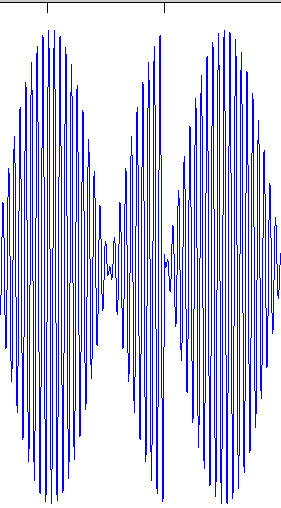
CreateSequence

* Different stimtypes, e.g. 3&4 for tactile
* ~~Different nX for different CP~~
* Block randomisation for ‘cond’; Set randomisation across conditions; ‘divide’ blocking to ensure similar distribution.

Setup for low-latency audio:

* install ASIO4ALL and select this option in Matlab preferences
* Set up: <http://www.lrdc.pitt.edu/maplelab/matlab_audio.html#test1>
* use Psychportaudio option in Settings.
* ~~Create a generic version of TSOT to include auditory stim~~
* ~~Get auditory stims working with Psychtoolbox~~
* ~~Fully test stim sequences and adaptive thresholding~~
* Create CORE fMRI sequences
  + ~~Different blocks with different stim types~~
  + ~~Different h.Settings.f0 for the different stim/block types~~
  + ~~How to set tactile and auditory “volume” separately?~~
  + ~~Audio file intros~~
  + ~~“oddball on oddball” – unique patterns occurring rarely~~
  + ~~Fix psych close error when expt has ended.~~
  + ~~EEG triggers: currently only on half of pitch changes~~
  + ~~Re-write oddball settings as design matrix.~~ 
    - ~~Discrim expt can be done with duration oddball design:~~
      * ~~However, pitch change is part of patternmethod – response to every 2 trials paired.~~
    - ~~Roving oddball:~~
      * ~~Specify min number of stims to include during randomisation process~~
      * ~~Start with just two condition numbers: change v no change, and randomise using 1~~~~st~~ ~~rule (gap of at least nX).~~
      * ~~“If rovingoddball”: update condition numbers to 4 in total~~
      * ~~2~~~~nd~~ ~~Rule during randomisation: total number of each stimulus type should be similar to within N stims.~~
  + For adaptive:
    - ~~Option to run either at end of every trial, or only every trial prior to oddball. Top of AdaptStair.~~
    - ~~In latter case, identify all responses since the trial of last oddball (h.Seq.condnum – even numbered).~~
    - ~~Omit if later than n seconds/trials after change~~
    - ~~Pattern method is flawed if not accounting for phase~~
    - ~~Options:~~
      * ~~Re-write pattern method~~



* + Generify CreateSequence to allow any oddball design
    - Add oddballprob as a conditionmethod in settings and sinwave
  + ~~Labjack block trigger~~
* ~~Create NTIP sequences~~
  + ~~Need for continuous stimulation waveform at high frequency is not compatible with trial design.~~
  + ~~If cannot use trial design, also cannot~~ 
    - ~~Send eeg triggers for oddballs, unless the entrainment and oddball stimuli are separate processes (e.g. different modalities).~~
      * ~~Solution: Can query audioplayer while it is playing with “~~**~~cs=get(pl,'CurrentSample');~~**~~” – would need a separate function that can continuously monitor for a particular sample and then send EEG trigger.~~
    - implement adaptive changes in stimulus once an audioplayer object has started.
      * Solution: modify audio object, get currentsample and start playing new object at currentsample. Might be some delay between getting currentsample and playing?
      * Responsive to participant response (e.g. adaptive threshold) but does not need to be part of a trial design.
  + if using timer from labjack for tactile stimuli, adaptive changes in stimulus intensity are possible (intensity is controlled separately from the timer) and EEG triggers are possible (as these are responsive to intensity change) but syncing to the phase of stimulation is more challenging. Phase syncing is not possible for tactile stimuli anyway because there are no between-stimulus changes when the 10Hz stimulus is the carrier stimulus. Still may be possible by querying the duty cycle of the Labjack timer (see link below) – this would be needed anyway to ensure intensity changes (e.g. oddball) are triggered on the exact correct stimulus in the carrier sequence.
* Adaptive:
  + Sinwav.m: Allow calculation of the sinwav for n (e.g. 2) number of future trials (when h.i=1) or for single trials occurring in m trials (when h.i>1). Define n and m in settings to be called by sinwav.
    - h.Seq.stimseq and h.Seq.aud become trial-specific – perhaps store in h.Seq.trial(n) but clear previous data
  + Trials.m: create new function “adaptivetrials” or modify ‘”trials” for compatibility? If using ‘trials’ design, the start of each new trial could get out of sync with the continuous sinwav. Therefore need to using continuous mode to monitor for start of new trial.
    - on every trial, send stimulus m and record EEG etc.
    - at end of ContLoop, if m> 0 send the next trial to stimtrain.m
  + Stimtrain.m:
    - Create the next trial sinwav
    - Splice in with current trial sinwav and create new audio object h.Seq.trial(n).aud
    - Find current sample of aud that is playing, stop play and start new aud play on the next sample. Options:
      * Playblocking – will freeze all code
      * Play interruption – no need to stop first?
      * play(playerObj,[start,stop])
      * Play without blocking. The audio can overlap.
        + play(trial1&2,[1,stop])
        + start trial 2
        + cs=CurrentSample
        + play(trial2&3,[cs,stop])
      * Latency in “play” function is up to 70ms!
    - Try low-latency PsychPortAudio: allows early filling the buffer and then use of start cue. Can also change volume.
      * Current latency of 10ms – but seems to be consistent at least.
* Multi-modality testing

**Getting sample periodically:**

first make an new m file and save it as UpdateWin.m and write the following in this

**function UpdateWin(pl)**  
**%global pl;**  
**c=get(pl,'CurrentSample');**  
**t=get(pl,'TotalSamples');**  
**disp(c/t);**

Now make a function f where you can pass this pl.

**f=@() UpdateWin(pl);**

Now set this f as timer function of pl.

**set(pl,'TimerFcn','f()');**

Timercall time is automatically set as TimerPeriod: 0.0500 in seconds  
  
  
Whole piece of code will look like as

**cfile='aa.mp3';**  
**extn=cfile[end-2:end];**  
**switch (extn)**  
  
**case 'mp3'**  
**[y f]=mp3read(cfile);**  
  
**case  'wav'**  
**[y f]=wavread(cfile);**  
**end**  
**pl=audioplayer(y,f);**  
  
**f=@() UpdateWin(pl);**  
**set(pl,'TimerFcn','f()');**

set(player, 'TimerPeriod', 0.001); % MAYBE SOME JITTER??

**play(pl);**

Querying duty cycle in a timer in Labjack: <https://labjack.com/support/datasheets/u3/high-level-driver/example-pseudocode/timers-counters>