Basic E-Prime Topics
• Overview of E-Prime 2.0.8.90 Professional Suite
• Overview of E-Studio
• Benefits of planning ahead for experiment
• Storyboard task
• Building the Lexical Decision Task
• Run and Review Data

Intermediate E-Prime Topics
• Working with multiple ES2 files
• Converting from Standard to Professional
• From Lexical Decision Task to Priming Task
• Writing E-Basic Script
• E-Basic Script Style Guidelines
• E-DataAid and Analysis

Advanced E-Prime Topics
• Experiment Object Properties
• Advanced Input
• Timing Considerations
• Advantages of updating to Production Release/Service Pack

How To Get Experiment Help
Overview of E-Prime 2.0.8.90 Professional Suite

Core Components:

- **E-Studio**: Graphical design environment
- **E-Run**: Real-time experiment generator
- **E-Merge**: Data merging utility
- **E-DataAid**: Spreadsheet-like application for analyzing data files

E-Prime consists of four major components: E-Studio, E-Run, E-Merge and E-DataAid. These four components make up the core of the E-Prime suite of applications, but they are not the complete list of applications.
Overview of E-Prime 2.0.8.90 Professional Suite

Support Components:

**StartupInfo Editor**
Create read-only startup info options

**PackageFile Editor**
Create re-usable pieces of script

**FactorTable Wizard**
Easily create a cross-section of independent variables

**Sound Tester**
Test sound APIs on your computer (Production Release+)

**Codec Config**
Optimize your computer for sound and movie files

The E-Prime suite also includes CodecConfig, FactorTableWizard, PackageFileEditor, Sound Tester (Production Release+) and StartupInfoEditor. These applications are made to enhance the quality of the stimuli presented to the participant.
Overview of E-Studio
1. **Structure Window**- This provides an overview of the objects in your experiment. This is used to quickly and easily determine the order of events.

2. **Toolbox** - A repository for all of the E-Objects that can possibly be used in an experiment. They are as follows:
   1. **ImageDisplay** – Used to display only images to participants. Makes formatting images fast and simple.
   2. **InLine** – Allows you to type script into here to gain a higher level of control over the experiment.
   3. **Label** – Controls experiment flow. Place a Label object anywhere in a procedure to jump directly to that point.
   4. **List** – Used to create and control Attributes and Procedures. Also useful for logging and controlling independent variables.
   5. **MovieDisplay** – Used to only display videos to participants. Movie types include wmv, mp4 and avi.
   6. **PackageCall**- These objects allow you to import re-usable sections of script quickly. Also useful for E-Prime extensions.
   7. **Procedure** – A timeline of events for an experiment. E-Objects are placed on these.
   8. **Slide** – The most versatile of all of the E-Objects. Primarily used to display multiple stimuli at a time.
   9. **FeedbackDisplay**- A unique Slide object that switches SlideStates depending on the accuracy of a pre-determined E-Object.
   10. **SoundIn** - Used to create a .wav file of participants’ vocal responses.
   11. **SoundOut** - Does not have a visual component, only plays sound to participants. Sound types include wav and mp3.
   12. **TextDisplay**- Used to easily manipulate the presentation of text to a participant.

3. **Properties Window** – Contains a quick overview of the properties of a selected object. Used for fast alterations of an object’s properties.

4. **Output Window** – Displays experiment information including “Debug” statements and compile information.

5. **Attribute Window** – A quick reference of all of the Attributes created in your experiment.
Benefits of Planning Ahead

- Keep track of dependent and independent variables more easily
- Data dictates design, not the other way around
- Makes programming the experiment easier
- Allows for faster recognition of problem areas in the experiment
- A properly designed experiment leads to a clean data file and faster analysis
- Refer to the Data File tab of the Experiment Object properties for proper experiment structure
The basic lexical decision task experiment includes a fixation, stimulus, and feedback. The participant is asked to identify whether the stimulus is a word or non-word. It is hypothesized that participants will take longer and will be less accurate when responding to non-words. We will need to make sure to log RT and accuracy to test this.
Building the Lexical Decision Task

The following slides contain information on specific techniques used to create the basic Lexical Decision task.
Nested lists provide a method to organize a list of items (e.g., words and non-words) and reference the organized list in the List object used to run the Procedure. One benefit to this method is the ability to set the selection method for each of the List objects independently. Also, by using nested lists, it is not necessary to create a list that contains a full crossing of all conditions and exemplars. Often a full crossing of the independent variables results in more exemplars than is feasible to present within an experimental session.

Create two new lists – “WordList” and “NonWordList” by typing their names in the Nested column of TrialList. Add three levels to each list for a total of four each. Create a new attribute on each List called “Stim”. Fill the Stim attribute with the stimulus words and non-words.
List sampling can be affected in several different ways.
- Weight can be used to essentially duplicate levels
- The Reset/Exit tab of the List’s Property Pages can be used to control when the List resets its cycle or when it exits
- The Selection tab sets the Order property of the List (random, sequential, counterbalance, etc.), and if using counterbalance, offset, or permutation order, bases the selection order on one of the StartupInfo values.
Duration/Input timing

Duration vs. Time Limit
Two very important properties on the Duration/Input tab are Duration and Time Limit. These are important for making sure that objects are properly collecting responses, and allowing for more advanced experiment design techniques such as extended input. Duration controls the duration of the object itself, while Time Limit controls only the duration of the input mask on the object. Duration and Time Limit are independent of each other.

Cumulative vs. Event Mode
In Event mode, delays in the onset of an event do not affect the specified duration of the event.
In Cumulative mode, delays in the onset of an event result in an equivalent reduction in the duration of the event, such that the cumulative timing error is minimized. It is generally recommended to use Event mode timing to ensure that each object is presented for as close to its specified duration as possible, and use Cumulative timing to ensure that the length of an entire trial is as close to the specified duration as possible. Custom mode should not be used without specific instructions from PST technical support.
To determine in the .EDAT2 file that experiment timing is consistent and the experiment is working well, log timing information for objects that are displaying but not collecting responses. Do this for the Fixation and FeedbackDisplay objects; the Stimulus object is using Standard logging, which means that it will already be logging response information in addition to timing information.

To enable logging of timing statistics for the Fixation object, open it in the workspace, select the Property Pages, click the Duration/Input tab, and change the Data Logging option from (none) to Time Audit Only. Determine which specific properties are being logged by selecting the Logging tab at the top of the Property Pages window. Note that when Time Audit Only is selected, DurationError, OnsetDelay, and OnsetTime statistics are checked. Compare the Logging tab when Data Logging is set to (none) and when it is set to other options.
Run and review data

Run the experiment several more times to create more data files, making sure to use a different subject and/or session number for each one. Open E-Merge (from the Tools menu in E-Studio) and navigate to the newly created data files. Use the Select Unmerged button to select the files, then click Merge. Follow the prompts to merge the files into a single file.

Exiting early

Ctrl + Alt + Shift: force exit
Ctrl + Shift: graceful exit with InLine script
Ctrl + Alt + Backspace: graceful exit without script
Open E-DataAid (from the Tools menu in E-Studio), find a single or merged data file, and open it. On the far left of the file, notice experiment information such as the file name, as well as independent variables such as the Subject and Session StartupInfo values. As you continue right, note timing and response data logged from the session in a format similar to an Excel file. Use the Arrange Columns and Filter buttons in the top toolbar to filter out trials or temporarily hide columns which are not of immediate concern.

Use the Analyze tool to create an analysis evaluating reaction time in relation to congruency. Press Run to view the results of the analysis. In the newly created table, check whether the hypothesis is true: response time for congruent trials should be shorter than response time for incongruent trials.
One way to quickly create an experiment is to reuse objects that have already been created in past experiments. This is also a method for converting programs from Professional to Standard. If you would like to do this, you will first have to open two instances of E-Studio. You then have two options for moving objects between experiments. You can copy and paste objects by either right clicking on the object and selecting “Copy”, then moving over to the new experiment, finding the procedure that you would like to contain this object, right clicking and selecting “Paste”. There is also the “drag and drop” method of transferring objects between experiments. To do this, simply drag and object from one experiment and drop it onto the other. When you do this, blue arrows will appear where the object will be dropped into the experiment.
Converting from Standard to Professional

There are several versions of the E-Prime software with corresponding experiment formats. E-Prime 2.0 Standard can open/edit E-Prime 1.x or E-Prime 2.0 Standard experiments. E-Prime 2.0 Professional can open/edit E-Prime 1.x, E-Prime 2.0 Standard, and E-Prime 2.0 Professional experiments. E-Prime is not backwards compatible, so any experiment created or edited with a newer version cannot be opened again in an older version. For example, it is not possible to open an E-Prime 2.0 Production Release experiment in E-Prime 2.0.8.90 or in E-Prime 1.2.

For a chart of differences between Standard and Professional, see http://www.pstnet.com/eprimecompare.cfm.
The basic lexical decision task experiment includes a fixation, stimulus, and feedback. The participant is asked to identify whether the stimulus is a word or non-word. For the intermediate version of the lexical decision task, a Prime Slide will occur directly before the stimulus. The prime will contain either a red or green image. For congruent trials, the red signal will prime non-words and the green signal will prime words. For incongruent trials, the opposite will be the case. This will demonstrate the effect of the congruency of the prime on participants’ response time to the stimulus. SlideStates will be used to create a custom feedback slide, demonstrating the use of basic scripting techniques. Experiment-wide logging of response time average and accuracy average will occur using a Summation object. The following slides demonstrate a few techniques used to create this task.
SlideState objects are used to enter and organize the stimuli to be presented at run-time. The collection of SlideStates allows customization for specific conditions independently. A SlideState can present almost any combination of Slide sub-objects available in the top toolbar. This method is used to share object-wide properties while varying what is presented to the participant.

Set the ActiveState of the Slide to change by adding the [FeedbackState] attribute reference to the ActiveState property in the Feedback Slide object’s General tab.

Add the SlideText sub-objects to both Correct and Incorrect Slide states, type the word “Correct!” and “Incorrect!”, and make them either red or green depending on feedback type.

Create an InLine object called “ProcessFeedback” directly between the Stimulus and Feedback Slide objects. You will need to write a small bit of script here to determine which Slide State of the Feedback Slide is displayed.

```vbnet
If Stimulus.ACC = "1" Then
    c.SetAttrib "FeedbackState", "Correct"
Else
    c.SetAttrib "FeedbackState", "Incorrect"
End If
```
Create the List for the Practice Procedure by copying and pasting the TrialList into the new PracBlockProc Procedure. This will create a copy of the List, instead of having to create a new one from scratch. Right-click TrialList in the Structure window and select “Copy”. Double-click the new PracBlockProc Procedure so that it opens in the Workspace. Right-click inside the PracBlockProc window and select “Paste”. A prompt will appear asking if you want to paste all of the child objects (all objects referenced in the TrialList, including TrialProc and those in TrialProc). To reuse TrialProc instead of creating TrialProc1, Prime1, Stimulus1, etc., select “No”. Rename the new TrialList1 to “PracList”.

If the E-Object Paste Conflict ever appears, examine the options and choose the best option for the situation.
The Summation object is an object created in script which is used to collect observations of data and easily retrieve more complicated statistics concerning those observations. To implement the Summation object, like many other intermediate/advanced scripting techniques, it is often helpful to look in E-Basic help for examples.

To find more information about the Summation Object, navigate to E-Basic Help in the top toolbar by going to Help > E-Basic Help. Once E-Basic Help is open, type “Summation Object” in the search field. Click the List Topics button to view results; the Summation Object will appear in the List. The (topic) heading of any object will include a list of all the object’s Properties and Methods as well as any related information or objects.
Writing E-Basic Script Cont.

Create, set, and fill Summations

```
Dim CorrectRT As Summation
Dim TotalACC As Summation
```

```
Set CorrectRT = New Summation
Set TotalACC = New Summation
```

```
If Stimulus.ACC = 1 Then
    CorrectRT.AddObservation Stimulus.RT
End If

TotalACC.AddObservation Stimulus.ACC
```
In order for the Summation to be accessed at the trial level (to add trial-level observations) but be displayed at the session level, it must be declared as a globally accessible object. To create objects or variables accessible at all levels of the experiment, use the User tab of the Script window. The Script window can be opened through the View menu or by typing “Alt + 5”; click the User tab at the bottom of the Script window to edit global script.

Declare two Summation objects (CorrectRT and TotalACC) in User Script. Set them to a new Summation by adding an InLine at the top of SessionProc. This step cannot be completed in User Script, as the information in User Script is processed before objects are declared by E-Prime. Rename the InLine to “SetSummations”.

Use an If...Then statement to add response times to the CorrectRT Summation Object when the trial meets the correct criteria. By only including the AddObservation method in the “If” portion of the statement, we are telling the script to add only correct responses. Use TotalACC.AddObservation outside the If...Then statement to include all accuracy data in the collection.

To exclude practice values in the final summations, reset the summations at the end of PracBlockProc. The Reset method should not be used at the beginning of BlockProc in case another experimental block is added later. Add CorrectRT.Reset and MeanACC.Reset in a new InLine at the end of PracBlockProc.
In order to display the statistics at the end of the experiment, the Summation value variables must be saved as attributes. Create two new attributes called FinalCorrectRT and FinalMeanACC. Convert the decimal created by calculating the mean of the .ACC values into a percentage using the Format function (by searching for Percent in E-Basic Help). Restrict CorrectRT.Mean to two decimal points by using the 0.### custom format.

<table>
<thead>
<tr>
<th>Macro</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>@ACC.MEAN</td>
<td>Mean accuracy for a collection of responses</td>
</tr>
<tr>
<td>@ACC.N</td>
<td>Number of observations in the collection of responses</td>
</tr>
<tr>
<td>@ACC</td>
<td>Accuracy for an individual (i.e., the current number of correct responses)</td>
</tr>
<tr>
<td>@RT.MEAN</td>
<td>Mean reaction time for a collection of responses</td>
</tr>
<tr>
<td>@RT.MIN</td>
<td>Minimum reaction time for a collection of responses</td>
</tr>
<tr>
<td>@RT.MAX</td>
<td>Maximum reaction time for a collection of responses</td>
</tr>
<tr>
<td>@RT.STDDEVVP</td>
<td>Calculates standard deviation based on the number of observations</td>
</tr>
<tr>
<td>@RT.STDDEVVS</td>
<td>Calculates standard deviation based on the number of correct responses</td>
</tr>
<tr>
<td>@RT.N</td>
<td>Number of observations in the collection of correct responses</td>
</tr>
<tr>
<td>@RT</td>
<td>Reaction time for an individual (i.e., correct responses)</td>
</tr>
<tr>
<td>@CorrectRT.MEAN</td>
<td>Mean reaction time for the collection of correct responses (i.e., correct responses)</td>
</tr>
<tr>
<td>@CorrectRT.MIN</td>
<td>Minimum reaction time for the collection of correct responses (i.e., correct responses)</td>
</tr>
</tbody>
</table>

FeedbackDisplay macros allow access to statistics calculated on a collection of responses without requiring the user to write script and set attributes. These macros provide shortcuts for displaying statistics within a SlideText sub-object on a FeedbackDisplay object only. A full list is available in the Reference Guide.
E-Basic Script Style Guidelines

Style

• Indent script contained within If..Then statements, Select..Case statements, Do..While loops, For..Next loops, etc.
• Heavily comment everything you do
• Place comments on the top of script versus to the right or below it
• Explain variables used
• Use mixed case when writing script. Avoid using all upper or lower case
• Use Debug.Print statements when possible, especially immediately before the script is throwing an error
• When concatenating strings, use multiple lines instead of keeping the entire concatenation on a single line

Avoiding common errors

• Follow style guidelines to make script portable and easy to read
• Implement only small portions of script at a time
• Test small portions of the program for functionality before expanding the program
• Use Debug.Print to send information to the Output window during runtime. This information may be evaluated after the run terminates in order to verify values, determine cause and location of errors, etc.
• Include comments in the script to delimit and identify the purpose of various sections of script
Variables in E-Basic

- Use descriptive names for variables (e.g., nResponseCount rather than x).
- Do not be afraid to use long variable names. They make script much more readable.
- Attempt to use the Long data type instead of Integer. An Integer can only hold up to 32767, while a Long can hold up to 2 billion. By using Long, you curb the possibility of an overflow error in your experiment.
- Use the g_ prefix when naming global variables (e.g. g_strText)
- Use only one Dim statement per line
- A common naming convention used in Windows programming is called "Hungarian Notation" and encourages the use of common prefixes indicating a variable's type.

<table>
<thead>
<tr>
<th>Variable Type</th>
<th>Prefix</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>str</td>
<td>strText</td>
</tr>
<tr>
<td>Long</td>
<td>n</td>
<td>nDuration</td>
</tr>
<tr>
<td>Integer</td>
<td>n</td>
<td>nOffset</td>
</tr>
<tr>
<td>Double</td>
<td>dbl</td>
<td>dblRefreshRate</td>
</tr>
<tr>
<td>Boolean</td>
<td>bool</td>
<td>boolCriteriaMet</td>
</tr>
<tr>
<td>Array</td>
<td>arr, the</td>
<td>arrMyIntegers, theIntegerArray</td>
</tr>
<tr>
<td>E-Basic Objects</td>
<td>The</td>
<td>theResponseObject, theResponseData, theDisplayCanvas</td>
</tr>
</tbody>
</table>
To increase script portability when accessing an E-Object via script, use the Rte.GetObject function. When using the Set statement, the next line of script should check for a valid reference to avoid run-time errors. At the end of any subroutine, function, or InLine, release references by setting the object to Nothing.

```vbs
Set theMouseResponseData = CMouseResponseData()
If Not theMouseResponseData Is Nothing Then
    Set theMouseResponseData = Nothing
End If
```

Bad Practices Example:

```vbs
Dim x As Integer
Dim y As Integer

For x = 1 To MouseStimulus.Input Masks(2).Responses.Count
    Dim MR As MouseResponseData
    Set MR = CMouseResponseData(MouseStimulus.Input Masks(2).Responses(x))
    y = y + 1 'increment counter
    c.SetAttrib "MouseClick" & y & "RT", MR.RT
    c.SetAttrib "MouseClick" & y & "RESP", MR.RESP
    Next x
```

Good Practices Example:

```vbs
'Declare variable to hold the object collecting the response.
Dim theResponseObject As RteRunnableInputObject
Set theResponseObject = CRteRunnableInputObject(Rte.GetObject("MouseStimulus"))

'If the assert below fires, then the object named in the line above does not exist.
Debug.Assert Not theResponseObject Is Nothing

'Counter variable for the number of mouse responses made
Dim nClickCount As Long

'Counter variable used in for loop.
Dim nIndex As Long
```
StartupInfo files can hold any information that needs to be available throughout the entire experiment, especially if it should change between runs. A good example of using global StartupInfo is specifying machine name. A good example of using local StartupInfo is specifying experimenter name before the run starts. This value should be local (used for current experiment only) because the machine will most likely have several different experimenters running subjects at different times.

In the Devices tab of the Experiment object, check the properties of the Display Device. If the Match desktop resolution at runtime property is set to Yes, E-Prime will run the experiment using the same resolution that the display is natively using. This should be set to No unless necessary for testing purposes.
Input masks have additional advanced properties customized according to device. For more information, please see section 1.3.1.2 Duration/Input Tab in the E-Prime Reference Guide. To access documentation, open any instance of E-Studio, click on “Help” in the Menu bar at the top of the screen, click “Documentation”, then select the PDF you want to open.

Inputting advanced keyboard responses such as the Spacebar and the Enter key are accomplished with the use of the “{". This would make the Spacebar look like this as an input mask “{SPACE}” without quotes. Please keep in mind that the text between the brackets is case-sensitive. A complete chart is located in E-Basic Help. To access E-Basic Help, open any instance of E-Studio, click on “Help” in the Menu bar at the top of the screen and then select E-Basic Help. The easiest way to find anything in E-Basic Help is to click on the Search tab and type whatever you are looking for into the “Search” box. Search {key} to find the chart pictured here.
PreRelease serves to eliminate the delay caused by the time it takes to set up the next object in an experiment. Using PreRelease allows the currently displayed stimulus to lend time to the following stimulus so that following stimulus can perform setup activities (e.g., reading data from a disk, getting the image ready in memory). Thus, when the second stimulus is scheduled to display, it is simply copied to the screen and any delay caused by setup activities is eliminated.
Advantages of Updating to Production Release

Refer to the following Knowledge Base topics for more information:

KB5345 - INFO: Should I Update to the Production Release, 2.0.10.242?
KB4756 - INFO: Guide to upgrading experiments to E-Prime 2.0

To download the Production Release, visit http://www.pstnet.com/support/download.asp?Type=e-prime_2_0 (requires login).
E-Run Test
E-Run Test allows automation of the test (including keyboard input) in order to speed up the testing process. To start E-Run Test, select the option from the main toolbar or from the E-Run menu. Select the speed of the test, whether Auto Response will be enabled or disabled, and whether the Experiment Advisor Report will be created.

Auto-Response
As part of E-Run Test, E-Prime can automatically respond to stimuli looking for keyboard input based on user-defined criteria. To get to this screen, navigate to the Experiment object’s Property Pages, select the Devices tab, double-click Keyboard, and select the Auto Response tab.
Optimizing timing: Generate PreRun

Generate PreRun specifies whether to load objects directly before they run (as in E-Prime 1.x and older 2.0) or at the beginning of their parent procedure (as in Production Release). Only images, movies, and sounds require load time. Loading before the object may affect inter-stimulus intervals, whereas loading at the top of the procedure will only affect inter-trial intervals. All new procedures such as TrialProc have Generate PreRun set to “top of procedure” by default, and all child objects are set to inherit that value by default. If Prime’s Generate PreRun property was set to “before object run”, there would be an unintended delay between Fixation and Prime. The defaults are set in a way that allows most experiments to get good timing without altering any values.
Miscellaneous New Features

**No Repeat After Reset**
When sampling rows from a List object using the “Random” order, exemplars from the List Object are selected at random, without replacement, until no exemplars remain. Once all of the exemplars have been selected, they are returned to the pool of available exemplars and random selection then begins again.

If No Repeat After Reset is set to Yes, E-Prime checks the first exemplar that is selected after resetting, and ensures that it does not match the exemplar that was the last item selected from the prior pool of exemplars.

<table>
<thead>
<tr>
<th>BEFORE</th>
<th>AFTER</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Orange</td>
<td>1 Orange</td>
</tr>
<tr>
<td>2 Yellow</td>
<td>2 Yellow</td>
</tr>
<tr>
<td>3 Red</td>
<td>3 Red</td>
</tr>
<tr>
<td>4 Blue</td>
<td>4 Blue</td>
</tr>
<tr>
<td>5 Green</td>
<td>5 Green</td>
</tr>
<tr>
<td>6 Green</td>
<td>6 Red</td>
</tr>
<tr>
<td>7 Red</td>
<td>7 Orange</td>
</tr>
<tr>
<td>8 Yellow</td>
<td>8 Blue</td>
</tr>
<tr>
<td>9 Orange</td>
<td>9 Green</td>
</tr>
<tr>
<td>10 Blue</td>
<td>10 Yellow</td>
</tr>
</tbody>
</table>

**Time Limit Options**
New options for the Time Limit property include *until feedback* and *end of proc*. *Until feedback* is useful for trials of varying duration that include a FeedbackDisplay object. If one of the other options was selected in that case, input could extend into or beyond the FeedbackDisplay, therefore displaying inaccurate statistics. *End of proc* is useful for trials of varying duration without FeedbackDisplay. Using infinite in that situation could result in responses mistakenly being accepted after the end of the procedure.
After running an experiment, any available links will be visible in the Output tab of the Debug window. The first link will open the .EDAT2 file associated with that run. The second link will open the Experiment Advisor Report if it was created.

Experiment Advisor assists with timing and structural issues that could arise in an experiment. The first link will take you to your .edat2 file, while the second link (the bottom one) will direct you to the Experiment Advisor Report. The Experiment Advisor Report contains a lot of very useful information about your experiment, including potential problems and timing information related to your experiment.
These diagrams show the positive effect Refresh Alignment has on the timing of an experiment. The yellow block represents the first object and the green block represents the second object. The gray line represents the intended offset time of the first object. The red line represents the actual offset time of the first object. When Refresh Alignment is enabled, the actual durations of both objects are closer to their intended durations compared to when Refresh Alignment is disabled.

For example, the second object was scheduled to be drawn at 500 ms, but the next refresh cycle was scheduled to begin at 498 ms; with Refresh Alignment enabled, the object was allowed to draw at 498 ms, but with Refresh Alignment disabled, the object was not allowed to draw until the next refresh cycle at 514 ms, resulting in an OnsetDelay of 14 ms for the second object.

Note that these examples assume Cumulative timing mode for both objects, which means the second object’s duration is shortened by its OnsetDelay. If both objects used Event timing mode, only the first object’s duration would be directly affected by the second object’s OnsetDelay.
How to Get Experiment Help

If you find that you would like a little more help with your experiment, you can create a support request by logging onto the Product Service and Support site here: http://www.pstnet.com/support/login.asp and entering your email address and password. If you have not registered for the Support site, please click on the “New User” link toward the bottom of the window and enter the information requested. Once you have logged onto the support site, you can create a support request by clicking on the “Support” button to the left. We are always happy to help you with any questions you may have.

If you would like to talk to other E-Prime users, you may want to check out the PST User Forum (http://www.pstnet.com/forum) or the E-Prime Google Group (http://groups.google.com/group/e-prime). These are both great resources for chatting with E-Prime users from around the world.