\_Different and the SimpleVarGen.VarGen independent variable configuration plugin. The VarGen plugin presents fixed set of independent variable values within a specified range to be presented n times, resulting in a percent correct score for each independent variable value. Example 6 differs from Example 4 in that the two button KVM subject interface is used rather than a response box and digital I/O.

**RPvdsEx Circuit:** APE.rco

**Stimulus Scripts:** Ex01stim.sc

**Calibration file:** DE\_RE\_AD\_ER2\_test1.bcf

**Subject Interface Scripts:** TwoButtonKVM.sc

**Subject Interface Configuration:** TwoButtonKvm.kvm

#### **Stimulus Generation Configuration**

Sinusoidal amplitude modulation detection using a broadband noise carrier. Stimulus generation is based on APOS commands and the APE (Array Processor Emulator). The stimulus generation scripts used in this design were written in VBScript using the SykofizX scripting interface. Necessary parameters to define: Rise/Fall time and Overall Level.

#### **Presentation Paradigm**

A same-different presentation paradigm with two intervals.

#### **Subject Interface Configuration**

Subject response via a keyboard-video-mouse (KVM) interface designed to mimic a two-button response box. The subject interface plugin uses a configuration file created using the KVM configuration utility available whenever the KVM.KeyboardVideoMouse plugin is selected. Properties of the KVM display elements are set in the object inspector within the configuration utility. Actions associated with these elements are defined in the TwoButtonKVM.sc script library.

#### **Stimulus Presentation Timing**

2000 ms Initial Delay used as Inter-Trial Interval, 400 ms Inter-Stimulus Interval.

### **Subject Response Timing**

10000 ms response window with feedback after response.

### **Define Experimental Variables**

Independent variable: modulation depth. Condition variables: modulation frequency and stimulus duration.

### **Independent Variable Configuration**

Simple variable generation via the SimpleVarGen.VarGen Configuration. The VarGen plugin presents fixed set of independent variable values within a specified range to be presented n times, resulting in a percent correct score for each independent variable value.

## **Example 7 - Amplitude Modulation Detection Using a Forced Choice Paradigm and a Reminder Interval**

### **File Name**

Ex07\_S3\_Ape\_AMD\_comp\_FCR\_UD\_KVM.sfx

### **Description**

Sinusoidal amplitude modulation detection using a broadband noise carrier. Presented with a two-interval, forced-choice adaptive tracking paradigm. Subject response via a keyboard-video-mouse (KVM) subject interface designed to mimic a two-button response box.

The listener hears two sounds, an unmodulated noise and a sinusoidally amplitude modulated noise, presented in random order and separated by a silent period. The listener must indicate, by button press, which interval was amplitude modulated. Presentation intervals are marked by LEDs and feedback is provided indicating correct/incorrect responses.

Example 7 differs from Example 1 with the inclusion of a reminder interval beginning each trial. This reminder or "cue" interval is always associated with the standard stimulus class as specified in the stimulus generation plugin.

**RPvdsEx Circuit:** APE.rco

**Stimulus Scripts:** Ex01stim.sc

**Calibration file:** DE\_RE\_AD\_ER2\_test1.bcf

**Subject Interface Scripts:** ThreeButtonKVM.sc

**Subject Interface Configuration:** ThreeButtonKvm.kvm

### **Stimulus Generation Configuration**

Stimulus generation is based on APOS commands and the APE (Array Processor Emulator). The stimulus generation scripts used in this design were written in VBScript using the SykofizX scripting interface. Necessary parameters to define: Rise/Fall time and Overall Level.

### **Presentation Paradigm**

A forced choice presentation with two intervals (signal and standard presented). A reminder stimulus, associated with the standard stimulus class, is also presented before the two interval presentation.

### **Subject Interface Configuration**

Subject response via a keyboard-video-mouse (KVM) interface designed to mimic a two-button response box. The subject interface plugin uses a configuration file created using the KVM configuration utility available whenever the KVM.KeyboardVideoMouse plugin is selected. Properties of the KVM display elements are set in the object inspector within the configuration utility. Actions associated with these elements are defined in the ThreeButtonKVM.sc script library.



**Stimulus Presentation Timing**

1000 ms Initial Delay used as Inter-Trial Interval, 400 ms Inter-Stimulus Interval.

**Subject Response Timing**

10000 ms response window with feedback after the response. Response time begins at end of 2nd interval.

**Define Experimental Variables**

Independent variable: modulation depth. Condition variables: modulation frequency and stimulus duration.

**Independent Variable Configuration**

Adaptive Tracking Up/Down with 2-Down and 1-Up adaptive rule. Fixed number of trials set to 20.

## Example 8 - Amplitude Modulation Detection Using a Yes-No Paradigm and Keyboard-Video-Mouse

**File Name**

Ex08\_S3\_Ape\_AMD\_comp\_YN\_ML\_KVM.sfx

**Description**

Sinusoidal amplitude modulation detection using a broadband noise carrier. Presented with a single interval and yes-no paradigm, method of limits. Subject response via a keyboard-video-mouse (KVM) subject interface designed to mimic a two-button response box with labels of YES and NO.

The listener will hear one sound, either an unmodulated noise or a sinusoidally amplitude modulated noise, presented on a random schedule. The listener must indicate, by button press, whether the sound was amplitude modulated. Presentation intervals are marked by LEDs and feedback is provided indicating correct/incorrect responses.

**RPvdsEx Circuit:** APE.rco

**Stimulus Scripts:** Ex01stim.sc

**Calibration file:** DE\_RE\_AD\_ER2\_test1.bcf

**Subject Interface Scripts:** YesNoKVM.sc

**Subject Interface Configuration:** YesNoKvm.kvm

**Stimulus Generation Configuration**

Stimulus generation is based on APOS commands and the APE (Array Processor Emulator). The stimulus generation scripts used in this design were written in VBScript using the SykofizX scripting interface. Necessary parameters to define: Rise/Fall time and Overall Level.

**Presentation Paradigm**

A yes-no paradigm with a single interval (signal or standard presented). The catch trials option is selected, specifying that the standard (unmodulated) stimulus be presented on a user-specified 25% of the trials.

**Subject Interface Configuration**

Subject response via a keyboard-video-mouse (KVM) interface designed to mimic a two-button response box. The subject interface plugin uses a configuration file created using the KVM configuration utility available whenever the KVM.KeyboardVideoMouse plugin is selected.

Properties of the KVM display elements are set in the object inspector within the configuration utility. Actions associated with these elements are defined in the YesNoKVM.sc script library.

**Stimulus Presentation Timing**

1000 ms Initial Delay used as Inter-Trial Interval, 1000 ms Inter-Stimulus Interval.

**Subject Response Timing**

10000 ms response window with feedback after the response. Response time begins at end of 1st interval.

**Define Experimental Variables**

Independent variable: modulation depth. Condition variables: modulation frequency and stimulus duration.

**Independent Variable Configuration**

Adaptive Tracking Up/Down with 2-Down and 1-Up adaptive rule. Fixed number of trials set to 20.

**Example 9 - Word Recognition Task with Rhyming Foils**

**File Name**

Ex09\_S3\_Ape\_WRT\_Buff\_CSIL\_KVM.sfx

**Description**

Single-interval closed-set identification word recognition test. Each stimulus token has four possible responses including the target and three rhyming foils. The listener hears a single word and then must judge from four possible response alternatives which word was heard.

Subject response via a keyboard-video-mouse (KVM) subject interface designed to mimic a four-button response box with labels for the actual word and the three rhyming foils.

**RPvdsEx Circuit:** APE.rco

**Stimulus Scripts:** Ex09stim.sc

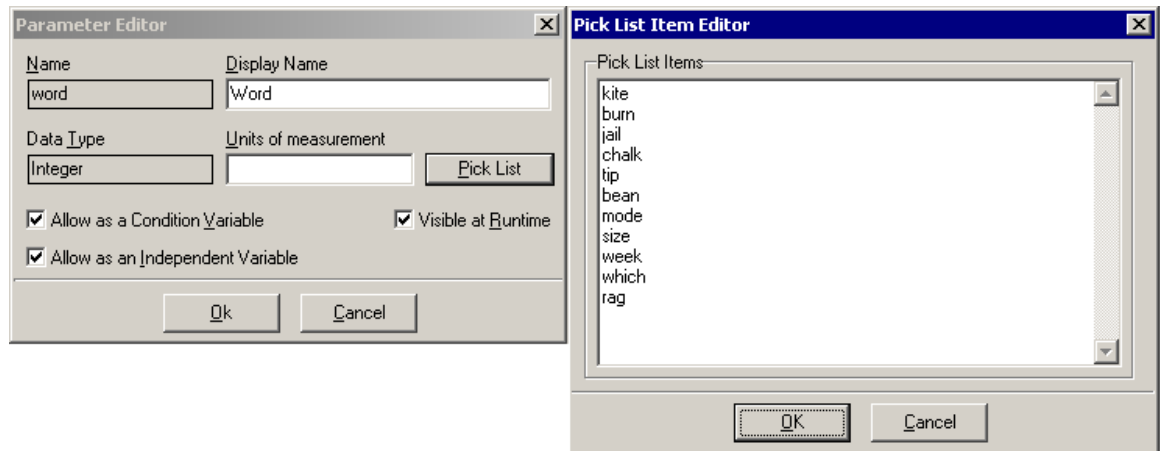
**Calibration file:** DE\_RE\_AD\_ER2\_test1.bcf

**Subject Interface Scripts:** WRT\_KVM.sc

**Subject Interface Configuration:** WRT\_KVM.kvm

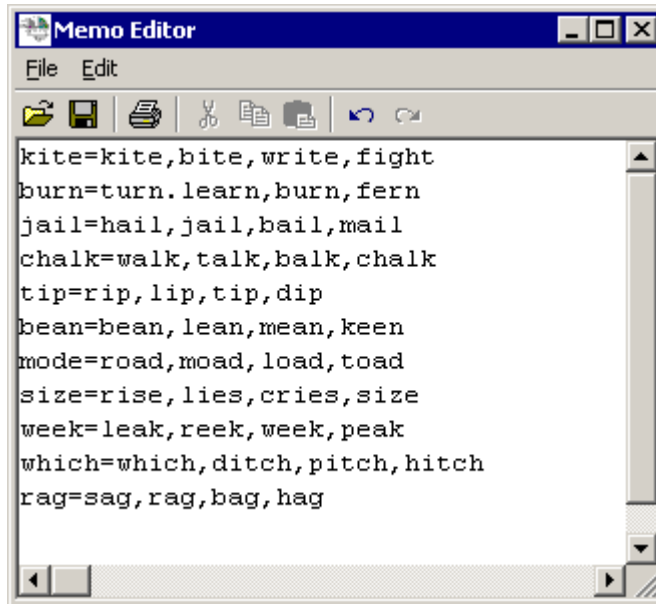
**Stimulus Generation Configuration**

Stimulus generation is based on APOS commands and the APE (Array Processor Emulator). The stimulus generation scripts used in this design were written in VBScript using the SykofizX scripting interface. Necessary parameters to define: Overall Level. Interesting features of this design include the User Parameter "word". This parameter defines a list of stimuli (words) that are the basis for the independent variable specified on the Define Experimental Variables wizard.



### Presentation Paradigm

A closed set identification list paradigm with a single interval. The target and associated foils are defined in the Closed\_Set\_Identification\_List presentation plugin in a plugin-specific memo as shown below.



### Subject Interface Configuration

Subject response via a keyboard-video-mouse (KVM) interface designed to mimic a four-button response box with word labels. The subject interface plugin uses a configuration file created using the KVM configuration utility available whenever the KVM.KeyboardVideoMouse plugin is selected. Properties of the KVM display elements are set in the object inspector within the configuration utility. Actions associated with these elements are defined in the WRT\_KVM.sc script library.

### Stimulus Presentation Timing

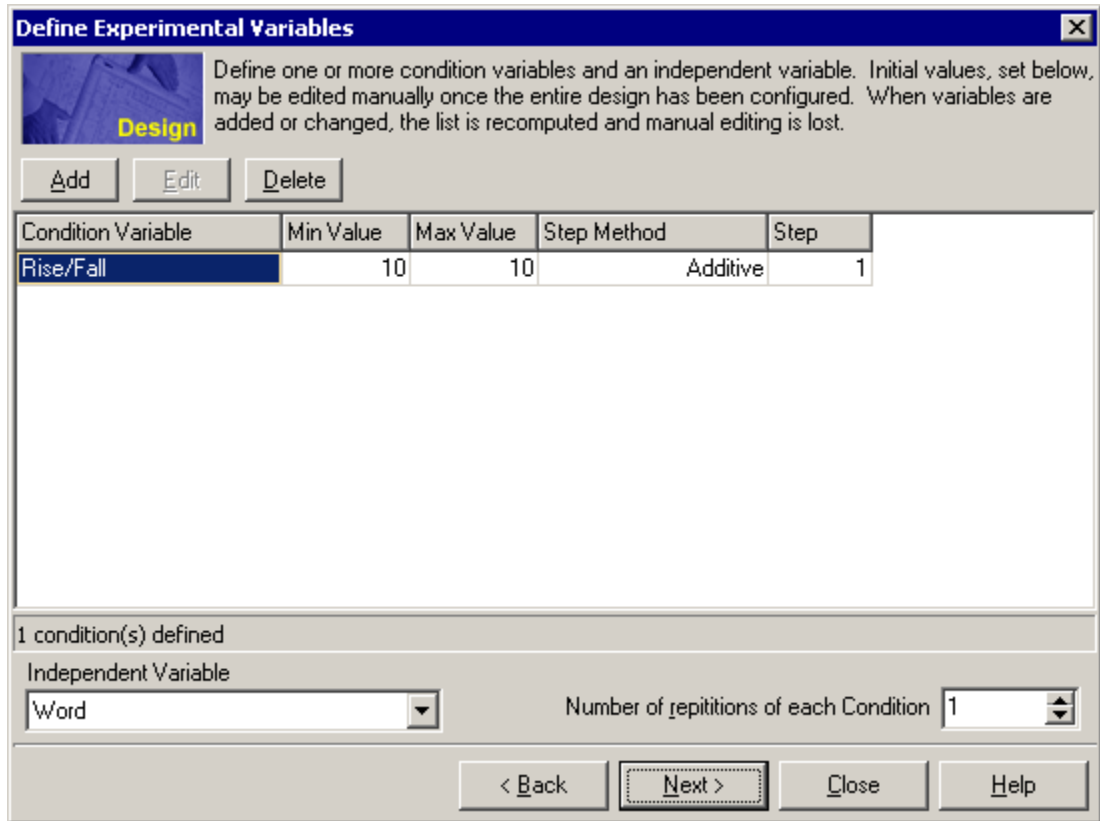
2000 ms Initial Delay used as Inter-Trial Interval, 1000 ms Inter-Stimulus Interval.

### Subject Response Timing

10000 ms response window with feedback after the response. Response time begins at end of 1st interval.

### Define Experimental Variables

Independent variable: Word. In this simple experiment, no condition variables were necessary, so a "dummy" condition variable was created by choosing stimulus rise/fall window as a condition variable ranging from a minimum of 10 ms to a maximum of 10ms, leading to a single condition.



### Independent Variable Configuration

Simple variable generation repeats independent variable value 10 times with the outcome displaying the percent correct.

## Example 10 - Vowel Identification

### File Name

Ex10\_S3\_Ape\_VID\_Buff\_CSIM\_KVM.sfx

### Description

Single interval vowel identification experiment. Each of the seven stimuli is presented a user-specified 10 times in random order, generating a confusion matrix. The listener hears a single vowel sound and then must judge from seven possible response alternatives which vowel was heard.

Similar to Example 9, vowel stimuli are specified as the User Parameter "vowel" defined in the stimulus generation plugin. This parameter defines a list of stimuli that are the basis for the independent variable specified on the Define Experimental Variables wizard.

**RPvdsEx Circuit:** APE.rco

**Stimulus Scripts:** Ex10stim.sc

**Calibration file:** DE\_RE\_AD\_ER2\_test1.bcf

**Subject Interface Scripts:** VID7stim\_KVM.sc

**Subject Interface Configuration:** VID7stim\_KVM.kvm

**Stimulus Generation Configuration**

Stimulus generation is based on APOS commands and the APE (Array Processor Emulator). The stimulus generation scripts used in this design were written in VBScript using the SykofizX scripting interface. Necessary parameters to define: Rise/Fall Time. Interesting features of this design include the User Parameter "Vowel Stimulus". This parameter defines a list of stimuli (vowels) that are the basis for the independent variable specified on the Define Experimental Variables wizard.

**Presentation Paradigm**

A closed set identification matrix paradigm with a single interval.

**Subject Interface Configuration**

Subject response via a keyboard-video-mouse (KVM) interface designed to mimic a four-button response box with labels. The subject interface plugin uses a configuration file created using the KVM configuration utility available whenever the KVM.KeyboardVideoMouse plugin is selected. Properties of the KVM display elements are set in the object inspector within the configuration utility. Actions associated with these elements are defined in the VID7stim\_KVM.sc script library.

**Stimulus Presentation Timing**

2000 ms Initial Delay used as Inter-Trial Interval, 1000 ms Inter-Stimulus Interval.

**Subject Response Timing**

10000 ms response window with feedback after the response. Response time begins at end of 1st interval.

**Define Experimental Variables**

Independent variable: Vowel Stimulus. Experimental Variable: Overall Level.

**Independent Variable Configuration**

Simple variable generation repeats independent variable value 10 times with the outcome displaying the percent correct.

## Example 11 - Classification Experiment - Noun or Verb Recognition Task

**File Name**

Ex11\_S3\_Ape\_WRT\_Buff\_CSC\_KVM.sfx

**Description**

Single interval closed-set classification experiment. Each of the eleven stimulus tokens is presented a user-specified 10 times in random order. The listener must classify the sounds in one of two categories: noun or verb.

**Note:** This fictitious experiment uses stimuli that may be classified in several ways and in that respect does not model a realistic experiment.

Similar to Example 9 and Example 10, stimulus targets are specified as the User Parameter "vowel" defined in the stimulus generation plugin. This parameter defines a list of stimuli that are the basis for the independent variable specified on the Define Experimental Variables wizard.

**RPvdsEx Circuit:** APE.rco

**Stimulus Scripts:** Ex11stim.sc

**Calibration file:** DE\_RE\_AD\_ER2\_test1.bcf

**Subject Interface Scripts:** TwoButtonClass.sc

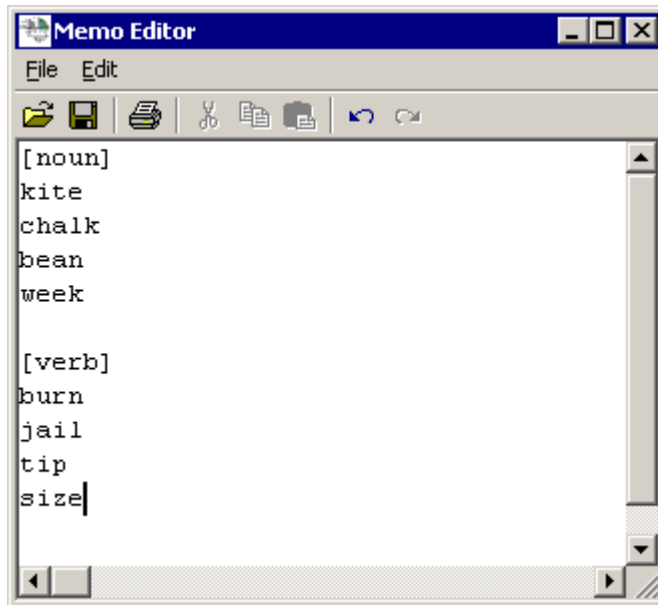
**Subject Interface Configuration: TwoButtonClass.kvm**

**Stimulus Generation Configuration Plugin**

Stimulus generation is based on APOS commands and the APE (Array Processor Emulator). The stimulus generation scripts used in this design were written in VBScript using the SykofizX scripting interface. Necessary parameters to define: Rise/Fall Time. Interesting features of this design include the User Parameter "Vowel Stimulus". This parameter defines a list of stimuli (vowels) that are the basis for the independent variable specified on the Define Experimental Variables wizard.

**Presentation Paradigm**

The target and associated responses (noun,verb) are defined in the Closed\_Set\_Classification presentation plugin in a plugin-specific memo as shown below.



**Subject Interface Configuration**

Subject response via a keyboard-video-mouse (KVM) interface designed to mimic a four-button response box with labels. The subject interface plugin uses a configuration file created using the KVM configuration utility available whenever the KVM.KeyboardVideoMouse plugin is selected. Properties of the KVM display elements are set in the object inspector within the configuration utility. Actions associated with these elements are defined in the TwoButtonClass.sc script library.

**Stimulus Presentation Timing**

1000 ms Initial Delay used as Inter-Trial Interval, 1000 ms Inter-Stimulus Interval.

**Subject Response Timing**

10000 ms response window with feedback after the response. Response time begins at end of 1st interval.

**Define Experimental Variables**

Independent variable: Word. Experimental Variable: Overall Level.

**Independent Variable Configuration**

Simple variable generation repeats independent variable value 10 times with the outcome displaying the percent correct.

## Example 12 - Tone in Noise Detection

### File Name

Ex12\_S3\_Ape\_Tone\_Noise\_Comp\_FC\_UD\_KVM.sfx

### Description

Detection of a tonal signal masked by broadband noise using a two-interval, forced-choice adaptive tracking paradigm, and a keyboard-video-mouse (KVM) subject interface designed to mimic a two-button response box.

The listener will hear two sounds, a noise in isolation and a noise presented simultaneously with a tonal signal. The noise and tone-plus-noise intervals will be presented in random order and separated by a silent period. The listener must indicate, by button press, which interval included the tone.

**RPvdsEx Circuit:** APE.rco

**Stimulus Scripts:** Ex12stim.sc

**Calibration file:** DE\_RE\_AD\_ER2\_test1.bcf

**Subject Interface Scripts:** TwoButton.sc

**Subject Interface Configuration:** TwoButton.kvm

### Stimulus Generation Configuration

Stimulus generation is based on APOS commands and the APE (Array Processor Emulator). The stimulus generation scripts used in this design were written in VBScript using the SykofizX scripting interface. Necessary parameters to define: Duration and Rise/Fall Time.

### Presentation Paradigm

A forced choice presentation with two intervals (signal and standard presented).

### Subject Interface Configuration

Subject response via a keyboard-video-mouse (KVM) interface designed to mimic a two-button response box with LEDs. The subject interface plugin uses a configuration file created using the KVM configuration utility available whenever the KVM.KeyboardVideoMouse plugin is selected. Properties of the KVM display elements are set in the object inspector within the configuration utility. Actions associated with these elements are defined in the TwoButtonKVM.sc script library.

### Stimulus Presentation Timing

1000 ms Initial Delay used as Inter-Trial Interval, 1000 ms Inter-Stimulus Interval.

### Subject Response Timing

10000 ms response window with feedback after the response. Response time begins at end of 2nd interval.

### Define Experimental Variables

Independent variable: Signal Level. Experimental Variables: Frequency and Noise Level.

### Independent Variable Configuration

Adaptive Tracking Up/Down with 2-Down and 1-Up adaptive rule. Fixed number of trials set to 20.

## Example 13 - Just Noticeable Difference Test

### File Name

Ex13\_S3\_APE\_JNDAMF\_Comp\_SJ\_MOA\_KVM.sfx

### Description

Just noticeable difference (JND) in modulation frequency using broadband carrier. A two-interval discrete method of adjustment tracking paradigm includes a reminder interval followed by the signal interval.

The listener will hear two sound intervals. In the first presentation interval, the noise is modulated with the standard modulation frequency. In the second interval, the noise is modulated with a frequency that is higher than the standard ( $f_m + \Delta f_m$ ). A keyboard-video-mouse (KVM) subject interface displays interval markers as well as "Decrease," "Increase," and "Done" buttons allowing the subject to increase or decrease the independent variable (modulation frequency) and to select "Done" when they can just barely detect a difference in modulation frequency.

**RPvdsEx Circuit:** APE.rco

**Stimulus Scripts:** Ex13stim.sc

**Calibration file:** DE\_RE\_AD\_ER2\_test1.bcf

**Subject Interface Scripts:** YesNoKVM2I.sc

**Subject Interface Configuration:** YesNoKVM2I.kvm

### Stimulus Generation Configuration

Stimulus generation is based on APOS commands and the APE (Array Processor Emulator). The stimulus generation scripts used in this design were written in VBScript using the SykofizX scripting interface. Necessary parameters to define: Rise/Fall time, Duration, and Overall Level.

### Presentation Paradigm

A subject judgment paradigm with two intervals (signal and standard presented). Listener must identify if a sound was modulated.

### Subject Interface Configuration

Subject response via a keyboard-video-mouse (KVM) interface designed to mimic a two-button response box. The subject interface plugin uses a configuration file created using the KVM configuration utility available whenever the KVM.KeyboardVideoMouse plugin is selected. Properties of the KVM display elements are set in the object inspector within the configuration utility. Actions associated with these elements are defined in the YesNoKVM2I.sc script library.

### Stimulus Presentation Timing

1000 ms Initial Delay used as Inter-Trial Interval, 500 ms Inter-Stimulus Interval.

### Subject Response Timing

10000 ms response window. Response time begins at end of 1st interval.

### Define Experimental Variables

Independent variable: Delta F. Condition variables: modulation frequency and modulation depth.

### Independent Variable Configuration

Adaptive Tracking discrete adjustment. Fixed number of trials set to 1000.



## Example 14 - Loudness Matching

### File Name

Ex14\_S3\_APE\_MatchLoud\_Comp\_SJ\_MOA\_KVM.sfx

### Description

Match the loudness of a sinusoidal signal (e.g., 250 Hz) to the loudness of a sinusoidal standard (e.g., 1000 Hz) as a function of sinusoidal standard level. This maps out equal loudness contours. A two-interval discrete method of adjustment tracking paradigm includes a reminder interval followed by the signal interval. A keyboard-video-mouse (KVM) subject interface displays interval markers as well as "Decrease," "Increase," and "Done" buttons allowing the subject to increase or decrease the independent variable (modulation frequency) and to select "Done" when they judge the two stimuli to be equal in loudness.

**RPvdsEx Circuit:** APE.rco

**Stimulus Scripts:** Ex14stim.sc

**Calibration file:** DE\_RE\_AD\_ER2\_test1.bcf

**Subject Interface Scripts:** YesNoKVM2I.sc

**Subject Interface Configuration:** YesNoKVM2I.kvm

### Stimulus Generation Configuration

Stimulus generation is based on APOS commands and the APE (Array Processor Emulator). The stimulus generation scripts used in this design were written in VBScript using the SykofizX scripting interface. Necessary parameters to define: Rise/Fall time and Duration.

### Presentation Paradigm

A subject judgment paradigm with one interval and a reminder (signal and standard presented). Listener must compare loudness of the signal presentation to the standard presentation.

### Subject Interface Configuration

Subject response via a keyboard-video-mouse (KVM) interface designed to mimic a two-button response box. The subject interface plugin uses a configuration file created using the KVM configuration utility available whenever the KVM.KeyboardVideoMouse plugin is selected. Properties of the KVM display elements are set in the object inspector within the configuration utility. Actions associated with these elements are defined in the YesNoKVM2I.sc script library.

### Stimulus Presentation Timing

1000 ms Initial Delay used as Inter-Trial Interval, 500 ms Inter-Stimulus Interval.

### Subject Response Timing

10000 ms response window. Response time begins at end of pre-interval reminder.

### Define Experimental Variables

Independent variable: Signal level. Condition variables: standard frequency, signal frequency, and standard level.

### Independent Variable Configuration

Adaptive Tracking discrete adjustment. Fixed number of trials set to 1000.

## Example 15 - Magnitude Estimation

### File Name

Ex15\_S3\_APE\_MagEstLoud\_Comp\_OSJ\_VG\_KVM.sfx

### Description

Magnitude Estimation of the loudness of sinusoidal signals. A single stimulus interval is presented and the listener must type an estimate of the loudness of that sound. The estimate is restricted to the range of 0 to 100 by the KVM configuration label and the max and min set in the Independent Variable Configuration. A keyboard-video-mouse (KVM) subject interface displays an interval marker as well an edit box and an "Enter" button.

**RPvdsEx Circuit:** APE.rco

**Stimulus Scripts:** Ex15stim.sc

**Calibration file:** DE\_RE\_AD\_ER2\_test1.bcf

**Subject Interface Scripts:** OpenResponse\_MagEst I.sc

**Subject Interface Configuration:** OpenResponse\_MagEst I.kvm

### Stimulus Generation Configuration

Stimulus generation is based on APOS commands and the APE (Array Processor Emulator). The stimulus generation scripts used in this design were written in VBScript using the SykofizX scripting interface. Necessary parameters to define: Rise/Fall time and Duration.

### Presentation Paradigm

An open subject judgment paradigm with one interval. Listener must estimate loudness of the signal presented.

### Subject Interface Configuration

Subject response via a keyboard-video-mouse (KVM) interface designed to mimic a two-button response box. The subject interface plugin uses a configuration file created using the KVM configuration utility available whenever the KVM.KeyboardVideoMouse plugin is selected. Properties of the KVM display elements are set in the object inspector within the configuration utility. Actions associated with these elements are defined in the OpenResponse\_MagEst.sc script library.

### Stimulus Presentation Timing

1000 ms Initial Delay used as Inter-Trial Interval, 500 ms Inter-Stimulus Interval.

### Subject Response Timing

10000 ms response window. Response time begins at end of 1st interval.

### Define Experimental Variables

Independent variable: Stimulus level. Condition variable: frequency.

### Independent Variable Configuration

Simple variable generation with each level presented three times.

## Example 16 - Amplitude Modulation Detection Using a Yes-No Paradigm and a Secondary Open Response

### File Name

Ex16\_S3\_APE\_AMD\_Comp\_YN\_OR\_ML\_KVM.sfx

### Description

Sinusoidal amplitude modulation detection using a broadband noise carrier, a single interval presentation and yes-no response paradigm with a secondary open response. The independent variable (modulation depth) varies according to the method of limits.

The subject interface includes a keyboard-video-mouse (KVM) interface designed to mimic a two-button response box with labels YES and NO and an open response box with a "Done" button used to end the response interval. The catch trials option is selected, specifying that the standard (unmodulated) stimulus be presented on a user-specified 25% of the trials.

**RPvdsEx Circuit:** APE.rco

**Stimulus Scripts:** Ex01stim.sc

**Calibration file:** DE\_RE\_AD\_ER2\_test1.bcf

**Subject Interface Scripts:** YesNoKVMopen.sc

**Subject Interface Configuration:** YesNoKVMopen.kvm

### Stimulus Generation Configuration

Stimulus generation is based on APOS commands and the APE (Array Processor Emulator). The stimulus generation scripts used in this design were written in VBScript using the SykofizX scripting interface. Necessary parameters to define: Rise/Fall time and Overall Level.

### Presentation Paradigm

A yes-no response paradigm with one interval. Listener must determine if the signal presented was modulated. The catch trials option is selected, specifying that the standard (unmodulated) stimulus be presented on a user-specified 25% of the trials.

### Subject Interface Configuration

Subject response via a keyboard-video-mouse (KVM) interface designed to mimic a three-button response box with labels Yes, No, and Done. The subject interface plugin uses a configuration file created using the KVM configuration utility available whenever the KVM.KeyboardVideoMouse plugin is selected. Properties of the KVM display elements are set in the object inspector within the configuration utility. Actions associated with these elements are defined in the YesNoKVMopen.sc script library.

### Stimulus Presentation Timing

1000 ms Initial Delay used as Inter-Trial Interval, 1000 ms Inter-Stimulus Interval.

### Subject Response Timing

10000 ms response window. Response time begins at end of 1st interval.

### Define Experimental Variables

Independent variable: Modulation Depth. Condition variables: Modulation Frequency and Duration.

### Independent Variable Configuration

Adaptive Tracking Up/Down with 2-Down and 1-Up adaptive rule. Fixed number of trials set to 20.

## System II

### Example 1 - Amplitude Modulation Detection Using a Forced Choice Paradigm and Keyboard-Video-Mouse

#### File Name

Ex01\_S2\_AP2\_AMD\_comp\_FC\_UD\_KVM.sfx

#### Description

Sinusoidal amplitude modulation detection using a broadband noise carrier, a two-interval, forced-choice adaptive tracking paradigm, and a keyboard-video-mouse (KVM) subject interface designed to mimic a two-button response box. Stimulus generation is based on APOS commands.

Note that the stimulus generation scripts used in this design were written in VBScript using the SykofizX scripting interface and have been exported to the script library file Ex01stim.sc so that they may be imported into other design files.

These scripts work with the System3.rcofile plugin and the SystemII.scriptedstimulus plugin without modification. The subject interface plugin uses a configuration file created using the KVM configuration utility available whenever the KVM.KeyboardVideoMouse plugin is selected. Properties of the KVM display elements are set in the object inspector within the configuration utility. Actions associated with these elements are defined in the TwoButton.sc script library which has been exported for easy use by other designs.

#### Stimulus Generation Configuration

Plugin: SystemII.scriptedstimulus, Parameters: Custom Parameters, Scripts Ex01stim.sc, Calibration file DE\_RE\_AD\_ER2\_test1.bcf

#### Presentation Paradigm

Plugin: PresentationParadigm.ForcedChoice

#### Subject Interface Configuration

Plugin: KVM.KeyboardVideoMouse, Configuration file TwoButtonKvm.kvm, Scripts TwoButton.sc

#### Stimulus Presentation Timing

Details: Initial Delay used as Inter-Trial Interval, 500 ms Inter-Stimulus Interval

#### Subject Response Timing

Details: 10000 ms response window, feedback

#### Define Experimental Variables

Details: Independent variable: modulation depth, Condition variables: modulation frequency, stimulus duration

#### Independent Variable Configuration

Plugin: AdaptiveTracking.UpDown: 2-Down, 1-Up adaptive rule with fixed number of trials

#### Practice Trial Configuration

Details: Automatic practice trials not selected. **Note:** Practice and sample trials may be executed manually at run-time.

#### Run-time Data Storage

Details: Subject name and booth identifier

## Example 2 - Amplitude Modulation Detection Using a Forced Choice Paradigm and Lowpass Filtering

### File Name

Ex02\_S2\_AP2\_AMD\_Comp\_LPF\_FC\_UD\_KVM.sfx

### Description

Sinusoidal amplitude modulation detection using a broadband noise carrier, a two-interval, forced-choice adaptive tracking paradigm, and a keyboard-video-mouse (KVM) subject interface designed to mimic a two-button response box. Stimulus generation is based on APOS commands. Note that the stimulus generation scripts used in this design were written in VBScript using the SykofizX scripting interface and have been exported to the script library file Ex02stim.sc so that they may be imported into other design files.

These scripts work with the System3.rcofile plugin and the SystemII.scriptedstimulus plugin without modification. The subject interface plugin uses a configuration file created using the KVM configuration utility available whenever the KVM.KeyboardVideoMouse plugin is selected. Properties of the KVM display elements are set in the object inspector within the configuration utility. Actions associated with these elements are defined in the TwoButton.sc script library which has been exported for easy use by other designs.

Example 2 differs from Example 1 in terms of the stimulus generation and experimental variables. In Example 2, the broadband noise is read from a data file and the modulated broadband noise is filtered by one of several lowpass filters specified by filter coefficients read from data files. Details regarding filtering using the FIR function may be found in the APOS Reference available at the TDT Website. The four lowpass filter parameters are used as condition variables in the experimental variables wizard.

### Stimulus Generation Configuration

Plugin: SystemII.scriptedstimulus, Parameters: Custom Parameters, Scripts Ex02stim.sc, Calibration file DE\_RE\_AD\_ER2\_test1.bcf

### Presentation Paradigm

Plugin: PresentationParadigm.ForcedChoice

### Subject Interface Configuration

Plugin: KVM.KeyboardVideoMouse, Configuration file TwoButtonKvm.kvm, Scripts TwoButton.sc

### Stimulus Presentation Timing

Details: Initial Delay used as Inter-Trial Interval, 500 ms Inter-Stimulus Interval

### Subject Response Timing

Details: 10000 ms response window, feedback

### Define Experimental Variables

Details: Independent variable: modulation depth, Condition variables: modulation frequency, stimulus duration

### Independent Variable Configuration

Plugin: AdaptiveTracking.UpDown: 2-Down, 1-Up adaptive rule with fixed number of trials

### Practice Trial Configuration

Details: Automatic practice trials not selected. **Note:** Practice and sample trials may be executed manually at run-time.

### Run-time Data Storage

Details: Subject name and booth identifier

## Example 3 - Amplitude Modulation Detection Using a Forced Choice Paradigm and Response Box

### File Name

Ex03\_S2\_AP2\_AMD\_comp\_FC\_UD\_RBox.sfx

### Description

Sinusoidal amplitude modulation detection using a broadband noise carrier, a two-interval, forced-choice adaptive tracking paradigm, and a response box subject interface. Stimulus generation is based on APOS commands. Note that the stimulus generation scripts used in this design were written in VBScript using the SykofizX scripting interface and the script library was imported from the file Ex01stim.sc.

These scripts work with the System3.rcofile plugin and the SystemII.scripstedstimulus plugin without modification. The subject interface is specified by the SystemII.ScripstedSubjectInterface plugin in conjunction with the scripts saved in the file S2\_SI\_PI2\_ButtonBox.sc for use in other designs. These scripts specify the RP2 digital I/O logic necessary to control a standard two-button response box with LED interval markers (or the first two buttons on a response box with more than two sets of buttons and LEDs).

Example 3 is the same as Example 1 except that Example 3 uses the SystemII.ScripstedSubjectInterface plugin rather than the KVM.KeyboardVideoMouse plugin.

### Stimulus Generation Configuration

Plugin: SystemII.scripstedstimulus, Parameters: Custom Parameters, Scripts Ex01stim.sc, Calibration file DE\_RE\_AD\_ER2\_test1.bcf

### Presentation Paradigm

Plugin: PresentationParadigm.ForcedChoice

### Subject Interface Configuration

Plugin: SystemII.ScripstedSubjectInterface, Scripts S2\_SI\_PI2\_ButtonBox.sc

### Stimulus Presentation Timing

Details: Initial Delay used as Inter-Trial Interval, 500 ms Inter-Stimulus Interval

### Subject Response Timing

Details: 10000 ms response window, feedback

### Define Experimental Variables

Details: Independent variable: modulation depth, Condition variables: modulation frequency, stimulus duration

### Independent Variable Configuration

Plugin: AdaptiveTracking.UpDown: 2-Down, 1-Up adaptive rule with fixed number of trials

### Practice Trial Configuration

Details: Automatic practice trials not selected. **Note:** Practice and sample trials may be executed manually at run-time.

### Run-time Data Storage

Details: Subject name and booth identifier

## Example 4 - Amplitude Modulation Detection Using a Same-Different Paradigm and Response Box

### File Name

Ex04\_S2\_AP2\_AMD\_comp\_SD\_VG\_RBox.sfx

### Description

Sinusoidal amplitude modulation detection using a broadband noise carrier, a same-different response paradigm, the method of constant stimuli, and a two-button response box. Stimulus generation is based on APOS commands. Note that the stimulus generation scripts used in this design were written in VBScript using the SykofizX scripting interface and the script library was imported from the file Ex01stim.sc. These scripts work with the System3.rcofile plugin and the SystemII.scripstedstimulus plugin without modification. The subject interface is specified by the System2.ScripstedSubjectInterface plugin in conjunction with the scripts saved in the file S2\_SI\_PI2\_ButtonBox.sc for use in other designs. These scripts specify the PI2 digital I/O logic necessary to control a standard two-button response box with LED interval markers (or the first two buttons on a response box with more than two sets of buttons and LEDs).

Example 4 is the same as Example 3 except that Example 4 is based on the Presentation\_Paradigm.Same\_Different and the SimpleVarGen.VarGen independent variable configuration plugin. The VarGen plugin presents fixed set of independent variable values within a specified range to be presented n times, resulting in a percent correct score for each independent variable value.

### Stimulus Generation Configuration

Plugin: System2.ScripstedStimulus, Parameters: Custom Parameters, Scripts Ex01stim.sc, Calibration file DE\_RE\_AD\_ER2\_test1.bcf

### Presentation Paradigm

Plugin: Presentation\_Paradigm.Same\_Different

### Subject Interface Configuration

Plugin: System3.ScripstedSubjectInterface, Scripts S2\_SI\_PI2\_ButtonBox.sc

### Stimulus Presentation Timing

Details: Initial Delay used as Inter-Trial Interval, 500 ms Inter-Stimulus Interval

### Subject Response Timing

Details: 10000 ms response window, feedback

### Define Experimental Variables

Details: Independent variable: modulation depth, Condition variables: modulation frequency, stimulus duration

### Independent Variable Configuration

Plugin: SimpleVarGen.VarGen: repeats independent variable value n times resulting in percent correct.

### Practice Trial Configuration

Details: Automatic practice trials not selected. **Note:** Practice and sample trials may be executed manually at run-time.

### Run-time Data Storage

Details: Subject name and booth identifier

## Example 6 - Amplitude Modulation Detection Using a Same-Different Paradigm and Keyboard-Video-Mouse

### File Name

Ex06\_S3\_Ape\_AMD\_comp\_SD\_VG\_KVM.sfx

### Description

Sinusoidal amplitude modulation detection using a broadband noise carrier, a same-different response paradigm, the method of constant stimuli, and a keyboard-video-mouse (KVM) subject interface designed to mimic a two-button response box with labels SAME and DIFFERENT. Stimulus generation is based on APOS commands and the APE (Array Processor Emulator). Note that the stimulus generation scripts used in this design were written in VBScript using the SykofizX scripting interface and the script library was imported from the file Ex01stim.sc.

These scripts work with the System3.rcofile plugin and the SystemII.scripstedstimulus plugin without modification. The KVM.KeyboardVideoMouse subject interface plugin uses a configuration file created using the KVM configuration utility. Properties of the KVM display elements are set in the object inspector within the configuration utility. Actions associated with these elements are defined in the TwoButton.sc script library.

Example 6 is similar to Example 4, using the Presentation\_Paradigm.Same\_Different and the SimpleVarGen.VarGen independent variable configuration plugin. The VarGen plugin presents fixed set of independent variable values within a specified range to be presented n times, resulting in a percent correct score for each independent variable value. Example 6 differs from Example 4 in that the two button KVM subject interface is used rather than a response box and digital I/O.

### Stimulus Generation Configuration

Plugin: System3.rcofile, Parameters: APE, Custom Parameters, Scripts Ex01stim.sc, Calibration file DE\_RE\_AD\_ER2\_test1.bcf

### Presentation Paradigm

Plugin: Presentation\_Paradigm.Same\_Different

### Subject Interface Configuration

Plugin: KVM.KeyboardVideoMouse, Configuration file: TwoButtonKVM.kvm, Scripts: TwoButtonKVMScripts.sc

### Stimulus Presentation Timing

Details: Initial Delay used as Inter-Trial Interval, 500 ms Inter-Stimulus Interval

### Subject Response Timing

Details: 10000 ms response window, feedback

### Define Experimental Variables

Details: Independent variable: modulation depth, Condition variables: modulation frequency, stimulus duration

### Independent Variable Configuration

Plugin: SimpleVarGen.VarGen: repeats independent variable value n times resulting in percent correct.

### Practice Trial Configuration

Details: Automatic practice trials not selected. **Note:** Practice and sample trials may be executed manually at run-time.

### Run-time Data Storage

Details: Subject name and booth identifier



## Example 7 - Amplitude Modulation Detection Using a Forced Choice Paradigm and a Reminder Interval

### File Name

Ex07\_S2\_AP2\_AMD\_comp\_FCR\_UD\_KVM.sfx

### Description

Sinusoidal amplitude modulation detection using a broadband noise carrier, a cued two-interval, forced-choice adaptive tracking paradigm, and a keyboard-video-mouse (KVM) subject interface designed to mimic a two-button response box. Stimulus generation is based on APOS commands. Note that the stimulus generation scripts used in this design were written in VBScript using the SykofizX scripting interface and have been exported to the script library file Ex01stim.sc so that they may be imported into other design files.

These scripts work with the System3.rcofile plugin and the SystemII.scripstedstimulus plugin without modification. The subject interface plugin uses a configuration file created using the KVM configuration utility available whenever the KVM.KeyboardVideoMouse plugin is selected.

Properties of the KVM display elements are set in the object inspector within the configuration utility. Actions associated with these elements are defined in the ThreeButton.sc script library which has been exported for easy use by other designs.

Example 7 differs from Example 1 with the inclusion of a reminder interval beginning each trial. This reminder or "cue" interval is always associated with the standard stimulus class as specified in the stimulus generation plugin.

### Stimulus Generation Configuration

Plugin: SystemII.ScripstedStimulus, Parameters: Custom Parameters, Scripts Ex01stim.sc, Calibration file DE\_RE\_AD\_ER2\_test1.bcf

### Presentation Paradigm

Plugin: PresentationParadigm.ForcedChoice

### Subject Interface Configuration

Plugin: KVM.KeyboardVideoMouse, Configuration file ThreeButtonKvm.kvm, Scripts ThreeButton.sc

### Stimulus Presentation Timing

Details: Initial Delay used as Inter-Trial Interval, 500 ms Inter-Stimulus Interval

### Subject Response Timing

Details: 10000 ms response window, feedback

### Define Experimental Variables

Details: Independent variable: modulation depth, Condition variables: modulation frequency, stimulus duration

### Independent Variable Configuration

Plugin: AdaptiveTracking.UpDown: 2-Down, 1-Up adaptive rule with fixed number of trials

### Practice Trial Configuration

Details: Automatic practice trials not selected. **Note:** Practice and sample trials may be executed manually at run-time.

### Run-time Data Storage

Details: Subject name and booth identifier

## Example 8 - Amplitude Modulation Detection Using a Yes-No Paradigm and Keyboard-Video-Mouse

### File Name

Ex08\_S2\_AP2\_AMD\_comp\_YN\_ML\_KVM.sfx

### Description

Sinusoidal amplitude modulation detection using a broadband noise carrier, a single interval presentation and yes-no response paradigm, the method of limits, and a keyboard-video-mouse (KVM) subject interface designed to mimic a two-button response box with labels YES and NO. Stimulus generation is based on APOS commands. Note that the stimulus generation scripts used in this design were written in VBScript using the SykofizX scripting interface and the script library was imported from the file Ex01stim.sc.

These scripts work with the System3.rcofile plugin and the SystemII.scriptedstimulus plugin without modification. The KVM.KeyboardVideoMouse subject interface plugin uses a configuration file (YesNoKVM.kvm) created using the KVM configuration utility. Properties of the KVM display elements are set in the object inspector within the configuration utility.

Actions associated with these elements are defined in the YesNoKVM.sc script library. The catch trials option is selected, specifying that the standard (unmodulated) stimulus be presented on a user-specified 25% of the trials.

### Stimulus Generation Configuration

Plugin: SystemII.ScriptedStimulus, Parameters: Custom Parameters, Scripts Ex01stim.sc, Calibration file DE\_RE\_AD\_ER2\_test1.bcf

### Presentation Paradigm

Plugin: Presentation\_Paradigm.Yes\_No with catch trials

### Subject Interface Configuration

Plugin: KVM.KeyboardVideoMouse, Configuration file: YesNoKVM.kvm, Scripts: YesNoKVMScripts.sc

### Stimulus Presentation Timing

Details: Initial Delay used as Inter-Trial Interval, 500 ms Inter-Stimulus Interval

### Subject Response Timing

Details: 10000 ms response window, feedback

### Define Experimental Variables

Details: Independent variable: modulation depth, Condition variables: modulation frequency, stimulus duration

### Independent Variable Configuration

Plugin: Adaptive\_Tracking.Method\_of\_Limits.

### Practice Trial Configuration

Details: Automatic practice trials not selected. **Note:** Practice and sample trials may be executed manually at run-time.

### Run-time Data Storage

Details: Subject name and booth identifier

## Example 9 - Word Recognition Test

### File Name

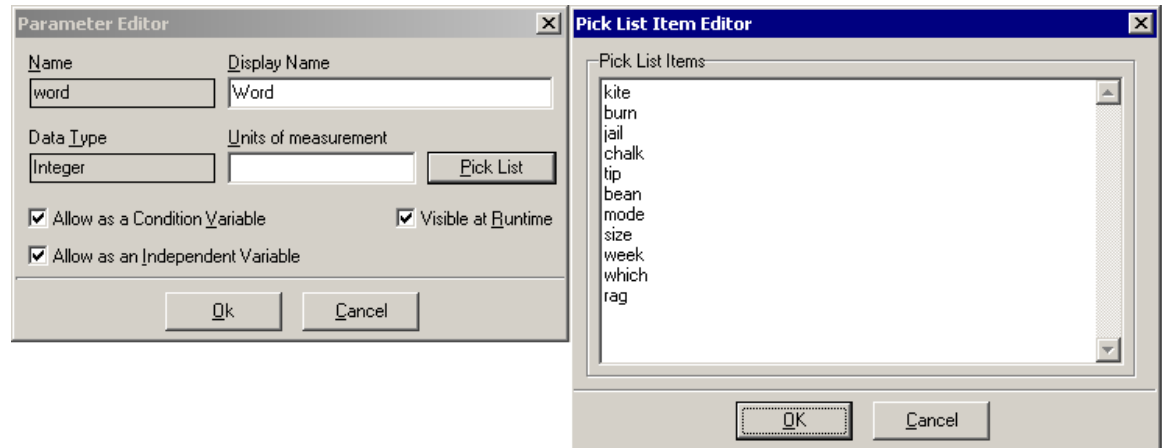
Ex09\_S2\_AP2\_WRT\_Buff\_CSIL\_KVM.sfx

### Description

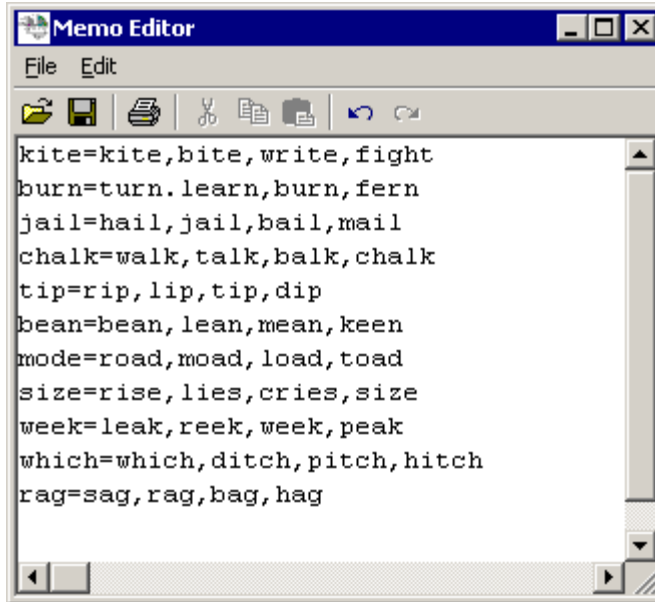
Single-interval closed-set identification word recognition test. Each stimulus token has four possible responses including the target and three rhyming foils. Stimulus generation is based on APOS commands. Note that the stimulus generation scripts used in this design were written in VBScript using the SykofizX scripting interface and have been exported to the script library file Ex09stim.sc so that they may be imported into other design files.

These scripts work with the System3.rcofile plugin and the SystemII.scriptedstimulus plugin without modification. The scripts load stimulus files based on the current independent variable value. The subject interface plugin uses a configuration file created using the KVM configuration utility. Properties of the KVM display elements are set in the object inspector within the configuration utility and are saved in the configuration file wrtkvm1.kvm. Actions associated with these elements are defined in the wrtkvm1.sc script library which has been exported for easy use by other designs.

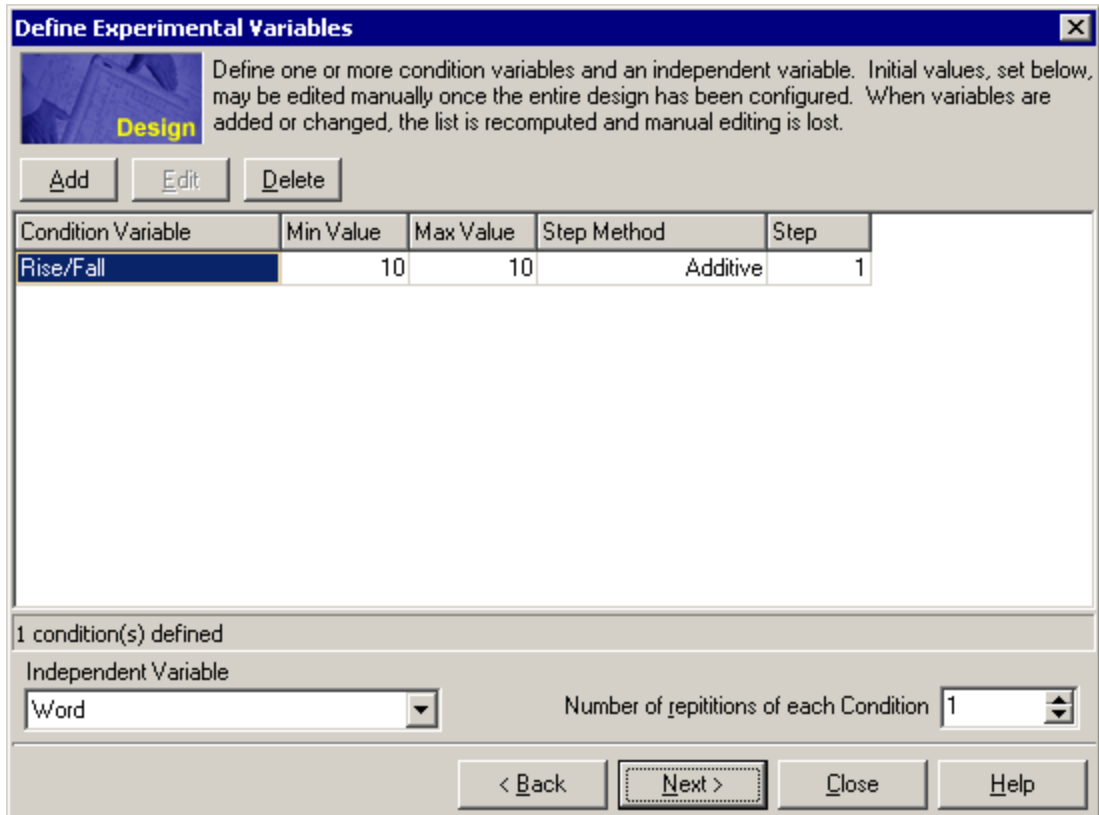
Interesting features of this design include the User Parameter "word" defined in the stimulus generation plugin. This parameter defines a list of stimuli (words) that are the basis for the independent variable specified on the Define Experimental Variables wizard.



The target and associated foils are defined in the Closed\_Set\_Identification\_List presentation plugin in a plugin-specific memo as shown below.



In this simple experiment, no condition variables were necessary, so a "dummy" condition variable was created by choosing stimulus rise/fall window as a condition variable ranging from a minimum of 10 ms to a maximum of 10ms, leading to a single condition (see image below).



### Stimulus Generation Configuration

Plugin: SystemII.ScriptedStimulus, Parameters: Custom Parameters, Scripts Ex09stim.sc, calibration file DE\_RE\_AD\_ER2\_test1.bcf

**Presentation Paradigm**

Plugin: Presentation\_Paradigm.Closed\_Set\_Identification\_List

**Subject Interface Configuration**

Plugin: KVM.KeyboardVideoMouse, Configuration file wrtkvm1.kvm, Scripts wrtkvm1.sc

**Stimulus Presentation Timing**

Details: Initial Delay used as Inter-Trial Interval

**Subject Response Timing**

Details: 10000 ms response window, feedback

**Define Experimental Variables**

Details: Independent variable: word, Condition variables: "dummy" Rise/Fall

**Independent Variable Configuration**

Plugin: SimpleVarGen.VarGen: repeats independent variable value 10 times resulting in percent correct.

**Practice Trial Configuration**

Details: Automatic practice trials not selected. **Note:** Practice and sample trials may be executed manually at run-time.

**Run-time Data Storage**

Details: Subject name and booth identifier

## Example 10 - Vowel Identification

**File Name**

Ex10\_S2\_AP2\_VID\_Buff\_CSIM\_KVM.sfx

**Description**

Single interval vowel identification experiment. Each of the seven stimuli is presented a user-specified 10 times in random order, generating a confusion matrix. Stimulus generation is based on APOS commands. Note that the stimulus generation scripts used in this design were written in VBScript using the SykofizX scripting interface and have been exported to the script library file Ex10stim.sc so that they may be imported into other design files.

These scripts work with the System3.rcofile plugin and the SystemII.scriptedstimulus plugin without modification. The scripts load stimulus files based on the current independent variable value. The subject interface plugin uses a configuration file created using the KVM configuration utility. Properties of the KVM display elements are set in the object inspector within the configuration utility and are saved in the configuration file ihtoiikvm.kvm. Actions associated with these elements are defined in the ihtoiikvm.sc script library which has been exported for easy use by other designs.

Similar to Example 9, vowel stimuli are specified as the User Parameter "vowel" defined in the stimulus generation plugin. This parameter defines a list of stimuli that are the basis for the independent variable specified on the Define Experimental Variables wizard.

**Stimulus Generation Configuration**

Plugin: SystemII.ScriptedStimulus, Parameters: Custom Parameters, Scripts Ex10stim.sc, Calibration file DE\_RE\_AD\_ER2\_test1.bcf

**Presentation Paradigm**

Plugin: Presentation\_Paradigm.Closed\_Set\_Identification\_Matrix

### **Subject Interface Configuration**

Plugin: KVM.KeyboardVideoMouse, Configuration file ihtoiikvm.kvm, Scripts ihtoiikvm.sc

### **Stimulus Presentation Timing**

Details: Initial Delay used as Inter-Trial Interval

### **Subject Response Timing**

Details: 10000 ms response window, feedback

### **Define Experimental Variables**

Details: Independent variable: vowel, Condition variables: overall level

### **Independent Variable Configuration**

Plugin: SimpleVarGen.VarGen: repeats independent variable value 10 times resulting in percent correct.

### **Practice Trial Configuration**

Details: Automatic practice trials not selected. **Note:** Practice and sample trials may be executed manually at run-time.

### **Run-time Data Storage**

Details: Subject name and booth identifier

## **Example 11 - Classification Experiment**

### **File Name**

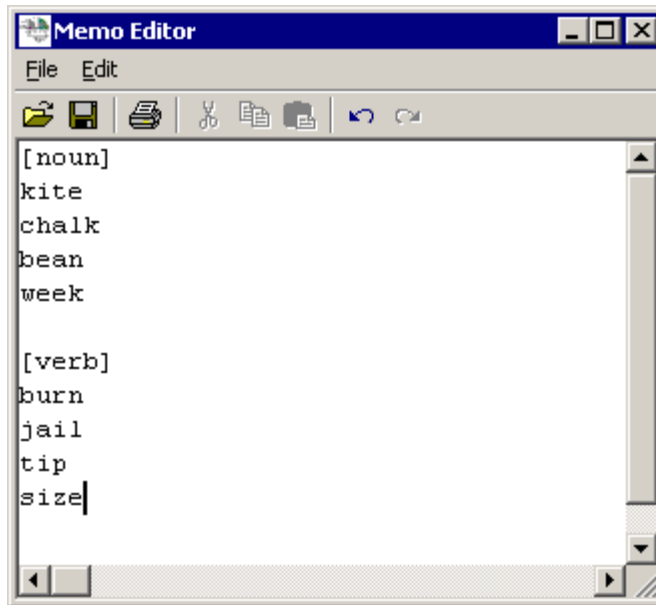
Ex11\_S2\_AP2\_WRT\_Buff\_CSC\_KVM.sfx

### **Description**

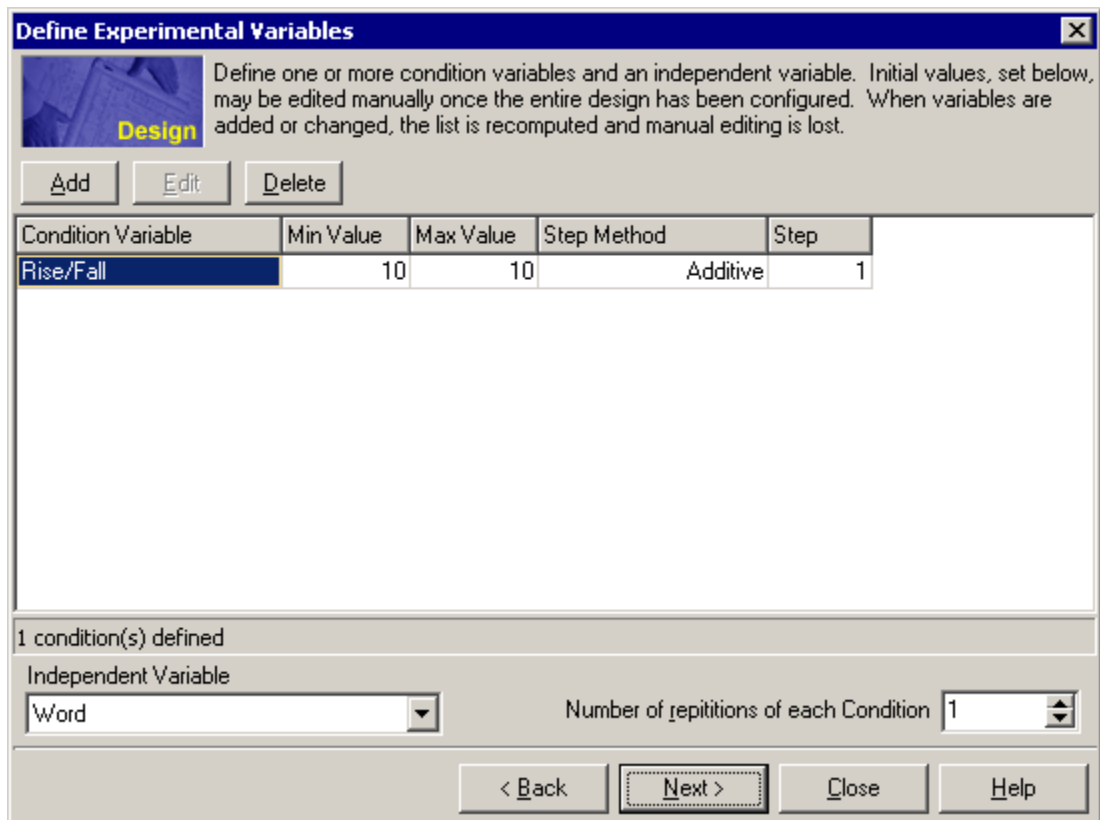
Single interval closed-set classification experiment. Each of the eleven stimulus tokens is presented a user-specified 10 times in random order. The listener must classify the sounds in one of two categories: noun or verb. Stimulus generation is based on APOS commands. Note that the stimulus generation scripts used in this design were written in VBScript using the SykofizX scripting interface and have been exported to the script library file Ex11stim.sc so that they may be imported into other design files. These scripts work with the System3.rcofile plugin and the SystemII.scriptedstimulus plugin without modification. The scripts load stimulus files based on the current independent variable value. The subject interface plugin uses a configuration file created using the KVM configuration utility. Properties of the KVM display elements are set in the object inspector within the configuration utility and are saved in the configuration file TwoButtonClass.kvm. Actions associated with these elements are defined in the TwoButtonClass.sc script library which has been exported for easy use by other designs.

Similar to Examples 9 and 10, stimulus targets are specified as the User Parameter "vowel" defined in the stimulus generation plugin. This parameter defines a list of stimuli that are the basis for the independent variable specified on the Define Experimental Variables wizard.

The target and associated responses (noun,verb) are defined in the Closed\_Set\_Classification presentation plugin in a plugin-specific memo as shown below.



Similar to Example 9, in this simple experiment, no condition variables were necessary, so a "dummy" condition variable was created by choosing stimulus rise/fall window as a condition variable ranging from a minimum of 10 ms to a maximum of 10ms, leading to a single condition (see image below).



### Stimulus Generation Configuration

Plugin: SystemII.ScriptedStimulus Parameters: Custom Parameters, Scripts Ex1Istim.sc, , Calibration file DE\_RE\_AD\_ER2\_test1.bcf

### **Presentation Paradigm**

Plugin: Presentation\_Paradigm.Closed\_Set\_Classification

### **Subject Interface Configuration**

Plugin: KVM.KeyboardVideoMouse, Configuration file TwoButtonClass.kvm, Scripts TwoButtonClass.sc

### **Stimulus Presentation Timing**

Details: Initial Delay used as Inter-Trial Interval

### **Subject Response Timing**

Details: 10000 ms response window, feedback

### **Define Experimental Variables**

Details: Independent variable: word, Condition variables: "dummy" Rise/Fall variable.

### **Independent Variable Configuration**

Plugin: SimpleVarGen.VarGen: repeats independent variable value 10 times resulting in percent correct.

### **Practice Trial Configuration**

Details: Automatic practice trials not selected. **Note:** Practice and sample trials may be executed manually at run-time.

### **Run-time Data Storage**

Details: Subject name and booth identifier

## **Example 12 - Tone in Noise Detection**

### **File Name**

Ex12\_S2\_AP2\_Tone\_Noise\_Comp\_FC\_UD\_KVM.sfx

### **Description**

Detection of a tonal signal masked by broadband noise using a two-interval, forced-choice adaptive tracking paradigm, and a keyboard-video-mouse (KVM) subject interface designed to mimic a two-button response box. Stimulus generation is based on APOS commands. Note that the stimulus generation scripts used in this design were written in VBScript using the SykofizX scripting interface and have been exported to the script library file Ex12stim.sc so that they may be imported into other design files.

These scripts work with the System3.rcofile plugin and the SystemII.scriptedstimulus plugin without modification. The subject interface plugin uses a configuration file created using the KVM configuration utility available whenever the KVM.KeyboardVideoMouse plugin is selected.

Properties of the KVM display elements are set in the object inspector within the configuration utility. The configuration file used is TwoButtonKVM.kvm. Actions associated with these elements are defined in the TwoButtonScripts.sc script library which has been exported for easy use by other designs.

In this design, signal level is the independent variable, and condition variables include signal frequency and masker level.

### **Stimulus Generation Configuration**

Plugin: SystemII.ScriptedStimulus, Parameters: Custom Parameters, Scripts Ex12stim.sc, Calibration file DE\_RE\_AD\_ER2\_test1.bcf



**Presentation Paradigm**

Plugin: Presentation\_Paradigm.Forced\_Choice

**Subject Interface Configuration**

Plugin: KVM.KeyboardVideoMouse, Configuration file: TwoButtonKVM.kvm, Scripts  
TwoButtonScripts.sc

**Stimulus Presentation Timing**

Details: Fixed Inter-Stimulus-Interval

**Subject Response Timing**

Details: 10000 ms response window, feedback

**Define Experimental Variables**

Details: Independent variable: signal level, Condition variables: signal frequency, masker level.

**Independent Variable Configuration**

Plugin: AdaptiveTracking.UpDown: 2-Down, 1-Up adaptive rule with fixed number of trials

**Practice Trial Configuration**

Details: Automatic practice trials not selected. **Note:** Practice and sample trials may be executed manually at run-time.

**Run-time Data Storage**

Details: Subject name and booth identifier