QEXfiles: Baseband Quadrature Multi-Band Modulator

Braddon Van Slyke, ACØZJ

13111 W. 60th Ave., Arvada CO 80004

email: **blindpigsaloon@q.com**

Baseband FM Modulation using Octave. See also **basebandIQ.m**

```
% Filename: basebandIQ.m
%
% Purpose: Generates FM or SSB IQ baseband data for use with the LTC5598
%          Quadrature Modulator Board.  The modulating source is a
%          previously recorded wav file, but it could be generated in
%          Octave/Matlab just as easily.
%
% B. Van Slyke AC0ZJ  1/3/2018
%
% Note - Octave users - The hilbert function requires you run
% 'pkg load signal' from the command line prior to running this script

clear
more off  % allows displaying variables during run


% load the modulating signal
[yfile, fs] = audioread('c:\temp\morseAudio.wav');


% data from audioread is in colums, one row.  Switch to one column for
% ease of use
y = yfile';


% need to subtract off any dc from input (audio) signal
y = y - mean(y);


% Normalize the amplitude of y such that it's same as a sine wave with
% peak-to-peak amplitude of 1.  WARNING - the audio contains portions of
% silence, so you can't use std - the more silence, the more it biases
% lower. Assuming a simple tone, just use the max of the signal to scale.
y = y  / max(y);

% Add zeros (silence) at the beginning of audio to allow 2m handheld
% squelch time.  Add to end as well to prevent abrupt ending.
y = [zeros(1, 4000) y zeros(1, 500)];


% optional test tone
% fs = 8000;
```

```
% t = 0 : 1/fs : 5 - 1/fs;
% y = sin(2*pi*t*1900);  %1.9 kHz 4.5 deviation => null at carrier


modgain = 2.4;


% Upsample the data.  If the baseband modulating frequency is to be 10
% kHz, then the final data rate should be at least 2x greater than that
% (Nyquist)
upsampleRate = 48000;

ys = resample(y, upsampleRate, fs);
fs = upsampleRate;


% Set to 1 (true) for FM modulation, 0 for SSB
fmMod = 0;

if fmMod == 1

   t = 0: 1/fs : (length(ys) - 1) / fs;

   % The baseband carrier frequency, that is, the signal to be modulated.
   % If the LO, for example, is set to 145.0 MHz, then the transmitted
   % signal will be at 145.010 MHz.  Negating Q will move the transmitted
   % signal to 144.090 MHz.  The modulating signal must be band-limited
   % to fc/2.
   fc = 10000;

   % This value was set by observing the RF output while transmitting a
   % tone of 1.9 kHz.  By using the Bessel Null method, a deviation of
   % 4.5 kHz can be set by adjusting the modulation gain so that the
   % carrier is set to a minimum.
   modulationGain = 2.4;

   % Assuming a band limited modulation signal
   % Integrate modulating signal up to current time using cumsum function
   integralSumYs = cumsum(ys);
   s = cos(2 * pi .* t * fc + modulationGain .* integralSumYs);
   H = hilbert(s);

   %  Here's an alternative to using the Hilbert transform
   %  H = cos(2 * pi .* t .* fc + modulationGain .* integralSumYs);
   %  H = H + i * -sin(2 * pi .* t .* fc + modulationGain .* integralSumYs);

else % assume usb

   H = hilbert(ys);
   % for LSB, multiply the Q signal by -1.

end
```

```
% Create array 'a' for output to audioplayer
a(1,:) = imag(H);
a(2,:) = real(H);

% This is sound card specific!!!
% Gain vs voltage out of sound card *speaker port*.  The "line level"
% output isn't high enough for the modulator board, as max 2 Vpp input is
% expected.
%
% gain   Vpp out FM
%  1.0     4.0v
%  0.8     3.2v
%  0.6     2.4v
%  0.4     1.6v
%  0.3     1.25v
%  0.2     0.83v
%  0.1     0.4
gain = 0.3;
a = gain * a;

% Display one channel of output for test/analysis
kk = 22000:23000;
plot(a(1,kk))


% Use audiodevinfo to get audio device info such as name of device
dev = audiodevinfo();
dev.output.ID
dev.output.Name

% *** You'll need to try different numbers to see what number matches to
% your specific sound card.
% Note - it seemed that "playblocking" didn't appear to work immediately
% with a new device after a usb sound device install, but a reboot fixed
% that.

outputDeviceID = 4;     % speakers USB sound device
numberOutputBits = 16;

player = audioplayer(a,fs,numberOutputBits,outputDeviceID);

% If you want to play more than once, use the loop
%for i = 1 : 3
   % don't use 'play' as it will immediately return
   %disp(i)
   playblocking(player);
%end

% If you don't want to regenerate every time, here are save and load
% variable examples...
% save morseCode1 a
% load morseCode1 a
```