```
function [decoded_data, STARTWORD_INDICES,
 ENDWORD_INDICES,CLOCKLOCK_INDICES] = MSKdecode(y, varargin)
%
%args: A, FS, SYMBOL_FREQUENCIES, DATA_RATE MODULATION_T
%

numvarargs = length(varargin);
if numvarargs > 5
    error('too many arguments');
end


optargs = {1, 44100, 551.25, 1600, 551.25 , 1/551.25*40};

optargs(1:numvarargs) = varargin; %set unset args to defaults

[A,FS,DATA_RATE,CENTER_FREQ, clock_ref_freq, clock_ref_duration] =
 optargs{:};

SYMBOL_FREQUENCIES = [CENTER_FREQ - DATA_RATE/4 CENTER_FREQ +
 DATA_RATE/4];
SAMPLING_T = 1/FS;
MODULATION_T = 1/DATA_RATE;
validateattributes(y,{'numeric'},{'row'})

SYNCWORD = [0 1 1 1 1 1 1 0];
SYNCWORD_BITLENGTH = length(SYNCWORD);
CONSECUTIVE_ONES_MAX = 5;
FRAME_PAYLOAD_MAXIMUM_BITLENGTH = 128;
decoded_data = [];
frame_data = [];
PAYLOAD_COUNT = 0;
LAST_WORD = zeros(1,SYNCWORD_BITLENGTH);
consec_last_ones = 0;
state ='WAITING';
bit_idx = 0;

startTime = 0; %start=0 seconds
endTime = SAMPLING_T*length(y); %end=1 seconds
Trange = startTime:SAMPLING_T:endTime-SAMPLING_T;
idx = 1;

% http://www.electronics.dit.ie/staff/amoloney/lecture-26.pdf

r0_Isum = 0;
r0_Qsum = 0;

r1_Isum = 0;
r1_Qsum = 0;

rclock_Isum = 0;
rclock_Qsum = 0;
```

```
rlo_Isum = 0;

phase_acc_zero =0;
phase_acc_one =0;
phase_acc_clock =0;
phase_acc_lo=0;


phase_step_zero = 2*pi/(FS/SYMBOL_FREQUENCIES(1));
phase_step_one = 2*pi/(FS/SYMBOL_FREQUENCIES(2));
phase_step_clock = 2*pi/(FS/clock_ref_freq);
count = 0;

clock_conversion_factor = -(FS/DATA_RATE)/(2*pi);
bits_clocklike =0;
rmax = 0;
rmaxclock=0;
clock_samples = [];
samples_in=[];
      STARTWORD_INDICES = [];
      ENDWORD_INDICES = [];
      CLOCKLOCK_INDICES = [];
for time_idx=1:length(Trange)
      phi_d0_inphase = sin(phase_acc_zero + pi/2);
      phi_d0_quad    = sin(phase_acc_zero);

      phi_d1_inphase = sin(phase_acc_one + pi/2);
      phi_d1_quad    = sin(phase_acc_one);

      phi_clock_inphase = sin(phase_acc_clock + pi/2);
      phi_clock_quad = sin(phase_acc_clock);




      phase_acc_zero = phase_acc_zero + phase_step_zero;
      phase_acc_one  = phase_acc_one + phase_step_one;
      phase_acc_clock = phase_acc_clock + phase_step_clock;


      r0_Isum = r0_Isum + phi_d0_inphase*y(time_idx);
      r0_Qsum = r0_Qsum + phi_d0_quad*y(time_idx);

      r1_Isum = r1_Isum + phi_d1_inphase*y(time_idx);
      r1_Qsum = r1_Qsum + phi_d1_quad*y(time_idx);

      rclock_Isum = rclock_Isum + phi_clock_inphase*y(time_idx);
      rclock_Qsum = rclock_Qsum + phi_clock_quad*y(time_idx);

      clock_samples = [clock_samples phi_clock_inphase +
 phi_clock_quad*j];
      samples_in=[samples_in y(time_idx)];
      if count == FS/DATA_RATE
         r0 = r0_Isum^2 + r0_Qsum^2;
         r1 = (r1_Isum)^2 + (r1_Qsum)^2;
         r0 = r0*1.00;

         rmax=max([max([r0,r1]),rmax]);
         rclock = rclock_Isum^2 + rclock_Qsum^2;
         rmaxclock = max([rclock rmaxclock]);

         in_bit = r1 > r0;


%           fprintf('Rclock is: %f\n',rclock);
%           fprintf('R0 is: %f       %i\n',r0,in_bit);
%           fprintf('R1 is: %f \n\n',r1);

        commit_bit = LAST_WORD(1);
        LAST_WORD(1) = in_bit;
        if strcmp(state,'WAITING')
           if isequal(LAST_WORD, SYNCWORD) && max([r0, r1]) > 1000/64
             state = 'START_PAYLOAD';
             STARTWORD_INDICES=[STARTWORD_INDICES time_idx];

             PAYLOAD_COUNT = 1;
           end
           if rclock > 10
               if bits_clocklike > 20
                 fprintf('Rclock is: %f\n\n',rclock);
                 rclock_Qsum
                 rclock_Isum

                 CLOCKLOCK_INDICES=[CLOCKLOCK_INDICES time_idx];
                 rx_tx_sampling_phase = atan2(rclock_Qsum,rclock_Isum)
                 bits_clocklike = 0;

                 count_adjustment =
 round(rx_tx_sampling_phase*clock_conversion_factor);
                 if count_adjustment < 0
                     count_adjustment = count_adjustment + FS/
DATA_RATE
                 end
                 count = count + count_adjustment;

               else
                    bits_clocklike = bits_clocklike + 1;
               end
           else
               bits_clocklike = 0;
           end



        elseif strcmp(state,'START_PAYLOAD')
```

```
            if isequal(LAST_WORD, SYNCWORD)
                disp('FSM DONE\n');
                ENDWORD_INDICES=[ENDWORD_INDICES time_idx];

                state = 'WAITING';
                bits_clocklike = 0;

            elseif PAYLOAD_COUNT == SYNCWORD_BITLENGTH
                state = 'PAYLOAD_PROCESSING';
                frame_data = [];
                consec_last_ones = 0;
                %frame_data = [commit_bit];
                %disp('GOTO PAYLOAD PROC');
            end
             PAYLOAD_COUNT = PAYLOAD_COUNT + 1;

        elseif strcmp(state,'PAYLOAD_PROCESSING')
             if consec_last_ones < CONSECUTIVE_ONES_MAX
                 frame_data = [frame_data commit_bit];
             end
             if commit_bit == 1
                 consec_last_ones = consec_last_ones + 1;
             else
                 consec_last_ones = 0;
             end
            if LAST_WORD == SYNCWORD
            state = 'START_PAYLOAD';
            PAYLOAD_COUNT = 0;
            decoded_data = [decoded_data frame_data];
            end

             PAYLOAD_COUNT = PAYLOAD_COUNT + 1;

        else
           disp('WE DONE GOOFED\n');
           break;
        end

        LAST_WORD = circshift(LAST_WORD',1)';

         bit_idx = bit_idx + 1;
         r0_Isum = 0;
         r0_Qsum = 0;
         r1_Isum = 0;
         r1_Qsum = 0;
         rclock_Isum=0;
         rclock_Qsum=0;
         clock_samples =[];
         samples_in = [];
         count = count - FS/DATA_RATE;

      end
      count = count + 1;
end %loop
```

```
end
```

Published with MATLAB® R2015b