LINE CODING

% Demo of using different line codings

bits = [1 0 1 0 0 0 1 1 0];

bitrate = 1; % bits per second

%Function of Unipolar Non – Return to Zero (UNRZ)

figure;

[t,s] = unrz(bits,bitrate);

plot(t,s,'LineWidth',3);

axis([0 t(end) -0.1 1.1])

grid on;

title(['Unipolar NRZ: [' num2str(bits) ']']);

%Function of Unipolar Return to Zero (URZ)

figure;

[t,s] = urz(bits,bitrate);

plot(t,s,'LineWidth',3);

axis([0 t(end) -0.1 1.1])

grid on;

title(['Unipolar RZ: [' num2str(bits) ']']);

%Function of Polar Return to Zero (PRZ)

figure;

[t,s] = prz(bits,bitrate);

plot(t,s,'LineWidth',3);

axis([0 t(end) -1.1 1.1])

grid on;

title(['Polar RZ: [' num2str(bits) ']']);

%Function of Manchester coding

figure;

[t,s] = manchester(bits,bitrate);

plot(t,s,'LineWidth',3);

axis([0 t(end) -1.1 1.1])

grid on;

title(['Manchester: [' num2str(bits) ']']);

**Unipolar Non- Return to Zero (UNRZ)**

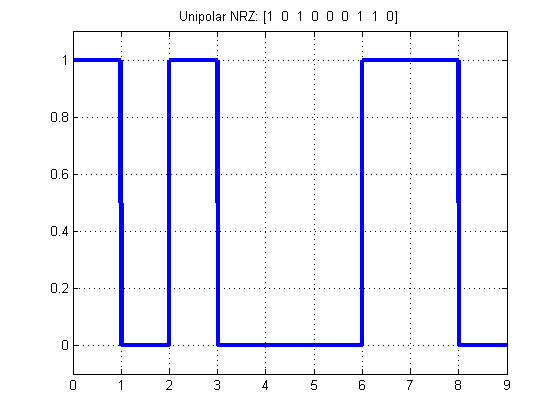
function [t,x] = unrz(bits, bitrate)

% UNRZ Encode bit string using unipolar NRZ code.

% [T, X] = UNRZ(BITS, BITRATE) encodes BITS array using unipolar NRZ

% code with given BITRATE. Outputs are time T and encoded signal

% values X.

T = length(bits)/bitrate; % full time of bit sequence

n = 200;

N = n\*length(bits);

dt = T/N;

display(dt);

t = 0:dt:T;

x = zeros(1,length(t)); % output signal

for i = 0:length(bits)-1

if bits(i+1) == 1

x(i\*n+1:(i+1)\*n) = 1;

else

x(i\*n+1:(i+1)\*n) = 0;

end

end

**Unipolar Return to Zero (URZ)**

function [t,x] = urz(bits, bitrate)

% URZ Encode bit string using unipolar RZ code.

% [T, X] = URZ(BITS, BITRATE) encodes BITS array using unipolar RZ

% code with given BITRATE. Outputs are time T and encoded signal

% values X.

T = length(bits)/bitrate; % full time of bit sequence

n = 200;

N = n\*length(bits);

dt = T/N;

t = 0:dt:T;

x = zeros(1,length(t)); % output signal

for i = 0:length(bits)-1

if bits(i+1) == 1

x(i\*n+1:(i+0.5)\*n) = 1;

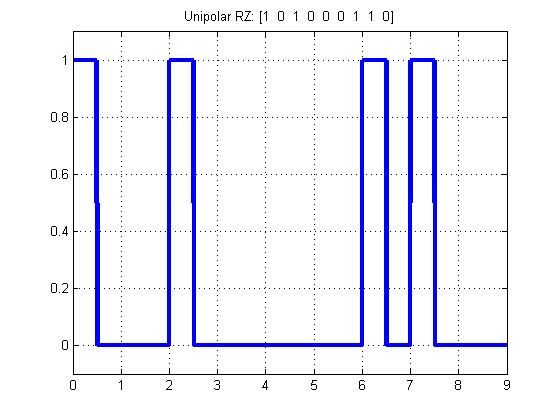
x((i+0.5)\*n+1:(i+1)\*n) = 0;

else

x(i\*n+1:(i+1)\*n) = 0;

end

end



**Polar Return to Zero (URZ)**

function [t,x] = prz(bits, bitrate)

% PRZ Encode bit string using polar RZ code.

% [T, X] = PRZ(BITS, BITRATE) encodes BITS array using polar RZ

% code with given BITRATE. Outputs are time T and encoded signal

% values X..

T = length(bits)/bitrate; % full time of bit sequence

n = 200;

N = n\*length(bits);

dt = T/N;

t = 0:dt:T;

x = zeros(1,length(t)); % output signal

for i = 0:length(bits)-1

if bits(i+1) == 1

x(i\*n+1:(i+0.5)\*n) = 1;

x((i+0.5)\*n+1:(i+1)\*n) = 0;

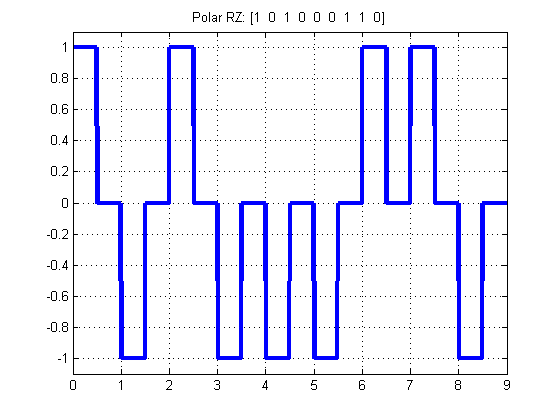
else

x(i\*n+1:(i+0.5)\*n) = -1;

x((i+0.5)\*n+1:(i+1)\*n) = 0;

end

end



**Polar Return to Zero (URZ)**

function [t,x] = manchester(bits, bitrate)

% MANCHESTER Encode bit string using Manchester code.

% [T, X] = MANCHESTER(BITS, BITRATE) encodes BITS array using Manchester

% code with given BITRATE. Outputs are time T and encoded signal

% values X.

T = length(bits)/bitrate; % full time of bit sequence

n = 200;

N = n\*length(bits);

dt = T/N;

t = 0:dt:T;

x = zeros(1,length(t)); % output signal

for i = 0:length(bits)-1

if bits(i+1) == 1

x(i\*n+1:(i+0.5)\*n) = 1;

x((i+0.5)\*n+1:(i+1)\*n) = -1;

else

x(i\*n+1:(i+0.5)\*n) = -1;

x((i+0.5)\*n+1:(i+1)\*n) = 1;

end

end

