asf Tutorials

jens.schwarzbach@ukr.de

asf can be easily set up as a picture playback machine. For installation please refer to https://github.com/jvschw/asf/wiki. The stimulus-definition file *(<any>.std*) tells asf which stimuli are available, the trial-definition file *<any>.trd* tells asf when to show what and for how long, when to respond, and how results are logged.
Below we refer to <asf_install_dir> as the installation directory of asf. Usually this is ~/asf.

1 Show single pictures

This tutorial is located in <asf_install_dir>/documentation/tutorials/animals
You can run it with:

```
<asf_install_dir>/documentation/tutorials/animals
ExpInfo = ASF('stimuli.std', 'trialdefs.trd', 'test')
```

However, for didactic purposes it is recommended you create this experiment from scratch in a different folder following the instructions below.

1.1 Simple Image classification

This is an experiment to demonstrate basic functionality. Here you learn how to set up an experiment that consists of showing one image per trial and querying a manual response.
Participants are shown pictures for 2 seconds each, which they have to classify into cats (left mouse button) and dogs (right mouse button).

1.1.1 Step by step

Start up Matlab or Octave and create a directory on your computer in which the experiment resides, which will be called the experiment-home from now on. For example choose ~/exp/animals as your experiment-home

```
mkdir ~/exp/animals
cd  ~/exp/animals
```

Create a subdirectory “stimuli” in which you store the images you want to present

```
mkdir stimuli
```

Find five images of cats and five images of dogs. Check out https://www.pexels.com/search/cats/ and https://www.pexels.com/search/dogs/. Store them in ./stimuli/stimuli (name them *cats001.jpg* – *cats005.jpg*, and *dogs001.jpg* – *dogs005.jpg*, respectively)

Using Matlab's editor or any external text editor create a text file *stimuli.std* in experiment-home that lists all ten pictures with their relative path (relative with respect to experiment-home), such as

```
./stimuli/cats001.jpg
./stimuli/cats002.jpg
./stimuli/cats003.jpg
./stimuli/cats004.jpg
./stimuli/cats005.jpg
./stimuli/dogs001.jpg
./stimuli/dogs002.jpg
./stimuli/dogs003.jpg
./stimuli/dogs004.jpg
```

```
./stimuli/dogs005.jpg
```

Create a trd file trialdefs.trd that determines what to show when: We will start with showing only cat001 (the first picture in the list) and dog001 (the sixth picture in the list).

```
2 category cat dog
1 0   1 120  1 1   1
2 0   6 120  1 1   2
```

- The first line depicts design information: The experiment has one factor with two levels, the factor is called 'category', and the levels are called 'cat' and 'dog', respectively.
- The following lines depict trial information, with one line per trial. Here, the definition of a trial is show a stimulus and collect a response.
- The first column indicates the code of the experimental condition (1 for cat, 2 for dog).
- The second column allows to specify the absolute onset time of a trial (not used here, therefore always 0).
- Columns 3 and 4 depict a so called page. A page consists of a stimulus number (1 or 6 in this case) and its duration (in frames). With a frame rate of 60 Hz, the value 120 means 2 seconds.
- Columns 5 and 6 depict on which page to start and end response collection (here 1 and 1).
- Column 7 indicates the correct response key (1 for cat, 2 for dog).

Run this very first version, the syntax is `ExpInfo = ASF(stimFileName, trialFileName, expName, [Cfg])`. In our case you can start the experiment with:

```
ExpInfo = ASF('stimuli.std', 'trialdefs.trd', 'test')
```

To show the entire stimulus set, create a corresponding trd file *trialdefs_10.trd*:

```
2 category cat dog
1 0   1 120  1 1   1
1 0   2 120  1 1   1
1 0   3 120  1 1   1
1 0   4 120  1 1   1
1 0   5 120  1 1   1
2 0   6 120  1 1   2
2 0   7 120  1 1   2
2 0   8 120  1 1   2
2 0   9 120  1 1   2
2 0  10 120  1 1   2
```

Run the experiment

```
ExpInfo = ASF('stimuli.std', 'trialdefs_10.trd', 'test' )
```

You will notice that the trials are played in the order in which they have been defined. In general, asf leaves it up to the experimenter to provide properly shuffled trial-definition files, but you can also configure asf to shuffle the presentations for you. To this aim, you create a variable Cfg, which can have many different fields (check `help asf` on the command line).

```
Cfg.randomizeTrials = 1
ExpInfo = ASF('stimuli.std', 'trialdefs_10.trd', 'test', Cfg)
```

Now you have your first animal-categorization experiment in asf, which presents pictures of cats and dogs in randomized order and records your responses (which key and which reaction time).

Additional tasks

- Change response mapping (keyboard instead of mouse keys)

2 Show sequence of pictures

This tutorial is located in <asf_install_dir>/documentation/tutorials/animals_dm
You can run it with:

```
<asf_install_dir>/documentation/tutorials/animals_dm
ExpInfo = ASF('stimuli.std', 'trialdefs.trd', 'test')
```

Usually, trials in an experiment contain more than a single page. In this tutorial we will develop a little experiment in which we ask participants to judge whether two subsequent pictures show animals of the same or different categories.

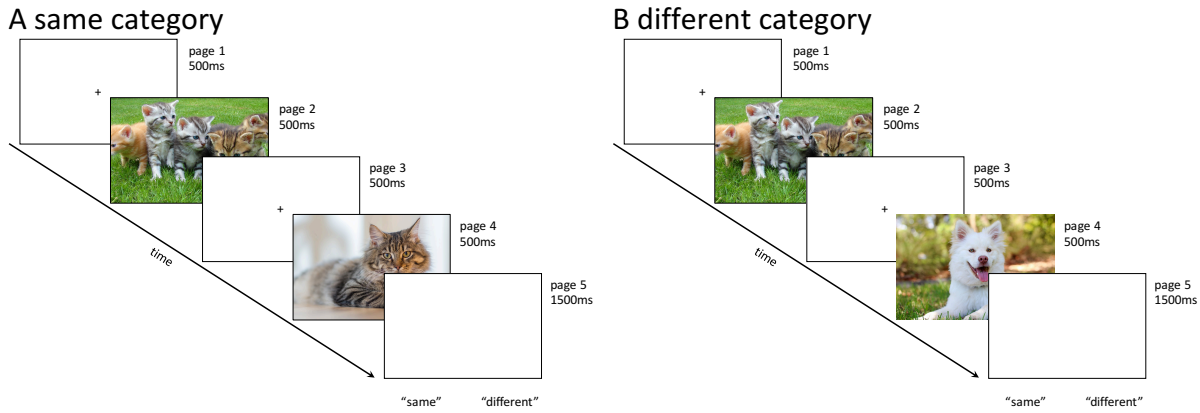


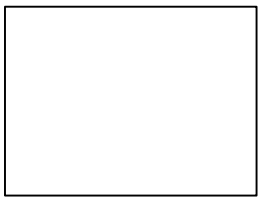
Figure 1. Example trials for a categorical decision making task (A) "same category" trial. B) "different category" trial). Each trial contains 5 pages. A trial starts with presenting a fixation cross for 500ms, followed by picture 1 (cats or dogs) for 500ms, a fixation cross (500 ms), picture 2 (cats or dogs) for 500 ms. The last page shows an empty screen for 1500ms during which participants are asked to indicate whether the two pictures showed animals of the same species (left mouse button) or different species (right mouse button).

We will use six pictures, storing their names in a textfile called stimuli.std

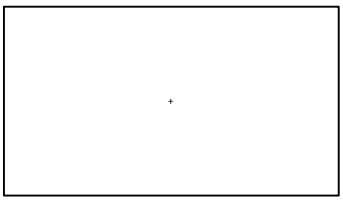
```
./stimuli/stim_empty.bmp
./stimuli/stim_fix.bmp
./stimuli/cats001.jpg
./stimuli/cats002.jpg
./stimuli/dogs001.jpg
./stimuli/dogs002.jpg
```

Internally asf refers to the stimuli in the order they were listed in the std-file (see Table 1).

picture 1: stim_empty



picture2: stim_fix.bmp



picture 3: cats001.jpg

picture 4: cats002.jpg

picture 5: dogs001.jpg

picture 6: dogs002.jpg

Table 1. Pictures are internally numbered by the order provided in the std-file.

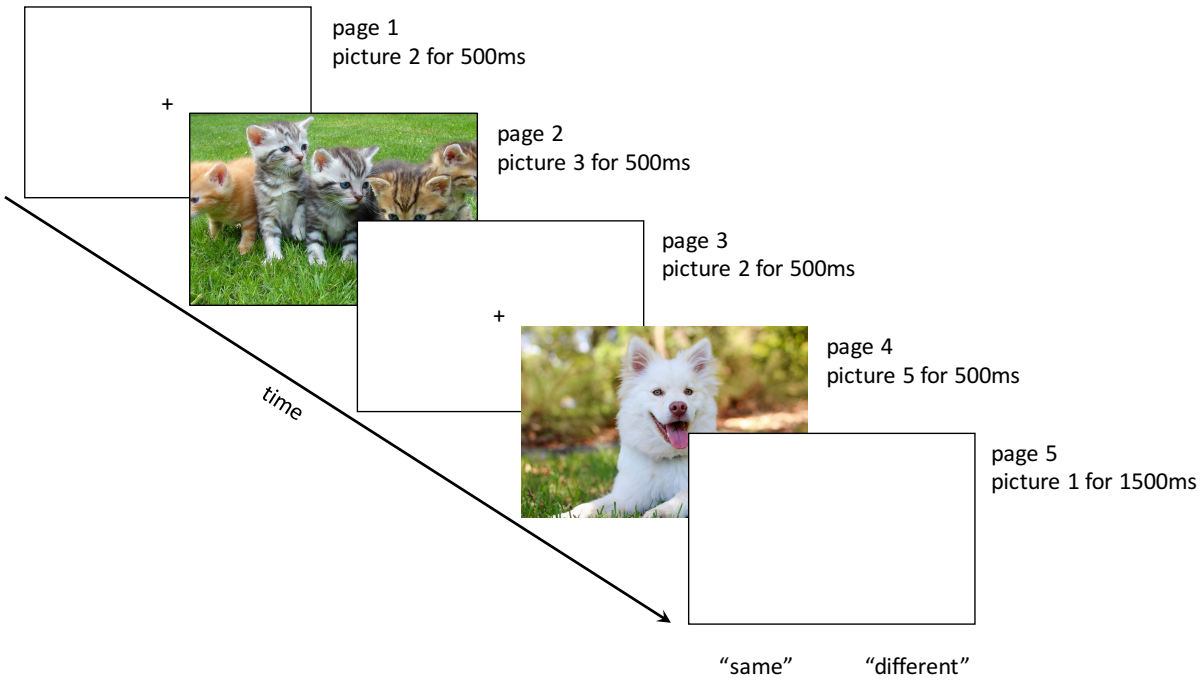


Figure 2. Example trial

We can describe the stimulus presentation as a sequence of 5 pages with each page being defined by two numbers: picture-number and page-duration (in frames).

page 1		page 2		page 3		page 4		page 5	
pic	dur	pic	dur	pic	dur	pic	dur	pic	dur
2	18	3	18	2	18	5	18	1	90

Table 2. Stimulus presentation in a given trial described as a sequence of pages on which a certain picture is shown for a certain time.

A trd-file (trialdefs_1.trd) for a single-trial experiment might look like this

```
2 category same different
2 0   2 18   3 18   2 18   5 18  1 90   5 5   2
```

You can tun this single-trial experiment with:

```
ExpInfo = ASF('stimuli.std', 'trialdefs_1.trd', 'test')
```

A more complete experiment consists of a number of trials that cover the different experimental conditions, here "same-category" or "different-category". A corresponding trd-file trialdefs_12.trd is listed here:

```
2 category same different
1 0   2 18   3 18   2 18   4 18  1 90   5 5   1
1 0   2 18   4 18   2 18   3 18  1 90   5 5   1
1 0   2 18   5 18   2 18   6 18  1 90   5 5   1
1 0   2 18   6 18   2 18   5 18  1 90   5 5   1
2 0   2 18   3 18   2 18   5 18  1 90   5 5   2
2 0   2 18   5 18   2 18   3 18  1 90   5 5   2
2 0   2 18   3 18   2 18   6 18  1 90   5 5   2
2 0   2 18   6 18   2 18   3 18  1 90   5 5   2
2 0   2 18   4 18   2 18   5 18  1 90   5 5   2
2 0   2 18   5 18   2 18   4 18  1 90   5 5   2
2 0   2 18   4 18   2 18   6 18  1 90   5 5   2
2 0   2 18   6 18   2 18   4 18  1 90   5 5   2
```

You can run this mini experiment with:

```
Cfg.randomizeTrials = 1;
ExpInfo = ASF('stimuli.std', 'trialdefs_12.trd', 'test', Cfg)
```

2.1 Analyzing the results

TBD

2.2 Programming trd-file generation

There are many reasons why you may want to generate trd-files under program-control:

- hand editing a trd-file can be tedious
- the builtin algorithm for shuffling trials is very simple (no functionality for avoiding certain sequences)
- in fmri experiments you may want to control aspects of trial-timing that require some sophistication
- in realtime-fMRI experiments you may want to know the trial-order and timing before you actually run the experiment

A trd-file generator should have the following elements:

- make trial-definitions (highly experiment specific)
- shuffle trials (simple shuffling or complex restrictions)
- write trd file (generic)

2.2.1 Make trial-definitions

The following section describes an approach on how to generate trd-files for the "same-different task"

Let's review the structure of one trial

code	tOn	UDC	page 1		page 2		page 3		page 4		page 5		sRT	eRT	c
			pic	dur	pic	dur	pic	dur	pic	dur	pic	dur			
1	0	-	2	18	3	18	2	18	4	18	1	90	5	5	2

Table 3. Full description of a trial comprising stimulus code (column 1), optional trial onset time (column 2), optional user-defined columns (UDC) (column 3, actual number is defined by Cfg.userDefinedSTMcolumns, here 0), nPages pairs of values (pictureNumber, pageDuration), and the final three columns refering to the page on which response collection starts, the page after which response collection ends, and the code for the correct response.

and actually generate a data-structure thisTrial that represents this information

```
thisTrial.code = 1;
thisTrial.tOnset = 0;
thisTrial.userDefined = [];
thisTrial.pages = [2, 3, 2, 4, 1];
thisTrial.durations = [18, 18, 18, 18, 90];
thisTrial.startRTonPage = 5;
thisTrial.endRTonPage = 5;
thisTrial.correctResponse = 1;
```

Now, we assign this trialdefinition to the first entry of a vector of trial definitions.

```
TrialDefinitions(1) = thisTrial;
```

In the next step we can assign the defining values of another experimental trial to thisTrial,

```
thisTrial.code = 1;
thisTrial.tOnset = 0;
thisTrial.userDefined = [];
thisTrial.pages = [2, 4, 2, 3, 1];
thisTrial.durations = [18, 18, 18, 18, 90];
thisTrial.startRTonPage = 5;
thisTrial.endRTonPage = 5;
thisTrial.correctResponse = 1;
```

and assign this trial definition to the second entry of a vector of trial definitions:

```
TrialDefinitions(2) = thisTrial;
```

This yields a vector of trial definitions

```
TrialDefinitions =
1x2 struct array with fields:
    code
    tOnset
    userDefined
    pages
    durations
    startRTonPage
    endRTonPage
    correctResponse
```

Following this scheme, we can create a vector of trial definitions that contains all trials that are necessary for our experiment. There are two noteworthy observations: 1. Creating a list of trials is a repetitive process during which many things stay the same. 2. Within a given experiment, often the only things that change are a) the code of a trial, b) which pictures to show on which page, and c) what counts as a correct response.

```
function makeTRD_animals_dm(trdName)
```

```
%function makeTRD_animals_dm(trdName)
%makeTRD_animals_dm('animals_dm.trd')

%SOMETHING ABOUT THE HARDWARE
Cfg.userDefinedSTMcolumns = 0;

Info.factorialStructure = 2; %one factor

Info.factorLevels = [0, 1]; %same category, different category
Info.nFactorLevels = numel(Info.factorLevels);
Info.nReps = 1; %number of replications per valid stimulus-configuration

Info.blankPic = 1;
Info.fixPic = 2;
Info.fac(1).pictures = [3, 4];
Info.fac(2).pictures = [5, 6];

%MAKE TRIALS
TrialDefinitions = makeTrialDefs(Info);

% RANDOMIZE TRIALS
TrialDefinitions = shuffleTrials(TrialDefinitions);

% WRITE OUT TRD TO DISK
writeTRD(TrialDefinitions, Info, trdName, Cfg)
```

The next function makeTrialDefs() is the one that you have to adjust to your experimental needs. Here, we keep it really simple. The task of this function is to create a vecor of trial definitions similar to our 'manual' procedure on the previous page.

```
function TrialDefinitions = makeTrialDefs(Info)
items = horzcat(Info.fac.pictures);

% Begin looping through factors to build trials
trialCounter = 0;
for i = 1:numel(items)
    item1 = items(i);
    %WHICH CATEGORY DOES ITEM 1 BELONG TO?
    for x = 1:Info.nFactorLevels
        if ismember(item1, Info.fac(x).pictures)
            catItem1 = x;
        end
    end

    for j = 1:numel(items)
        item2 = items(j);
        %WHICH CATEGORY DOES ITEM 1 BELONG TO?
        for y = 1:Info.nFactorLevels
            if ismember(item2, Info.fac(y).pictures)
                catItem2 = y;
            end
        end
        %DO THE TWO CRITICAL PICTURES (ITEMS) BELONG TO THE SAME CATEGORY?
        isSameCat = catItem1 == catItem2;

        %--------------------------
        % CODE
        %--------------------------
        thisTrial.code = ASF_encode(isSameCat, Info.factorialStructure);

        %--------------------------
        % TONSET
        %--------------------------
```

```
        thisTrial.tOnset = 0;

        %--------------------------
        % PAGES AND DURATIONS:
        %the only variable thing is the content of page 3
        %--------------------------
        thisTrial.page =...
            [Info.fixPic item1 Info.fixPic item2 Info.blankPic];
        thisTrial.durations = [18 18 18 18 90];

        %--------------------------
        % START RESPONSE PAGE
        %--------------------------
        thisTrial.startRTonPage = 5;

        %--------------------------
        % END RESPONSE PAGE
        %--------------------------
        thisTrial.endRTonPage = 5;

        %--------------------------
        % CORRECT RESPONSE
        %--------------------------
        if isSameCat
            thisTrial.correctResponse = 1; %LEFT MOUSE
        else
            thisTrial.correctResponse = 2; %RIGHT MOUSE
        end

        % Loop through each repetition of the given condition
        for iRep = 1:Info.nReps
            % Increment counter
            trialCounter = trialCounter + 1;
            %--------------------------
            % COPY INTO ARRAY OF TRIAL DEFINITIONS
            %--------------------------
            TrialDefinitions(trialCounter) = thisTrial;
        end %End rep

    end %End item2
end %End item1
```

The shuffling procedure described here also keeps it really simple. You can implement more complicated ones for example if you want to shuffle under certain restrictions.

```
function shuffledTrialDefinitions = shuffleTrials(TrialDefinitions)
% This subfunction randomizes the trials.
nTrials = length(TrialDefinitions);

% This is a simple randomization
shuffledTrialDefinitions = TrialDefinitions(randperm(nTrials));
```

The next function writes the trd file to disk. There is no need to change this function for your experiments

```
function TRDfileName = writeTRD(TrialDefinitions, Info, trdName, Cfg)
% This subfunction writes out the trd-file to a text file
fprintf(1, 'WRITING %s\n', trdName);

%Open the text file for writing
fid = fopen(trdName, 'wt');

%-------------------------------------------------------------------------
```

```
%FIRST LINE OF THE TRD FILE
%Write out the factoral structure onto the first line of the text file
fprintf(fid,'%d   ', Info.factorialStructure);
if isfield(Info, 'factorNames')
    %Write out the names of the factors onto the first line of the text
file
    fprintf(fid,'%s   ', Info.factorNames{:});
end

if isfield(Info, 'levelNames')
    %Then write out the names of the levels of each factor
    fprintf(fid,'%s   ', Info.levelNames{:});
end

nTrials = length(TrialDefinitions);

%------------------------------------------------------------------------
%Loop through TrialDefinitions
for i = 1:nTrials

    % Jump to the next line in the text file
    fprintf(fid,'\n');

    % CODE
    fprintf(fid, '%d  ', TrialDefinitions(i).code);

    % TONSET
    fprintf(fid, '%8.3f    ', TrialDefinitions(i).tOnset);

    % USER DEFINED COLUMNS CONTAIN STIMULUS PARAMETERS
    if isfield(TrialDefinitions(i), 'userDefined')
        if numel(TrialDefinitions(i).userDefined) > 0
            fprintf(fid, '%4d ', TrialDefinitions(i).userDefined);
            fprintf(fid, '  ');
        end
    end

    % PAGES AND DURATIONS
    fprintf(fid, '%2d %3d   ', [TrialDefinitions(i).page(:),...
        TrialDefinitions(i).durations(:)]');
    fprintf(fid, '  ');

    % RESPONSE PARAMS
    fprintf(fid, '%d  ', TrialDefinitions(i).startRTonPage,...
        TrialDefinitions(i).endRTonPage,...
        TrialDefinitions(i).correctResponse);
end
fclose(fid);
```

3 Online rendering by means of a plugin: Show one picture or a sequence of pictures at different locations

This tutorial is located in <asf_install_dir>/documentation/tutorials/dots_example
You can run it with:

```
<asf_install_dir>/documentation/tutorials/dots_example
run_dots
```

Imagine you want to program a perimetry experiment in which a small stimulus occurs at different locations on the screen and the participant must press a button in order to report whether she or he has detected a stimulus. in the end, the experimenter can analyze the speed and accuracy of visual detection as a function of stimulus location.
Let's say we want to test 25 locations on a 5x5 grid around the center of the screen.

-250, 250	-125, 250	0, 250	125, 250	250, 250
-250, 125	-125, 125	0, 125	125, 125	250, 125
-250, 0	-125, 0	0,0	125, 0	250, 0
-250, -125	-125, -125	0, -125	125, -125	250, -125
-250, -250	-125, -250	0, -250	125, -250	250, -250

Table 4. Positions of stimuli relative to center.

One way to achieve this would be to create 25 bitmaps each of which containing one stimulus at a defined location. Here we explore a more efficient alternative. We create one bitmap that contains the stimulus in the center of the image, and we present this bitmap at different positions on the screen.
So far you have learned that the default behavior of asf, which is to present one picture per frame, and that this picture is presented centrally. This is achieved by the built-in function ASF_showTrial.m (e.g. line numbers 89, 147, and 217 appendix).

```
Screen('DrawTexture',                                    windowPtr,...
     Stimuli.tex(atrial.pageNumber(iPage)));
```

We will write our own ShowTrial function (ASF_showTrialDotXY.m), in which we add a little piece of code that draws a dot at a certain location relative to the screen-center.

We will also have to tell our function what stimPosX and stimPosY are early on in the function, which we can define early on in the code (near original line 48)

```
stimPosX = atrial.userDefined(1);
stimPosY = atrial.userDefined(2);
```

Add right after the original line 147, which is

```
Screen('DrawTexture',                                    windowPtr,...
     Stimuli.tex(atrial.pageNumber(iPage)));

the following code
drawFix(windowPtr, Cfg, fixColor)
if iPage == atrial.startRTonPage
    Screen('DrawDots', windowPtr,...
    [ Cfg.Screen.centerX + stimPosX,...
    Cfg.Screen.centerY + stimPosY], 10, [160 160 160], [], 2);
end
```

This piece of code will draw a grey dot at the indicated stimulus position.
This also requires a certain type of trd-file, such as

```
5 5 px -250 -125 0 125 250 py -250 -125 0 125 250
1 0   -250 -250  1 1   1 120   1 2   1
1 0   -125 -250  1 1   1 120   1 2   1
1 0      0 -250  1 1   1 120   1 2   1
1 0    125 -250  1 1   1 120   1 2   1
1 0    250 -250  1 1   1 120   1 2   1
1 0   -250 -125  1 1   1 120   1 2   1
1 0   -125 -125  1 1   1 120   1 2   1
1 0      0 -125  1 1   1 120   1 2   1
1 0    125 -125  1 1   1 120   1 2   1
1 0    250 -125  1 1   1 120   1 2   1
1 0   -250    0  1 1   1 120   1 2   1
1 0   -125    0  1 1   1 120   1 2   1
1 0      0    0  1 1   1 120   1 2   1
```

```
1  0     125     0   1  1    1  120    1  2    1
1  0     250     0   1  1    1  120    1  2    1
1  0    -250   125   1  1    1  120    1  2    1
1  0    -125   125   1  1    1  120    1  2    1
1  0       0   125   1  1    1  120    1  2    1
1  0     125   125   1  1    1  120    1  2    1
1  0     250   125   1  1    1  120    1  2    1
1  0    -250   250   1  1    1  120    1  2    1
1  0    -125   250   1  1    1  120    1  2    1
1  0       0   250   1  1    1  120    1  2    1
1  0     125   250   1  1    1  120    1  2    1
1  0     250   250   1  1    1  120    1  2    1
```

As usual, the first two columns encode the trial-code and the onset time, both of which can be ignored here.
Columns 3 and 4 are the two user-supplied columns (relative x-position, relative y-position).
Columns 5 and 6 refer to page 1 (page, duration) on which we render our stimulus).
Columns 7 and 8 refer to page 2 (page, duration), which are part of the response period.
Columns 9 and 10 indicate the start- and end-pages for response collection.
Column 11 indicates the correct response (button 1).

4 Show several pictures at once

Here you learn how to set up an experiment that consists of showing two images side by side per trial and querying a manual response. Participants are shown two pictures of different animal categories (cats and dogs) for two seconds. One category will be on the left side of the screen, the other on the right. Which category is where will be randomized. Participants have to manually indicate the location (left or right) of the cat(s) using the left and right mouse button, respectively.

Figure 3. Sample trial from a " Where is the cat" experiment. Two pictures are presented simultaneously to the left and right of the center of the screen. One depicts a cat or cats, the other a dog or dogs. Participants indicate manually on which side the see a cat or cats.

4.1 Preparatory work

We create the experiment's base directory and switch into it

```
mkdir ~/exp/animals_sim
cd ~/animals_sim/
```

Copy the stimuli from the animals tutorial.

```
copyfile('~/exp/animals/stimuli/*', './stimuli')
```

Use some picture editor and create an empty (i.e. white) picture and save it under ./stimuli/stim_empty.bmp

Copy the stimuli and the stimulus definition file from the animals tutorial.

```
copyfile('~/exp/animals/stimuli.std', 'stimuli.std')
```

Edit this file and add ./stimuli/stim_empty.bmp at the beginning such that it now contains:

```
./stimuli/stim_empty.bmp
./stimuli/cats001.jpg
./stimuli/cats002.jpg
./stimuli/cats003.jpg
./stimuli/cats004.jpg
./stimuli/cats005.jpg
./stimuli/dogs001.jpg
./stimuli/dogs002.jpg
./stimuli/dogs003.jpg
./stimuli/dogs004.jpg
./stimuli/dogs005.jpg
```

4.2 Strategy

So far you have learned how to show one picture per page. This is achieved by the built-in function ASF_showTrial.m. You may want to spend some time in trying to understand this very

central function (see appendix). If you want to show several pictures on the same page you have to determine their respective location. A very tedious approach would be to create new combined images by placing two images on the same canvas in an image processing program and store them as combined stimuli. This takes a lot of time, and you may end up having to create a large number of compound stimuli.
A much more efficient approach is to have asf combine these pictures online. To achieve that, you need to tell ASF to use a different function to present stimuli. Rather than using the builtin function ASF_showTrial.m use ASF_showTrial2Pics.m, which puts two pictures on the screen according to parameters you provide in the trd-file.
To this aim save a copy of ASF_showTrial2Pics.m to your local directory (because this allows you editing and adapting this function to your own needs):

```
copyfile(fullfile(fileparts(which('asf')),...
'ASF_showTrial_plugins', 'ASF_showTrial2Pics.m'), '.')
```

Remember, the empty picture is the first picture in our std-file, cats have the indices 2-6, and dogs have the indices 7-11.
The function ASF_showTrial2Pics expects two so-called userDefinedSTMcolumns, which are extra entries in a trd-file.
A trd-file (trialdefs_short.trd) with only two trials, might look like this with userDefinedSTMcolumns printed in boldface:

```
2 whereiscat left right
1 0  2 7     1 120  1 30   1 1   1
2 0  7 2     1 120  1 30   1 1   2
```

The first line depicts design information: The experiment has one factor with two levels, the factor is called ‘whereiscat, and the levels are called ‘left’ and ‘right’, respectively.
The following lines depict trial information, with one line per trial. Here, the definition of a trial is show a stimulus and collect a response.
The first column indicates the code of the experimental condition (1 for left, 2 for right).
The second column allows to specify the absolute onset time of a trial (not used here, therefore always 0).
Columns 3 and 4 depict the userDefinedSTMcolumns. userDefinedSTMcolumn1 denotes the index of the picture presented on the left. userDefinedSTMcolumn2 denotes the index of the picture presented on the right.

Columns 5 and 6 refer to the first page of the trial (show ./stimuli/stim_empty.bmp for 120 frames).
Columns 7 and 8 refer to the second page of the trial (show ./stimuli/stim_empty.bmp for 30 frames).

Columns 9 and 10 depict on which page to start and end response collection (here 1 and 1).
Column 11 indicates the correct response key (1 for left, 2 for right).

The plugin ASF_showTrial2Pics.m is written such that it interprets trial-definitions accordingly:

Lines 60-77 define the variables destinationRect1 and destinationRect1, which depict rectangular selections to the left and right of the center of the screen.

Lines 80-81

```
pic1 = atrial.userDefined(1);
```

```
pic2 = atrial.userDefined(2);
```

Lines 133-142
draw the image (an empty picture),
put another image (pic1) on the left (destinationRect1), and
put another image (pic2) on the right (destinationRect2)

```
%PUT THE APPROPRIATE TEXTURE(S) ON THE BACK BUFFER
Screen('DrawTexture', windowPtr, Stimuli.tex(atrial.pageNumber(i)));
if pic1 > 0
     %if valid picture index provided, put it on the left
    Screen('DrawTexture', windowPtr, Stimuli.tex(pic1), [], destinationRect1);
end
if pic2 > 0
    %if valid picture index provided, put it on the right
    Screen('DrawTexture', windowPtr, Stimuli.tex(pic2), [], destinationRect2)
end
```

In order for asf to use ASF_showTrial2Pics-plugin instead of the builtin default ShowTrial-function one needs to run asf with a configuration variable.

```
Cfg = [];
Cfg.userSuppliedTrialFunction = @ASF_showTrial2Pics;
Cfg.userDefinedSTMcolumns = 2;
ExpInfo = ASF('stimuli.std', 'trialdefs_short.trd', 'test', Cfg)
```

To get all possible combinations of all cats and all dogs on either side you need to define a trd file with 51 lines:

```
2 category catleft catright
1 0   2 7      1 120   1 30    1 1    1
1 0   2 8      1 120   1 30    1 1    1
1 0   2 9      1 120   1 30    1 1    1
1 0   2 10     1 120   1 30    1 1    1
1 0   2 11     1 120   1 30    1 1    1
1 0   3 7      1 120   1 30    1 1    1
1 0   3 8      1 120   1 30    1 1    1
1 0   3 9      1 120   1 30    1 1    1
1 0   3 10     1 120   1 30    1 1    1
1 0   3 11     1 120   1 30    1 1    1
1 0   4 7      1 120   1 30    1 1    1
1 0   4 8      1 120   1 30    1 1    1
1 0   4 9      1 120   1 30    1 1    1
1 0   4 10     1 120   1 30    1 1    1
1 0   4 11     1 120   1 30    1 1    1
1 0   5 7      1 120   1 30    1 1    1
1 0   5 8      1 120   1 30    1 1    1
1 0   5 9      1 120   1 30    1 1    1
1 0   5 10     1 120   1 30    1 1    1
1 0   5 11     1 120   1 30    1 1    1
1 0   6 7      1 120   1 30    1 1    1
1 0   6 8      1 120   1 30    1 1    1
1 0   6 9      1 120   1 30    1 1    1
1 0   6 10     1 120   1 30    1 1    1
```

```
1  0   6  11     1  120   1  30    1  1    1
2  0   7  2      1  120   1  30    1  1    2
2  0   7  3      1  120   1  30    1  1    2
2  0   7  4      1  120   1  30    1  1    2
2  0   7  5      1  120   1  30    1  1    2
2  0   7  6      1  120   1  30    1  1    2
2  0   8  2      1  120   1  30    1  1    2
2  0   8  3      1  120   1  30    1  1    2
2  0   8  4      1  120   1  30    1  1    2
2  0   8  5      1  120   1  30    1  1    2
2  0   8  6      1  120   1  30    1  1    2
2  0   9  2      1  120   1  30    1  1    2
2  0   9  3      1  120   1  30    1  1    2
2  0   9  4      1  120   1  30    1  1    2
2  0   9  5      1  120   1  30    1  1    2
2  0   9  6      1  120   1  30    1  1    2
2  0   10  2      1  120   1  30    1  1    2
2  0   10  3      1  120   1  30    1  1    2
2  0   10  4      1  120   1  30    1  1    2
2  0   10  5      1  120   1  30    1  1    2
2  0   10  6      1  120   1  30    1  1    2
2  0   11  2      1  120   1  30    1  1    2
2  0   11  3      1  120   1  30    1  1    2
2  0   11  4      1  120   1  30    1  1    2
2  0   11  5      1  120   1  30    1  1    2
2  0   11  6      1  120   1  30    1  1    2
```

11

11.1 Understanding the builtin function ASF_showTrial.m

```
function TrialInfo = ShowTrial(atrial, windowPtr, Stimuli, Cfg)
if ~isfield(Cfg, 'feedbackResponseNumber'), Cfg.feedbackResponseNumber = 1; else end; %IF YOU WANT TO GIVE
FEEDBACK REFER BY DEFAULT TO THE FIRST RESPONSE GIVEN IN THIS TRIAL
%SAVE TIME BY ALLOCATING ALL VARIABLES UPFRONT

% VBLTimestamp system time (in seconds) when the actual flip has happened
% StimulusOnsetTime An estimate of Stimulus-onset time
% FlipTimestamp is a timestamp taken at the end of Flip's execution
VBLTimestamp = 0; StimulusOnsetTime = 0; FlipTimestamp = 0; Missed = 0;
Beampos = 0;

StartRTMeasurement = 0; EndRTMeasurement = 0;
timing = [0, VBLTimestamp, StimulusOnsetTime, FlipTimestamp, Missed, Beampos];
nPages = length(atrial.pageNumber);
timing(nPages, end) = 0;
this_response = [];

%ON PAGES WITH WITH RESPONSE COLLECTION MAKE SURE THE CODE RETURNS IN TIME
%BEFORE THE NEXT VERTICAL BLANK. FOR EXAMPLE IF THE RESPONSE WINDOW IS 1000
%ms TOLERANCE MAKES THE RESPONSE COLLECTION CODE RETURN AFTER 1000ms-0.3
%FRAMES, I.E. AFTER 995 ms AT 60Hz
toleranceSec = Cfg.Screen.monitorFlipInterval*0.3;

%HOWEVER, THIS MUST NOT BE LONGER THAN ONE FRAME
%DURATION. EXPERIMENTING WITH ONE QUARTER OF A FRAME
responseGiven = 0;
this_response.key = [];
this_response.RT = [];


%-----------------------------------------------------------------------
%TRIAL PRESENTATION HAS SEVERAL PHASES
% 1) WAIT FOR THE RIGHT TIME TO START TRIAL PRESENTATION. THIS MAY BE
%    IMMEDIATELY OR USER DEFINED (E.G. IN fMRI EXPERIMENTS)
%
% 2) LOOP THROUGH PAGE PRESENTATIONS WITHOUT RESPONSE COLLECTION
```

```
%
% 3) LOOP THROUGH PAGE PRESENTATIONS WHILE CHECKING FOR USER INPUT/RESPONSES
%
% 4) LOOP THROUGH PAGE PRESENTATIONS WITHOUT RESPONSE COLLECTION
%    (AFTER RESPONSE HAS BEEN GIVEN)
%
% 5) FEEDBACK
%--------------------------------------------------------------------------

%IF YOU WANT TO DO ANY OFFLINE STIMULUS RENDERING (I.E. BEFORE THE TRIAL
%STARTS), PUT THAT CODE HERE
responseCounter = 0; %COUNTS THE NUMBER OF RESPONSES GIVEN WITHIN THE ALLLOWED PERIOD
%LOG DATE AND TIME OF TRIAL
strDate = datestr(now); %store when trial was presented

%--------------------------------------------------------------------------
% PHASE 1) WAIT FOR THE RIGHT TIME TO START TRIAL PRESENTATION. THIS MAY BE
% IMMEDIATELY OR USER DEFINED (E.G. IN fMRI EXPERIMENTS)
%--------------------------------------------------------------------------

%IF EXTERNAL TIMING REQUESTED (e.g. fMRI JITTERING)
if Cfg.useTrialOnsetTimes
    wakeupTime = WaitSecs('UntilTime', Cfg.experimentStart + atrial.tOnset);
else
    wakeupTime = GetSecs;
end
%LOG TIME OF TRIAL ONSET WITH RESPECT TO START OF THE EXPERIMENT
%USEFUL FOR DATA ANALYSIS IN fMRI
tStart = wakeupTime - Cfg.experimentStart;


if Cfg.Eyetracking.doDriftCorrection
    EyelinkDoDriftCorrect(Cfg.Eyetracking.el);
end

%--------------------------------------------------------------------------
```

```
%END OF PHASE 1
%-------------------------------------------------------------------------

%MESSAGE TO EYELINK
Cfg = ASF_sendMessageToEyelink(Cfg, 'TRIALSTART');

%-------------------------------------------------------------------------
% PHASE 2) LOOP THROUGH PAGE PRESENTATIONS WITHOUT RESPONSE COLLECTION
%-------------------------------------------------------------------------
%CYCLE THROUGH PAGES FOR THIS TRIAL
atrial.nPages = length(atrial.pageNumber);
for iPage = 1:atrial.startRTonPage-1
    if (iPage > atrial.nPages)
        break;
    else
        %PUT THE APPROPRIATE TEXTURE ON THE BACK BUFFER
        Screen('DrawTexture', windowPtr, Stimuli.tex(atrial.pageNumber(iPage)));

        %PRESERVE BACK BUFFER IF THIS TEXTURE IS TO BE SHOWN
        %AGAIN AT THE NEXT FLIP
        bPreserveBackBuffer = atrial.pageDuration(iPage) > 1;

        %FLIP THE CONTENT OF THIS PAGE TO THE DISPLAY AND PRESERVE IT IN THE
        %BACKBUFFER IN CASE THE SAME IMAGE IS TO BE FLIPPED AGAIN TO THE SCREEN
        [VBLTimestamp StimulusOnsetTime FlipTimestamp Missed Beampos] =...
            ASF_xFlip(windowPtr, Stimuli.tex(atrial.pageNumber(iPage)),...
            Cfg, bPreserveBackBuffer);

        %SET TRIGGER (PARALLEL PORT AND EYELINK)
        ASF_setTrigger(Cfg, atrial.pageNumber(iPage));


        %LOG WHEN THIS PAGE APPEARED
        timing(iPage, 1:6) = [atrial.pageDuration(iPage), VBLTimestamp,...
            StimulusOnsetTime FlipTimestamp Missed Beampos];

```

```
109
110         %WAIT OUT STIMULUS DURATION IN FRAMES. WE USE PAGE FLIPPING RATHER
111         %THAN A TIMER WHENEVER POSSIBLE BECAUSE GRAPHICS BOARDS PROVIDE
112         %EXCELLENT TIMING; THIS IS THE REASON WHY WE MAY WANT TO KEEP A
113         %STIMULUS IN THE BACKBUFFER (NONDESTRUCTIVE PAGE FLIPPING)
114         %NOT ALL GRAPHICS CARDS CAN DO THIS. FOR CARDS WITHOUT AUXILIARY
115         %BACKBUFFERS WE COPY THE TEXTURE EXPLICITLY ON THE BACKBUFFER AFTER
116         %IT HAS BEEN DESTROYED BY FLIPPING
117         nFlips = atrial.pageDuration(iPage) - 1; %WE ALREADY FLIPPED ONCE
118         for FlipNumber = 1:nFlips
119             %PRESERVE BACK BUFFER IF THIS TEXTURE IS TO BE SHOWN
120             %AGAIN AT THE NEXT FLIP
121             bPreserveBackBuffer = FlipNumber < nFlips;
122
123             %FLIP THE CONTENT OF THIS PAGE TO THE DISPLAY AND PRESERVE IT
124             %IN THE BACKBUFFER IN CASE THE SAME IMAGE IS TO BE FLIPPED
125             %AGAIN TO THE SCREEN
126             ASF_xFlip(windowPtr, Stimuli.tex(atrial.pageNumber(iPage)),...
127                 Cfg, bPreserveBackBuffer);
128         end
129     end
130 end
131 %-----------------------------------------------------------------------
132 %END OF PHASE 2
133 %-----------------------------------------------------------------------
134
135 %-----------------------------------------------------------------------
136 % PHASE 3) LOOP THROUGH PAGE PRESENTATIONS WHILE CHECKING FOR USER
137 %          INPUT/RESPONSES
138 %-----------------------------------------------------------------------
139 %SPECIAL TREATMENT FOR THE DISPLAY PAGES ON WHICH WE ALLOW REACTIONS
140
141 for iPage = atrial.startRTonPage:atrial.endRTonPage
142     if (iPage > atrial.nPages)
143         break;
144     else
```

```
145
146         %PUT THE APPROPRIATE TEXTURE ON THE BACK BUFFER
147         Screen('DrawTexture', windowPtr, Stimuli.tex(atrial.pageNumber(iPage)));
148
149         %DO NOT PUT THIS PAGE AGAIN ON THE BACKBUFFER, WE WILL WAIT IT OUT
150         %USING THE TIMER NOT FLIPPING
151         bPreserveBackBuffer = 0;
152
153         %FLIP THE CONTENT OF THIS PAGE TO THE DISPLAY AND PRESERVE IT
154         %IN THE BACKBUFFER IN CASE THE SAME IMAGE IS TO BE FLIPPED
155         %AGAIN TO THE SCREEN
156         [VBLTimestamp StimulusOnsetTime FlipTimestamp Missed Beampos] =...
157             ASF_xFlip(windowPtr, Stimuli.tex(atrial.pageNumber(iPage)),...
158             Cfg, bPreserveBackBuffer);
159
160         %SET TRIGGER
161         ASF_setTrigger(Cfg, atrial.pageNumber(iPage));
162
163         if iPage == atrial.startRTonPage
164             StartRTMeasurement = VBLTimestamp;
165         end
166
167         %STORE TIME OF PAGE FLIPPING FOR DIAGNOSTIC PURPOSES
168         timing(iPage, 1:6) = [atrial.pageDuration(iPage), VBLTimestamp,...
169             StimulusOnsetTime, FlipTimestamp, Missed, Beampos];
170
171         pageDuration_in_sec =...
172             atrial.pageDuration(iPage)*Cfg.Screen.monitorFlipInterval;
173
174         [x, y, buttons, t0, t1] =...
175             ASF_waitForResponse(Cfg, pageDuration_in_sec - toleranceSec);
176
177         if any(buttons)
178             % ShowCursor
179             %A BUTTON HAS BEEN PRESSED BEFORE TIMEOUT
180             if Cfg.responseTerminatesTrial
```

```
                %ANY CODE THAT YOU FEEL APPROPRIATE FOR SIGNALING THAT
                %PARTICIPANT HAS PRESSED A BUTTON BEFORE THE TRIAL ENDED
                %Snd('Play','Quack')
            else
                %WAIT OUT THE REMAINDER OF THE STIMULUS DURATION WITH
                %MARGIN OF toleranceSec
                %MAKE THIS CONFIGURABLE AND GO HERE IF I ALLOW ONLY ONE
                %RESPONSE? JVS20130329
                wakeupTime = WaitSecs('UntilTime',...
                    StimulusOnsetTime + pageDuration_in_sec - toleranceSec);
            end
            if responseGiven == 0
                responseGiven = 1;
                responseCounter = responseCounter + 1;
                %FIND WHICH BUTTON IT WAS
                this_response.key = find(buttons);
                %COMPUTE RESPONSE TIME
                this_response.RT = (t1 - StartRTMeasurement)*1000;
            end
        end
    end
end
%-----------------------------------------------------------------------
%END OF PHASE 3
%-----------------------------------------------------------------------

%-----------------------------------------------------------------------
% PHASE 4) LOOP THROUGH PAGE PRESENTATIONS WITHOUT RESPONSE COLLECTION
% (AFTER RESPONSE HAS BEEN GIVEN) SAME AS PHASE 2
%-----------------------------------------------------------------------
%OTHER PICS
for iPage = atrial.endRTonPage+1:nPages
    if (iPage > atrial.nPages)
        break;
    else
        %PUT THE APPROPRIATE TEXTURE ON THE BACK BUFFER
        Screen('DrawTexture', windowPtr, Stimuli.tex(atrial.pageNumber(iPage)));
```

```
218
219             %PRESERVE BACK BUFFER IF THIS TEXTURE IS TO BE SHOWN
220             %AGAIN AT THE NEXT FLIP
221             bPreserveBackBuffer = atrial.pageDuration(iPage) > 1;
222
223             %FLIP THE CONTENT OF THIS PAGE TO THE DISPLAY AND PRESERVE IT
224             %IN THE BACKBUFFER IN CASE THE SAME IMAGE IS TO BE FLIPPED
225             %AGAIN TO THE SCREEN
226             [VBLTimestamp StimulusOnsetTime FlipTimestamp Missed Beampos] =...
227                 ASF_xFlip(windowPtr, Stimuli.tex(atrial.pageNumber(iPage)),...
228                 Cfg, bPreserveBackBuffer);
229
230             %SET TRIGGER (PARALLEL PORT AND EYELINK)
231             ASF_setTrigger(Cfg, atrial.pageNumber(iPage));
232
233
234             %LOG WHEN THIS PAGE APPEARED
235             timing(iPage, 1:6) = [atrial.pageDuration(iPage), VBLTimestamp,...
236                 StimulusOnsetTime FlipTimestamp Missed Beampos];
237
238             %WAIT OUT STIMULUS DURATION IN FRAMES.
239             nFlips = atrial.pageDuration(iPage) - 1; %WE ALREADY FLIPPED ONCE
240             for FlipNumber = 1:nFlips
241                 %PRESERVE BACK BUFFER IF THIS TEXTURE IS TO BE SHOWN
242                 %AGAIN AT THE NEXT FLIP
243                 bPreserveBackBuffer = FlipNumber < nFlips;
244
245                 %FLIP THE CONTENT OF THIS PAGE TO THE DISPLAY AND PRESERVE IT
246                 %IN THE BACKBUFFER IN CASE THE SAME IMAGE IS TO BE FLIPPED
247                 %AGAIN TO THE SCREEN
248                 ASF_xFlip(windowPtr, Stimuli.tex(atrial.pageNumber(iPage)),...
249                     Cfg, bPreserveBackBuffer);
250             end
251         end
252     end
253
```

```
%--------------------------------------------------------------------------
%END OF PHASE 4
%--------------------------------------------------------------------------

%--------------------------------------------------------------------------
% PHASE 5) FEEDBACK
%--------------------------------------------------------------------------
%IF YOU WANT TO FORCE A RESPONSE
if Cfg.waitUntilResponseAfterTrial && ~responseGiven
    [x, y, buttons, t0, t1] = ASF_waitForResponse(Cfg, 10);

    if any(buttons)
        %A BUTTON HAS BEEN PRESSED BEFORE TIMEOUT
        responseGiven = 1;  %#ok<NASGU>
        %FINDO OUT WHICH BUTTON IT WAS
        this_response.key = find(buttons);
        %COMPUTE RESPONSE TIME
        this_response.RT = (t1 - StartRTMeasurement)*1000;
    end
end

%TRIAL BY TRIAL FEEDBACK
if Cfg.feedbackTrialCorrect || Cfg.feedbackTrialError
    ASF_trialFeeback(...
        this_response.key(Cfg.feedbackResponseNumber) == atrial.CorrectResponse, Cfg, windowPtr);
end

%--------------------------------------------------------------------------
%END OF PHASE 5
%--------------------------------------------------------------------------


%PACK INFORMATION ABOUT THIS TRIAL INTO STRUCTURE TrialInfo (THE RETURN
%ARGUMENT). PLEASE MAKE SURE THAT TrialInfo CONTAINS THE FIELDS:
%   trial
%   datestr
```

```
290   %   tStart
291   %   Response
292   %   timing
293   %   StartRTMeasurement
294   %   EndRTMeasurement
295   %OTHERWISE DIAGNOSTIC PROCEDURES OR ROUTINES FOR DATA ANALYSIS MAIGHT FAIL
296   TrialInfo.trial = atrial;  %REQUESTED PAGE NUMBERS AND DURATIONS
297   TrialInfo.datestr = strDate; %STORE WHEN THIS HAPPENED
298   TrialInfo.tStart = tStart; %TIME OF TRIAL-START
299   TrialInfo.Response = this_response; %KEY AND RT
300   TrialInfo.timing = timing; %TIMING OF PAGES
301   TrialInfo.StartRTMeasurement = StartRTMeasurement; %TIMESTAMP START RT
302   TrialInfo.EndRTMeasurement = EndRTMeasurement; %TIMESTAMP END RT
```