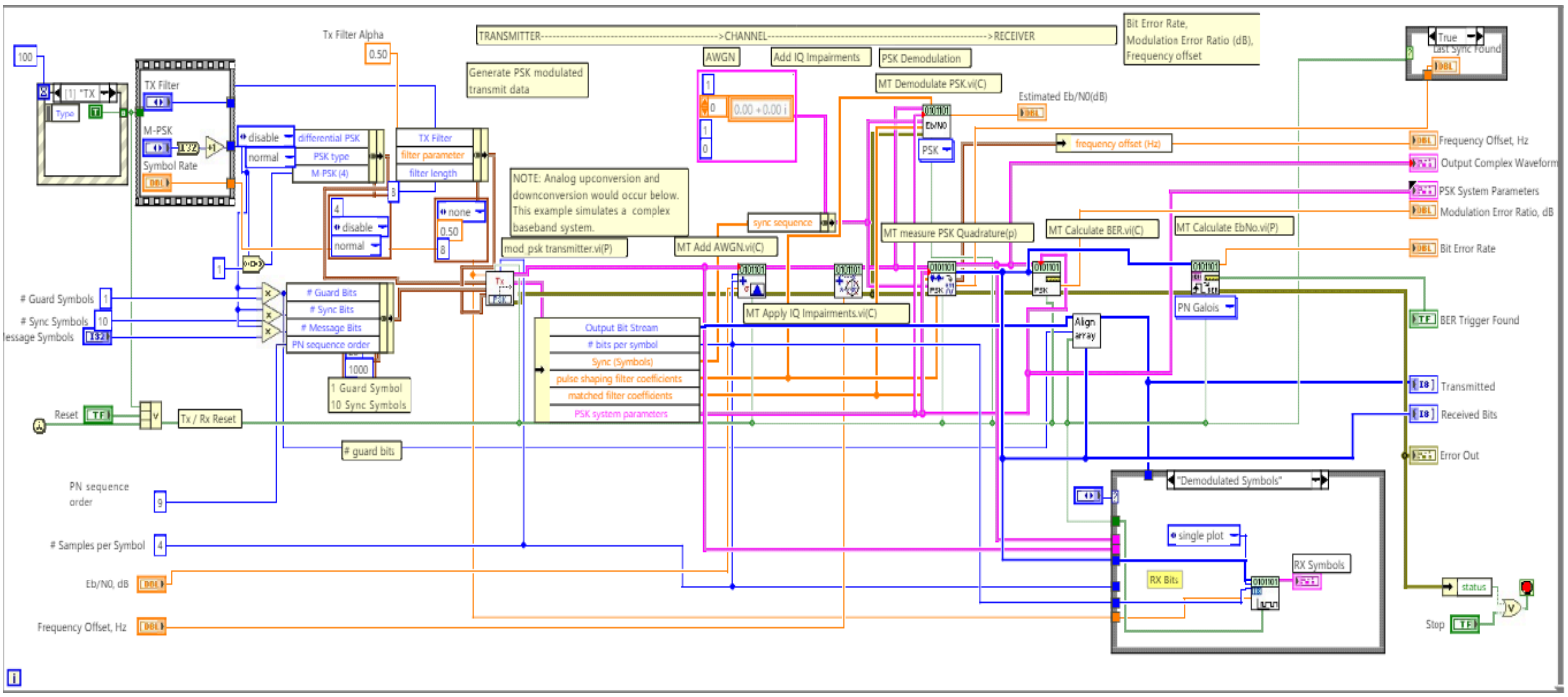
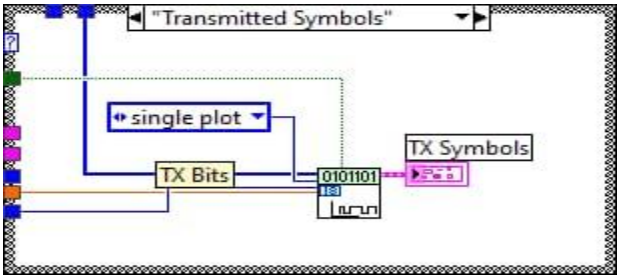
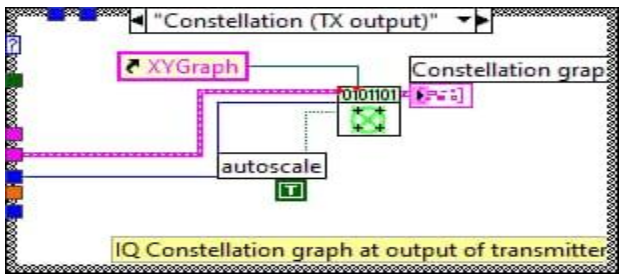
EXP-1

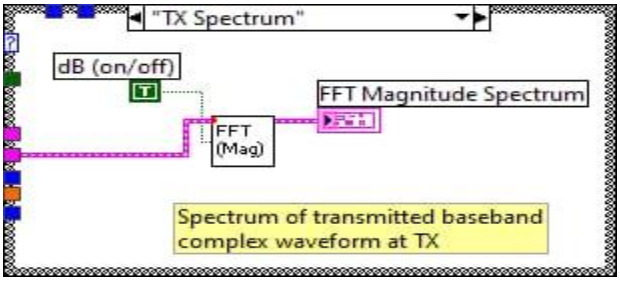
M-PSK transceiver Design

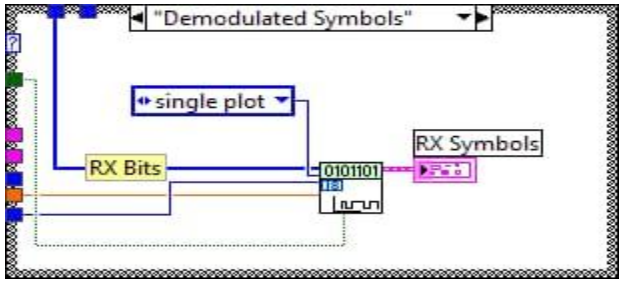


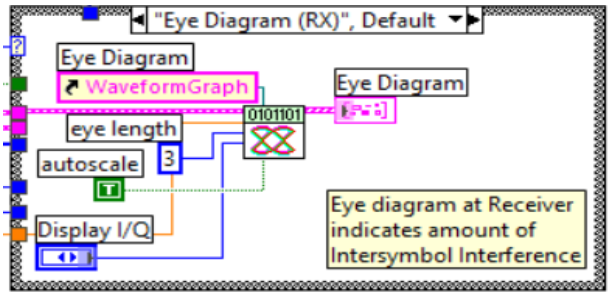
1. **mod_psk transmitter.vi:** Not available in Search Path: National Instruments/Labview2020/vi.lib/addons/Modulation/Digital/Support/Examples
2. **MT Add AWGN.vi:** Available in search or Quick drop (ctrl + Spacebar) Path: National Instruments/Labview2020/vi.lib/addons/Modulation/Digital/Impairments
3. **MT Apply IQ Impairments.vi:** Available in search or Quick drop (ctrl + Spacebar) Path: National Instruments/Labview2020/vi.lib/addons/Modulation/Digital/Impairments
4. **MT Demodulate PSK.vi:** Available in search or Quick drop (ctrl + Spacebar) Path: National Instruments/Labview2020/vi.lib/addons/Modulation/Digital/Demodulation
5. **MT measure PSK Quadrature Impairments.vi:** Not available in Search Path: National Instruments/Labview2020/vi.lib/addons/Modulation/Digital/Demodulation
6. **MT Calculate BER.vi:** Available in search or Quick drop (ctrl + Spacebar) Path: National Instruments/Labview2020/vi.lib/addons/Modulation/Digital/Demodulation
7. **MT Calculate EbNo.vi:** Not available in Search Path: National Instruments/Labview2020/vi.lib/addons/Modulation/Digital/Demodulation
8. **Mod_align Tx and Rx bit sequence:** Not available in Search Path: National Instruments/Labview2020/vi.lib/addons/Modulation/Digital/support/examples

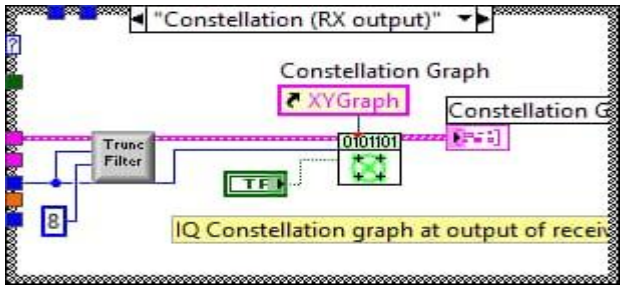






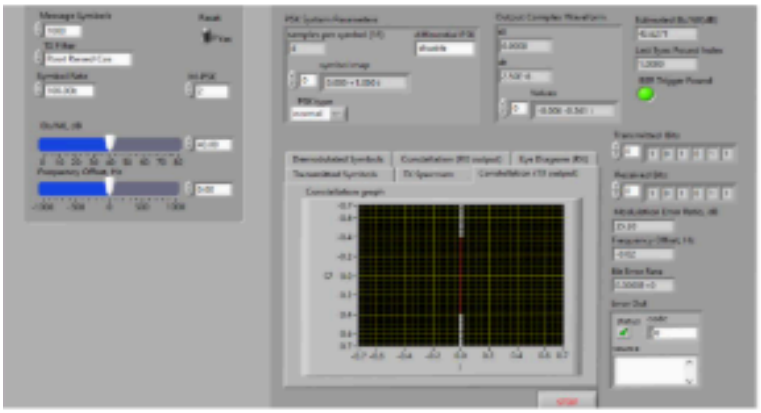




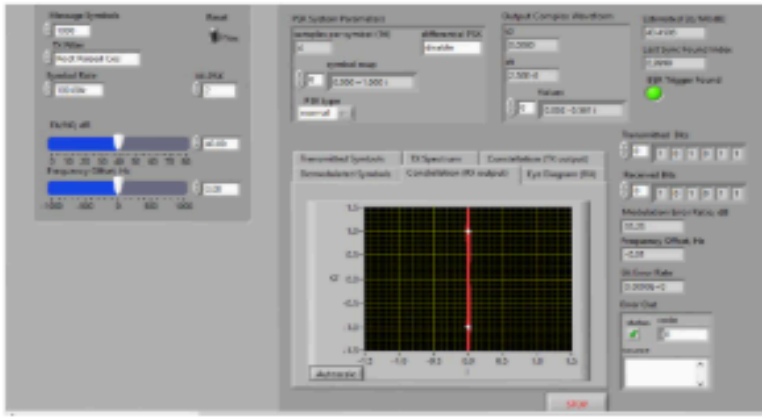


Output:

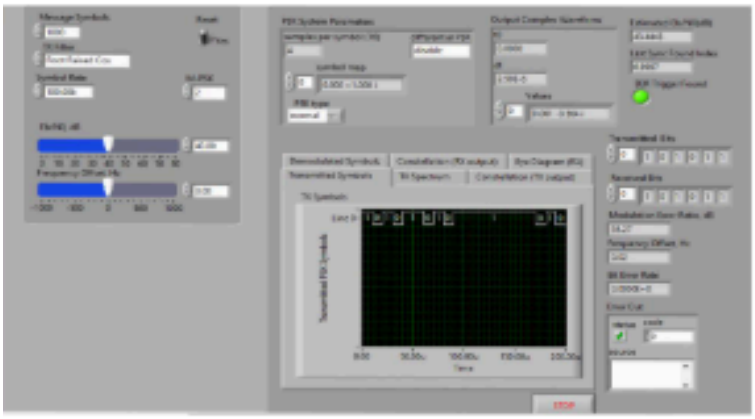
2-PSK



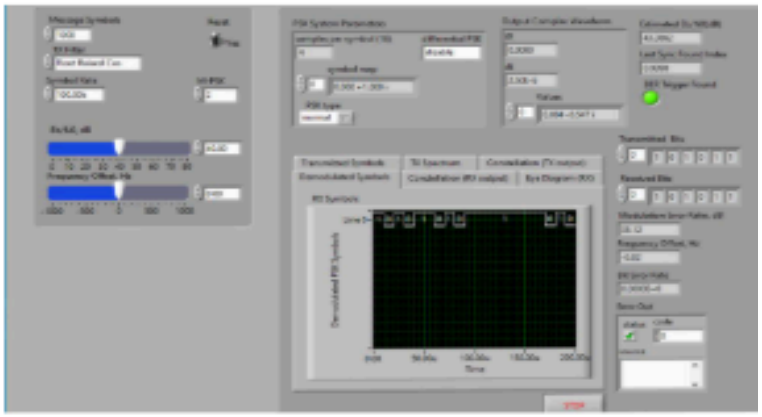
Constellation (TX Output)



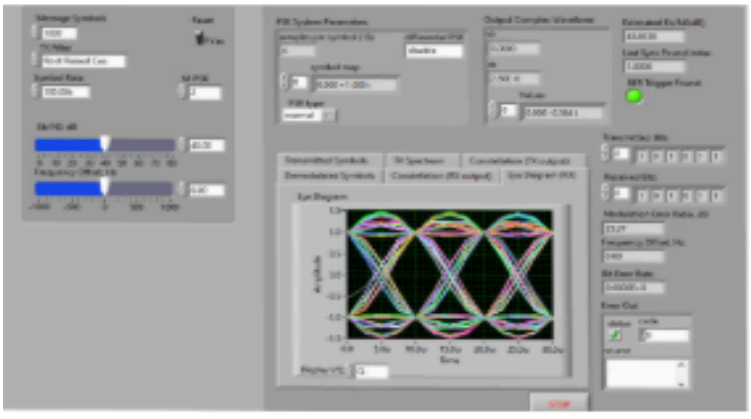
Constellation (RX Output)



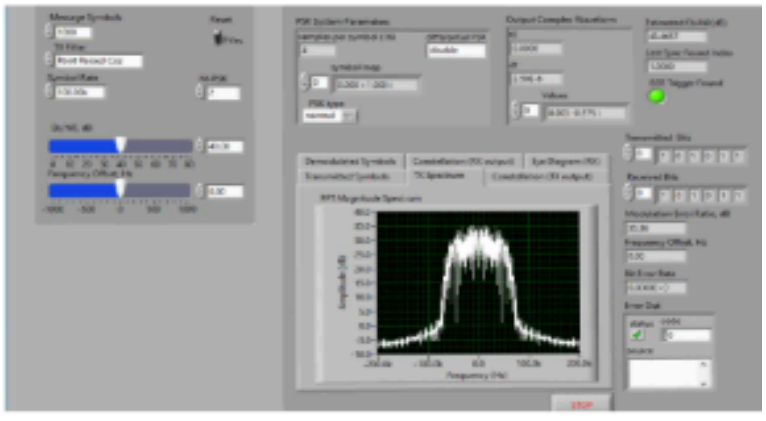
Transmitted symbols



Demodulated symbols



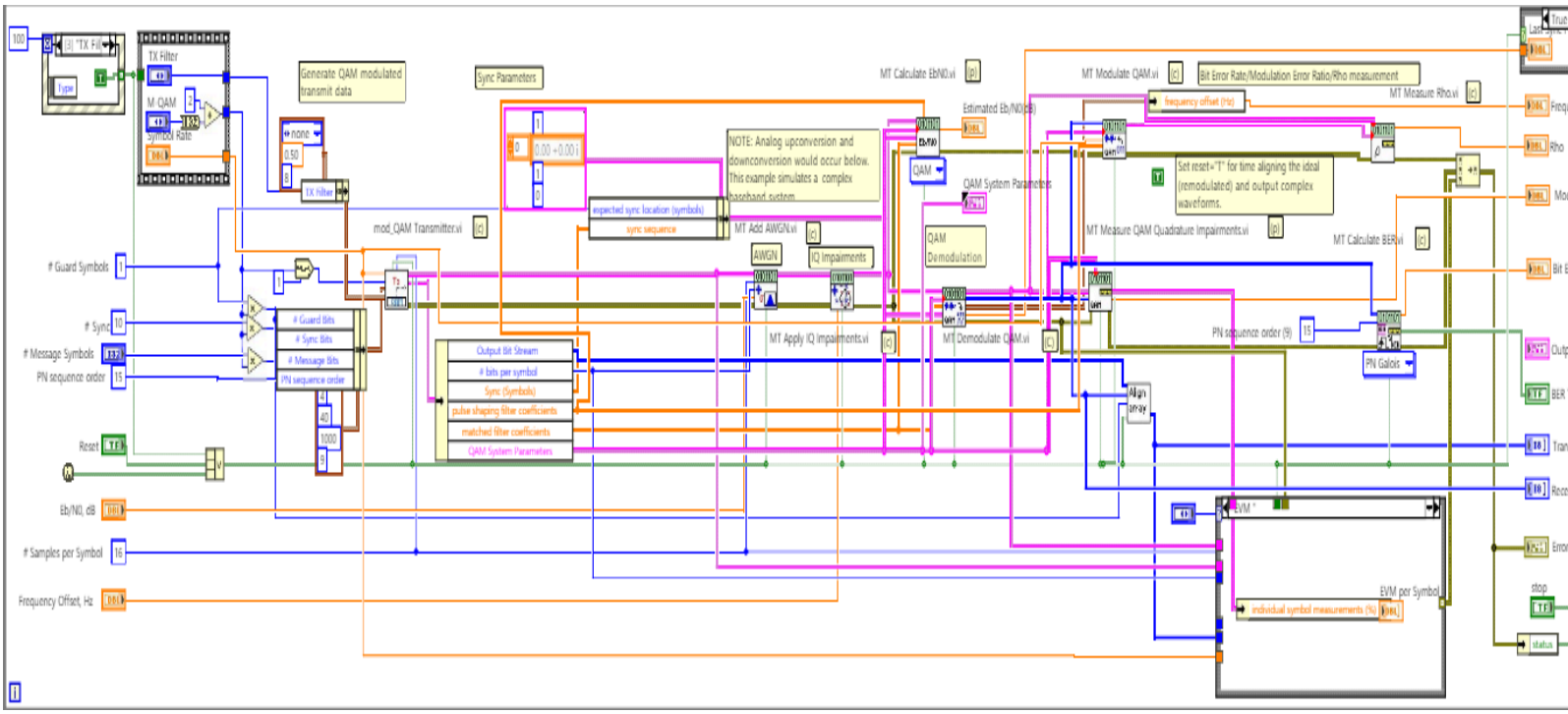
Eye diagram



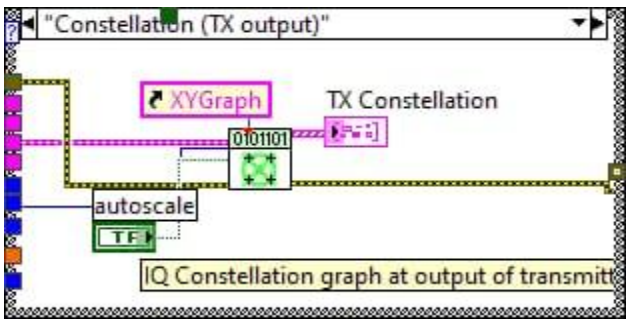
TX Spectrum

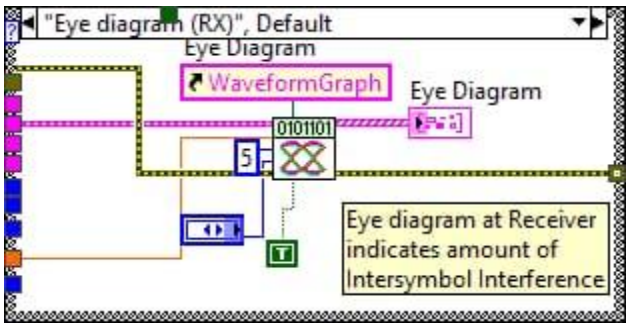
EXP-2

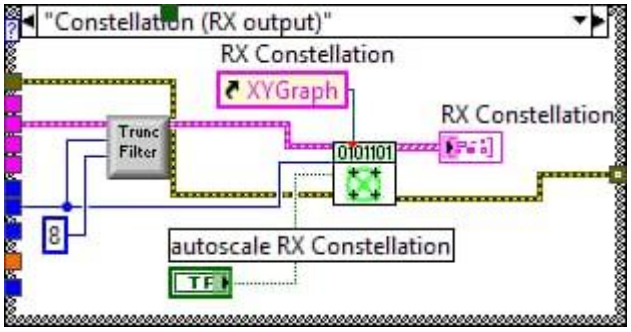
M-QAM Transceiver design

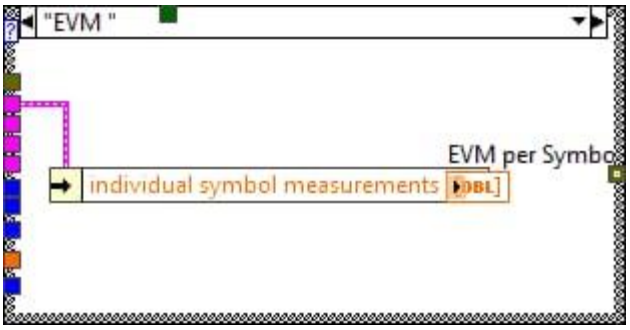


1. **mod_QAM transmitter.vi:** Available in search or Quick drop (ctrl + Spacebar) Path: National Instruments/Labview2020/vi.lib/addons/Modulation/Digital/Support/Examples
2. **MT Add AWGN.vi**: Available in search or Quick drop (ctrl + Spacebar) Path: National Instruments/Labview2020/vi.lib/addons/Modulation/Digital/Impairments
3. **MT Apply IQ Impairments.vi:** Available in search or Quick drop (ctrl + Spacebar) Path: National Instruments/Labview2020/vi.lib/addons/Modulation/Digital/Impairments
4. **MT Demodulate QAM.vi:** Available in search or Quick drop (ctrl + Spacebar) Path: National Instruments/Labview2020/vi.lib/addons/Modulation/Digital/Demodulation
5. **MT measure QAM Quadrature Impairments.vi:** Not available in Search Path: National Instruments/Labview2020/vi.lib/addons/Modulation/Digital/Demodulation
6. **MT Calculate BER.vi:** Available in search or Quick drop (ctrl + Spacebar) Path: National Instruments/Labview2020/vi.lib/addons/Modulation/Digital/Demodulation
7. **MT Calculate EbNo.vi:** Not available in Search Path: National Instruments/Labview2020/vi.lib/addons/Modulation/Digital/Demodulation
8. **MT Modulate QAM:** Available in search or Quick drop (ctrl + Spacebar) Path: National Instruments/Labview2020/vi.lib/addons/Modulation/Digital/Modulation
9. **MT Measure Rho:** Available in search or Quick drop (ctrl + Spacebar) Path: National Instruments/Labview2020/vi.lib/addons/Modulation/Digital/Demodulation

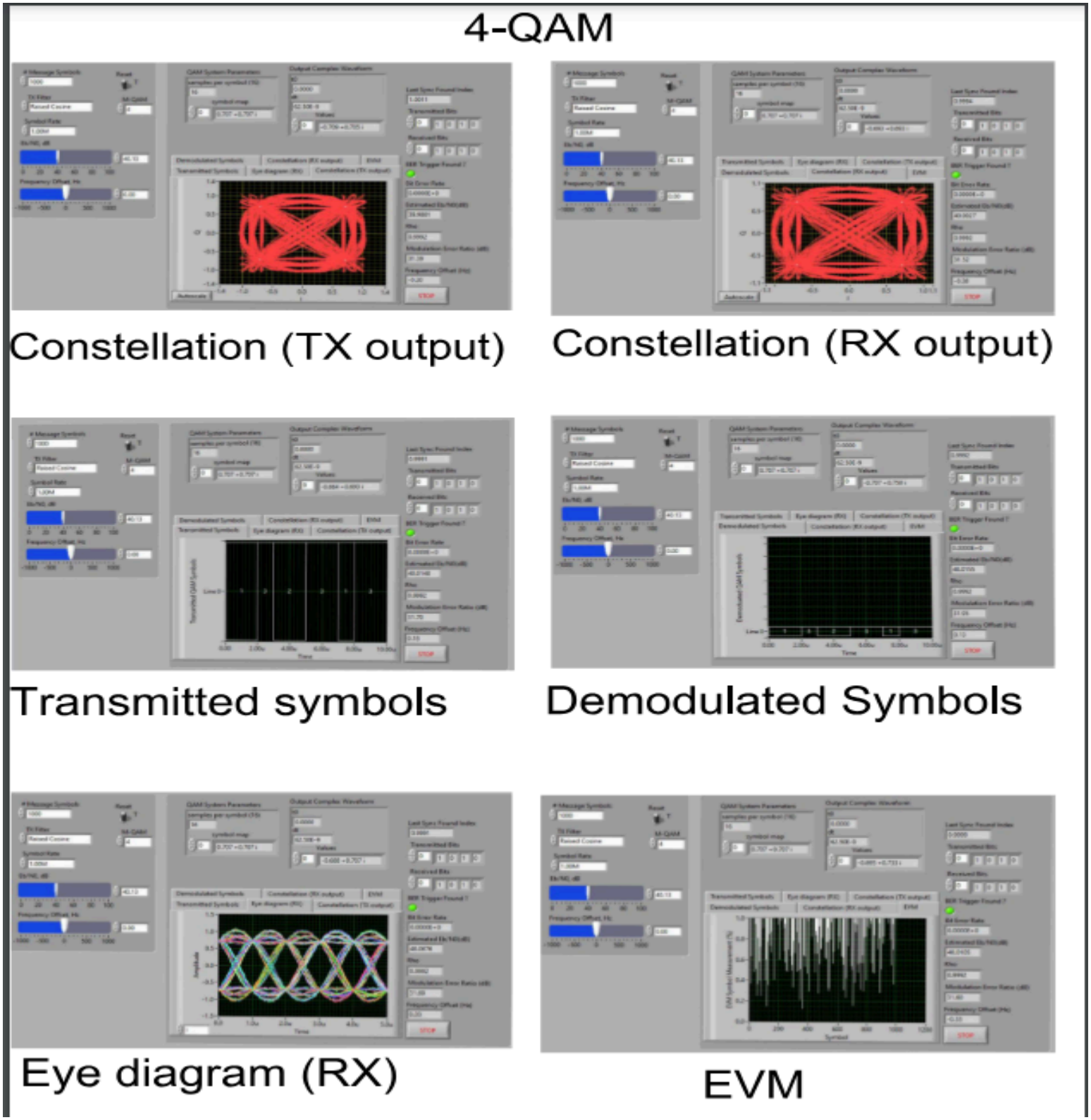





"Constellation (RX output)"
RX Constellation
XYGraph
RX Constellation
Trunc
Filter
0101101
8
autoscale RX Constellation
TF

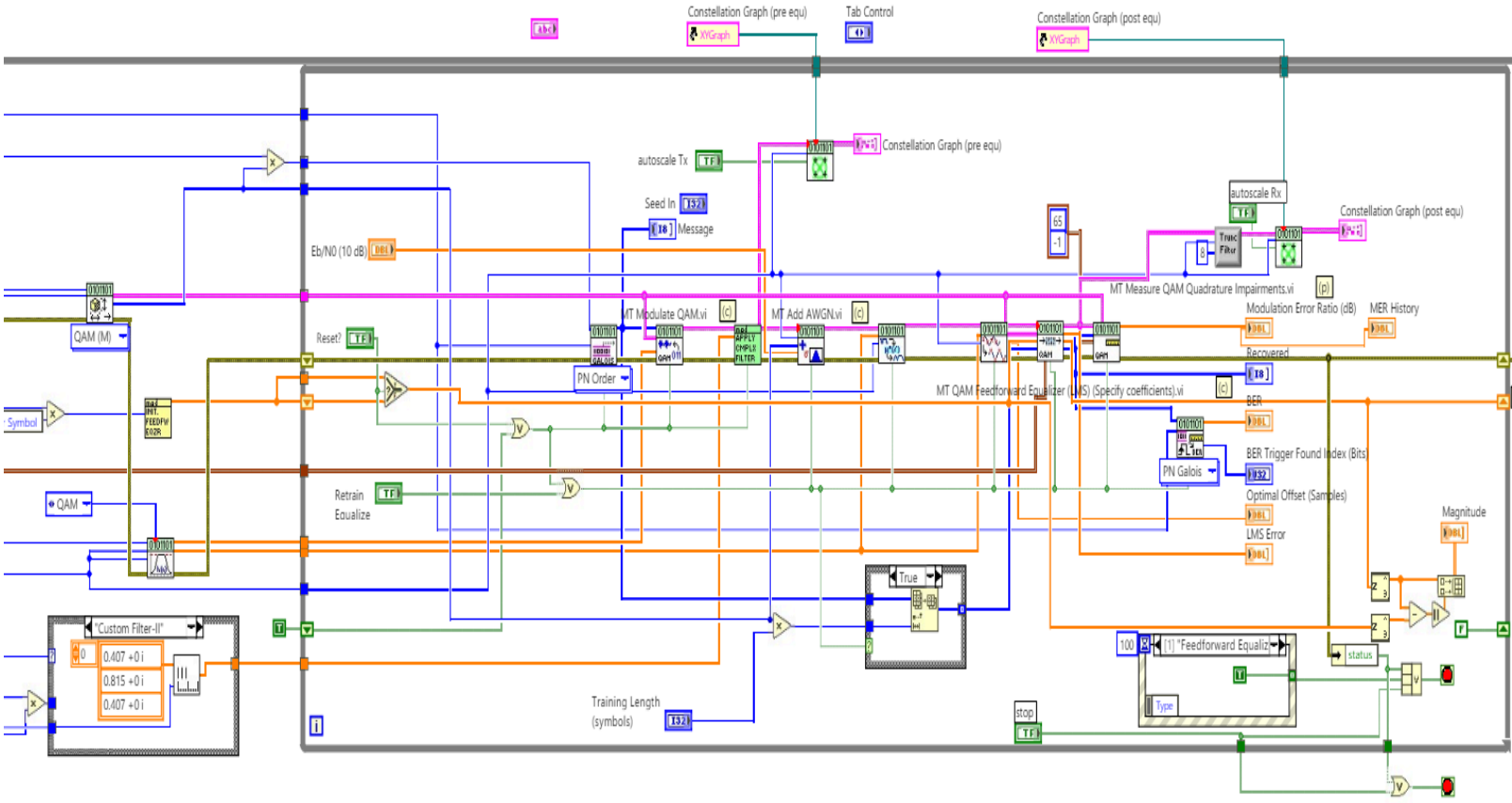

"EVM "
EVM per Symbo
individual symbol measurements
DBL

Output:


4-QAM
Constellation (TX output)
Constellation (RX output)
Transmitted symbols
Demodulated Symbols
Eye diagram (RX)
EVM

EXP-4

Equalization with QAM modulation



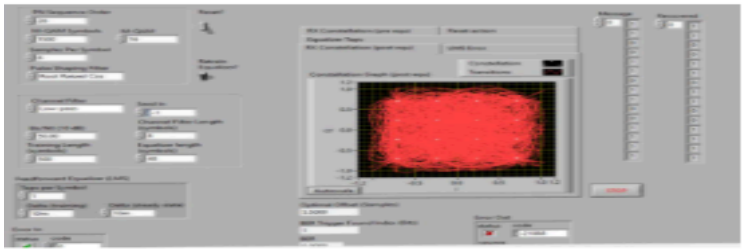
1. **MT Modulate QAM:** Available in search or Quick drop (ctrl + Spacebar) Path: National Instruments/Labview2020/vi.lib/addons/Modulation/Digital/Modulation
2. **MT Add AWGN.vi**: Available in search or Quick drop (ctrl + Spacebar) Path: National Instruments/Labview2020/vi.lib/addons/Modulation/Digital/Impairments
3. **MT QAM Feedforward Equalizer (LMS) (Specify coefficients).vi**:Available in search or Quick drop (ctrl + Spacebar) Path
4. **MT measure QAM Quadrature Impairments.vi**: Not available in Search Path: National Instruments/Labview2020/vi.lib/addons/Modulation/Digital/Demodulation

Output:

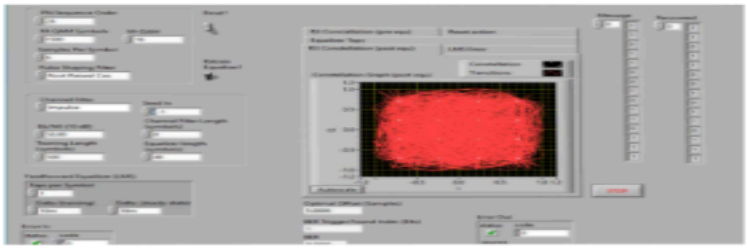
Root Raised Cosine

16-QAM

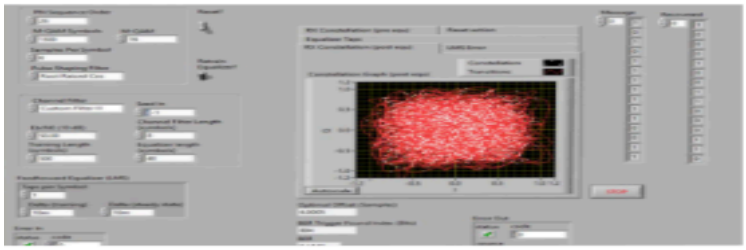
Channel Filter:-



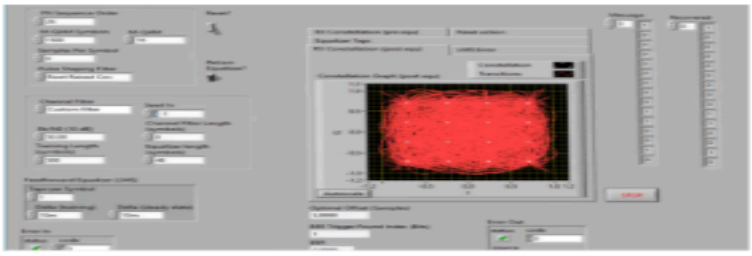
Low-pass



Impulse



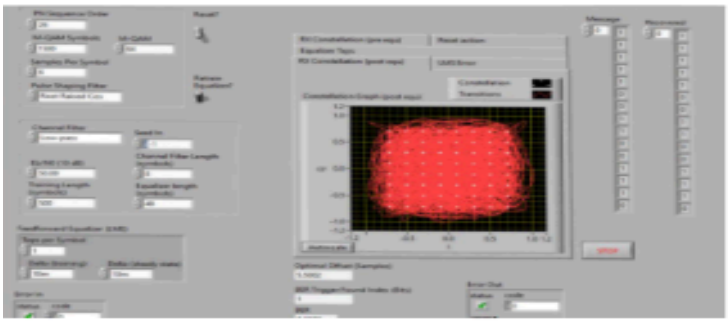
Custom Filter –1



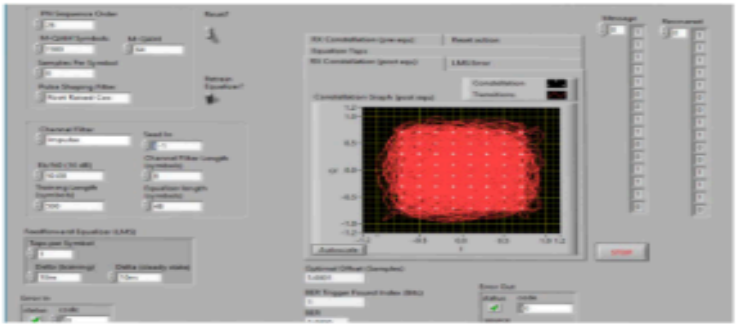
Custom Filter –2

64-QAM

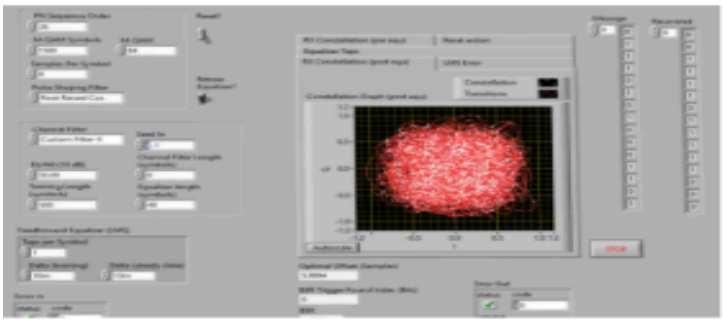
Channel Filter:-



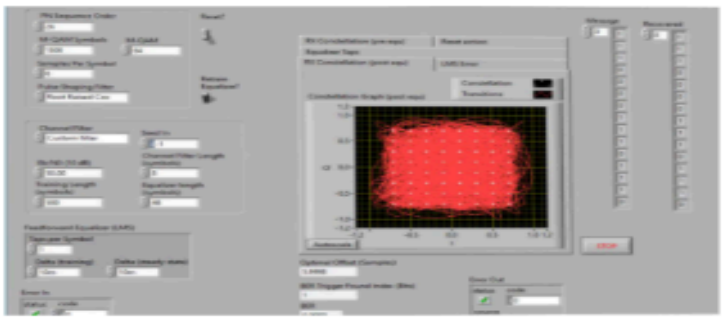
Low-pass



Impulse



Custom Filter-1



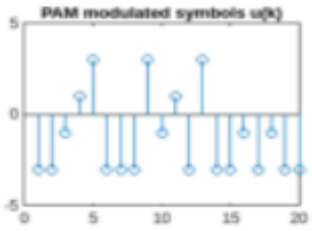
Custom Filter-2

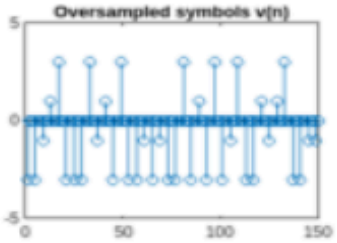
EXP-3

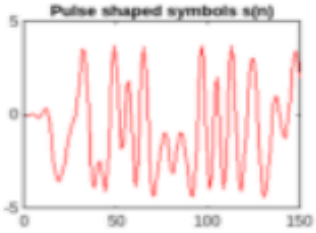
ISI With PAM modulation

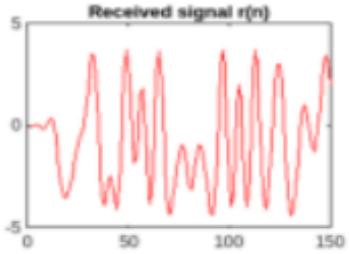
```
clear all;
clc;
N = 10^5;
MOD_TYPE = 'PAM';
M = 4;
d = ceil(M.*rand(1,N));
u = modulate(MOD_TYPE,M,d);
figure;
stem(real(u));
title('PAM modulated symbols u(k)');
xlim([0 20])
ylim([-5 5])

L=4;
v=[u;zeros(L-1,length(u))];
v=v(:).';
figure;stem(real(v));
title('Oversampled symbols v(n)');
xlim([0 150])
ylim([-5 5])

beta = 0.3;
Nsym=8;
L=4;
[p,t,filtDelay] = srrcFunction(beta,L,Nsym);
s=conv(v,p,'full');
figure; plot(real(s),'r');
title('Pulse shaped symbols s(n)');
xlim([0 150])
ylim([-5 5])

EbN0dB = 1000;
snr = 10*log10(log2(M))+EbN0dB;
r = add_awgn_noise(s,snr,L);
figure;
plot(real(r),'r');
title('Received signal r(n)');
xlim([0 150])
ylim([-5 5])

vCap=conv(r,p,'full');
figure; plot(real(vCap),'r');
title('After matched filtering $\hat{v}$(n)','Interpreter','Latex');
xlim([0 150])
ylim([-20 20])

uCap = vCap(2*filtDelay+1:L:end-(2*filtDelay))/L;
figure;
stem(real(uCap));
hold on;
title('After symbol rate sampler $\hat{u}$(n)',...
```

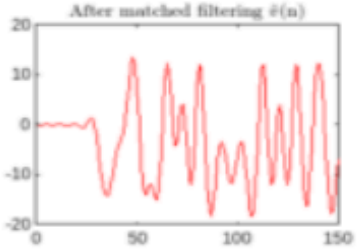
```
'Interpreter','Latex');
dCap = demodulate(MOD_TYPE,M,uCap);
xlim([0 20])
ylim([-5 5])

figure;
plotEyeDiagram(vCap,L,3*L,2*filtDelay,100);
xlim([0 3])
ylim([-15 15])
```

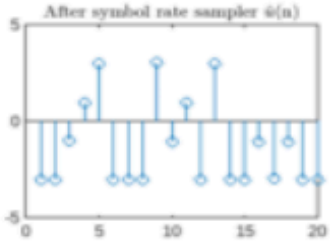


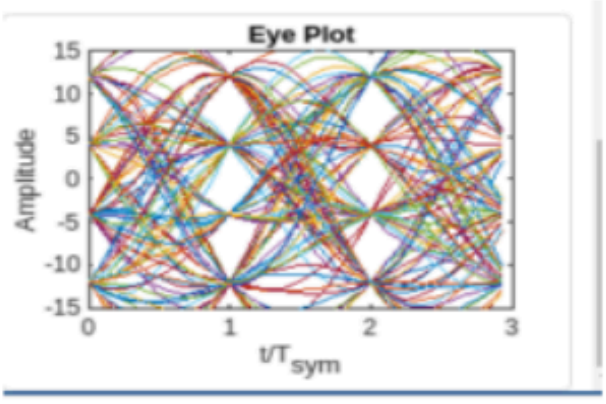


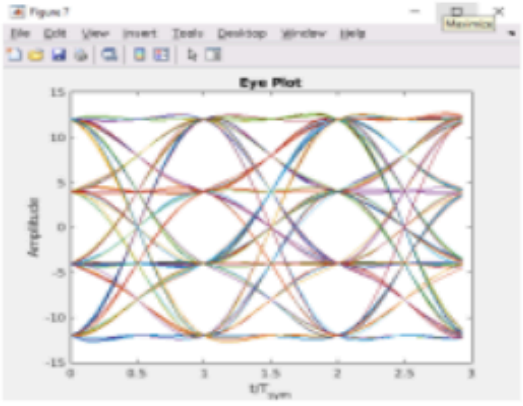












Exp-5

DSSS AND FHSS

DSSS

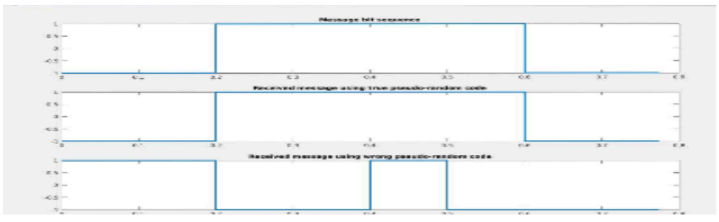
```
clear all;
%% parameters
Fs = 1000;
fc = 100;
fp = 4;
bit_t = 0.1;

%% message generation with BPSK
m = [0 0 1 1 1 1 0 0];
for bit = 1:length(m)
 if(m(bit)==0)
   m(bit) = -1;
 end
end
message = repmat(m,fp,1);
message = reshape(message,1,[]);

%% PN generation and multiply with message
pn_code = randi([0,1],1,length(m)*fp);
for bit = 1:length(pn_code)
 if(pn_code(bit)==0)
   pn_code(bit) = -1;
 end
end
DSSS = message.*pn_code;

%% create carrier and multipy with encoded sequence
t = 0:1/Fs:(bit_t-1/Fs);
s0 = -1*cos(2*pi*fc*t);
s1 = cos(2*pi*fc*t);
carrier = [];
BPSK = [];
for i = 1:length(DSSS)
   if (DSSS(i) == 1)
       BPSK = [BPSK s1];
    elseif (DSSS(i) == -1)
       BPSK = [BPSK s0];
     end
     carrier = [carrier s1];
end

%% demodulation
rx =[];
for i = 1:length(pn_code)
   if(pn_code(i)==1)
       rx = [rx BPSK((((i-1)*length(t))+1):i*length(t))];
   else
       rx = [rx (-1)*BPSK((((i-1)*length(t))+1):i*length(t))];
   end
end
```

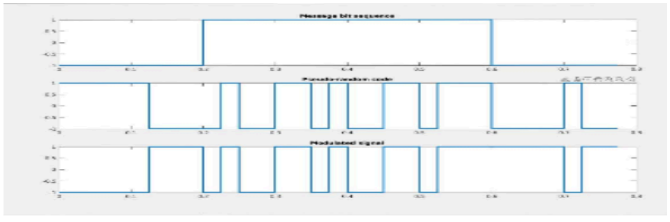
```
demod = rx.*carrier;
result = [];
for i = 1:length(m)
   x = length(t)*fp;
   cx = sum(carrier(((i-1)*x)+1:i*x).*demod(((i-1)*x)+1:i*x));
   if(cx>0)
      result = [result 1];
   else
      result = [result -1];
   end
end

pn_codeWrong = randi([0,1],1,length(m)*fp);
resultWrong = [];
rx2 =[];
for i = 1:length(pn_code)
   if(pn_codeWrong(i)==1)
       rx2 = [rx2 BPSK((((i-1)*length(t))+1):i*length(t))];
   else rx2 = [rx2 (-1)*BPSK((((i-1)*length(t))+1):i*length(t))];
   end
end

demod2 = rx2.*carrier;
for i = 1:length(m)
   x = length(t)*fp;
   cx = sum(carrier(((i-1)*x)+1:i*x).*demod2(((i-1)*x)+1:i*x));
   if(cx>0)
       resultWrong = [resultWrong 1];
   else
       resultWrong = [resultWrong -1];
   end
end

message1 = repmat(result,fp,1);
message1 = reshape(message1,1,[]);
message2 = repmat(resultWrong,fp,1);
message2 = reshape(message2,1,[]);

%% Draw original message, PN code , encoded sequence on time domain
pn_size = length(pn_code);
tpn = linspace(0,length(m)*bit_t-bit_t/fp,pn_size);
tm = 0:bit_t/fp:length(m)*bit_t-bit_t/fp;
figure
subplot(311)
stairs(tm,message,'linewidth',2)
title('Message bit sequence')
axis([0 length(m)*bit_t -1 1]);
subplot(312)
stairs(tpn,pn_code,'linewidth',2)
title('Pseudo-random code');
axis([0 length(m)*bit_t -1 1]);
subplot(313)
```

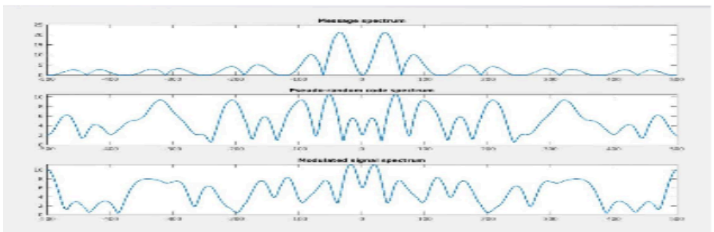
```
stairs(tpn,DSSS,'linewidth',2)
title('Modulated signal');
axis([0 length(m)*bit_t -1 1]);
figure
subplot(311)
stairs(tm,message,'linewidth',2)
title('Message bit sequence')
axis([0 length(m)*bit_t -1 1]);
subplot(312)
stairs(tm,message1,'linewidth',2)
title('Received message using true pseudo-random code')
axis([0 length(m)*bit_t -1 1]);
subplot(313)
stairs(tm,message2,'linewidth',2)
title('Received message using wrong pseudo-random code')
axis([0 length(m)*bit_t -1 1]);
```

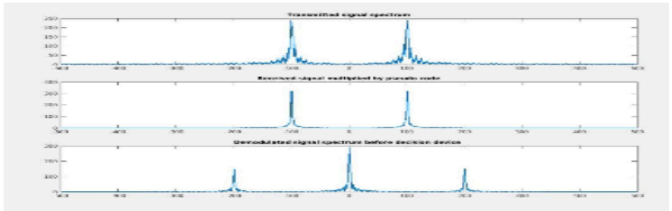
%% Draw original message, PN code , encoded sequence on frequency domain

```
f = linspace(-Fs/2,Fs/2,1024);
figure
subplot(311)
plot(f,abs(fftshift(fft(message,1024))),'linewidth',2);
title('Message spectrum')
subplot(312)
plot(f,abs(fftshift(fft(pn_code,1024))),'linewidth',2);
title('Pseudo-random code spectrum');
subplot(313)
plot(f,abs(fftshift(fft(DSSS,1024))),'linewidth',2);
title('Modulated signal spectrum');
figure;
subplot(311)
plot(f,abs(fftshift(fft(BPSK,1024))),'linewidth',2);
title('Transmitted signal spectrum');
subplot(312)
plot(f,abs(fftshift(fft(rx,1024))),'linewidth',2);
title('Received signal multiplied by pseudo code');
subplot(313)
plot(f,abs(fftshift(fft(demod,1024))),'linewidth',2);
title('Demodulated signal spectrum before decision device ');
```





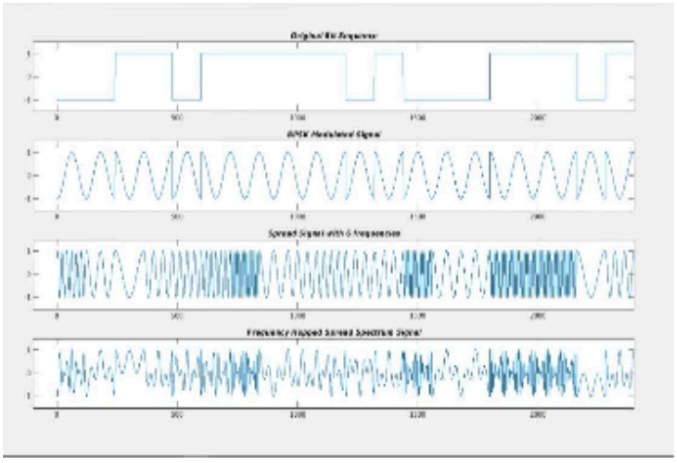


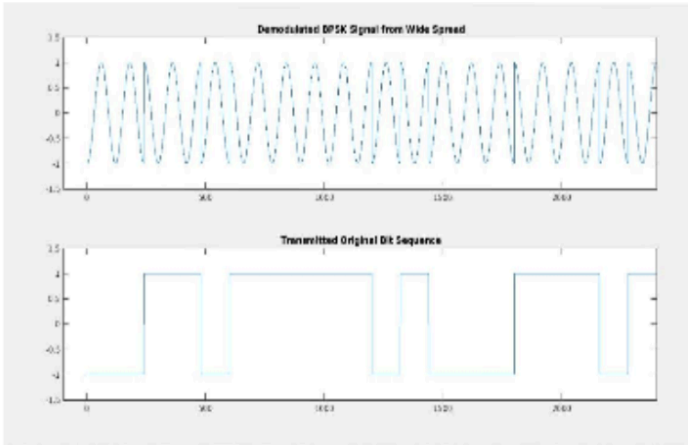


FHSS

```
clc; clear all;
num_bits = 20;
samples_per_bit = 120;
num_carriers = 6;
samples = [10, 20, 30, 40, 60, 120];

disp('Enter your bit sequence as a series of 1s and 0s separated by spaces (e.g., "1 0 1 1 0"):');
manual_input = input('', 's');
bit_sequence = str2num(manual_input);

if length(bit_sequence) ~= num_bits
    error('The number of bits entered must match the defined number of bits (%d).', num_bits);
end

sequence = 2 * bit_sequence - 1;
input_signal = repelem(sequence, samples_per_bit);
t_carrier = linspace(0, 2*pi*num_bits, samples_per_bit*num_bits);
carrier_signal = cos(t_carrier);

figure(1);
subplot(4,1,1);
plot(input_signal);
axis([-100 2400 -1.5 1.5]);
title('\bf\it Original Bit Sequence');

bpsk_mod_signal = input_signal .* carrier_signal;
subplot(4,1,2);
plot(bpsk_mod_signal);
axis([-100 2400 -1.5 1.5]);
title('\bf\it BPSK Modulated Signal');

carriers = cell(1, num_carriers);
for i = 1:num_carriers
     t = linspace(0, 2*pi, samples(i) + 1); t(end) = [];
     carriers{i} = repmat(cos(t), 1, ceil(samples_per_bit / length(t)));
     carriers{i} = carriers{i}(1:samples_per_bit);
end

spread_signal = [];
for i = 1:num_bits
    carrier_idx = randi([1, num_carriers]);
    spread_signal = [spread_signal carriers{carrier_idx}];
end
subplot(4,1,3);
plot(spread_signal);
axis([-100 2400 -1.5 1.5]);
title('\bf\it Spread Signal with 6 frequencies');

freq_hopped_sig = bpsk_mod_signal .* spread_signal;
subplot(4,1,4);
plot(freq_hopped_sig);
```

```
axis([-100 2400 -1.5 1.5]);
title('\bf\it Frequency Hopped Spread Spectrum Signal');

bpsk_demodulated = freq_hopped_sig ./ spread_signal;
figure(2);
subplot(2,1,1);
plot(bpsk_demodulated);
axis([-100 2400 -1.5 1.5]);
title('\bf Demodulated BPSK Signal from Wide Spread');

original_BPSK_signal = bpsk_demodulated ./ carrier_signal;
subplot(2,1,2);
plot(original_BPSK_signal);
axis([-100 2400 -1.5 1.5]);
title('\bf Transmitted Original Bit Sequence');
```



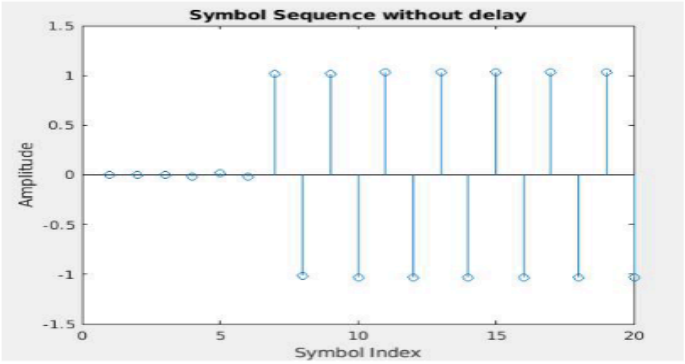


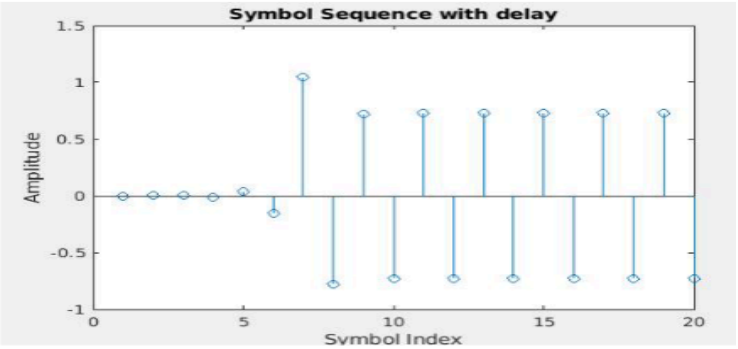
EXP-6

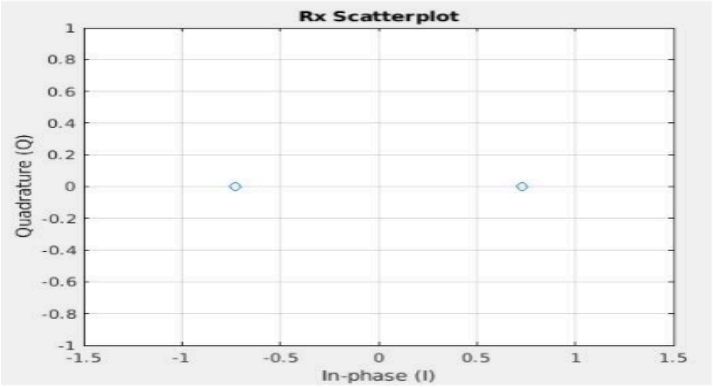
Symbol Timing Estimation

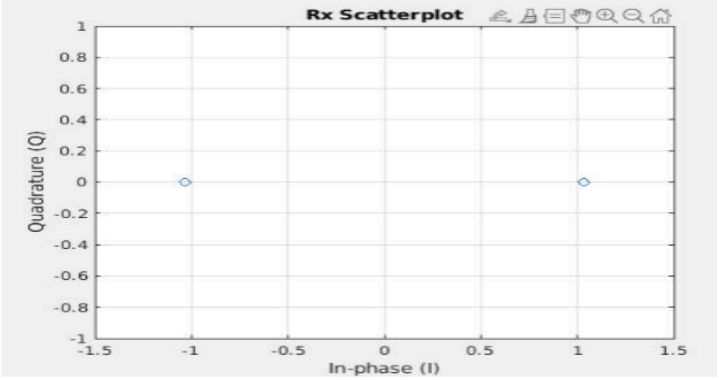
```
L        = 4;
rollOff  = 0.5;
rcDelay  = 10;
htx = rcosdesign(rollOff, 6, 4);
hrx  = conj(fliplr(htx));
p = conv(htx,hrx);

M = 2;
data = zeros(1, 2*rcDelay);
data(1:2:end) = 1;
txSym = real(pammod(data, M));
txUpSequence = upsample(txSym, L);
txSequence = filter(htx, 1, txUpSequence);

timeOffset = 1;
rxDelayed = [zeros(1, timeOffset), txSequence(1:end-timeOffset)];
mfOutput = filter(hrx, 1, rxDelayed);
rxSym = downsample(mfOutput, L);
selectedSamples = upsample(rxSym, L);
selectedSamples(selectedSamples == 0) = NaN;

figure
plot(complex(rxSym(rcDelay+1:end)), 'o')
```

```
grid on
xlim([-1.5 1.5])
title('Rx Scatterplot')
xlabel('In-phase (I)')
ylabel('Quadrature (Q)')

figure
stem(rxSym)
title('Symbol Sequence with delay')
xlabel('Symbol Index')
ylabel('Amplitude')

rxSym = downsample(mfOutput, L, timeOffset);
selectedSamples = upsample(rxSym, L);
selectedSamples(selectedSamples == 0) = NaN;

figure
plot(complex(rxSym(rcDelay+1:end)), 'o')
grid on
xlim([-1.5 1.5])
title('Rx Scatterplot')
xlabel('In-phase (I)')
ylabel('Quadrature (Q)')

figure
stem(rxSym)
title('Symbol Sequence without delay')
xlabel('Symbol Index')
ylabel('Amplitude')
```









EXP–7

Frequency selective and non-selective fading channel

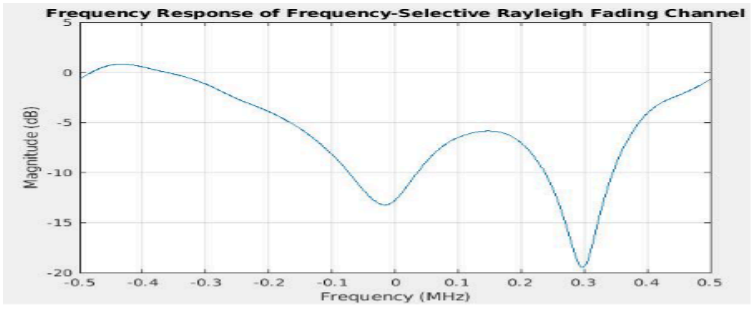
Frequency selective

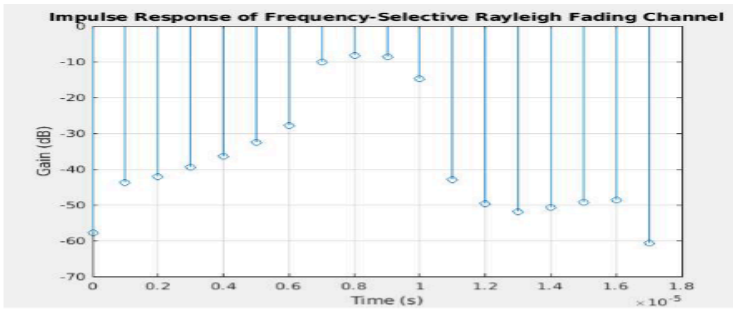
```
clc;clear all;close all;
fs = 1e6;
numSamples = 10000;
numPaths = 5;
maxDelay = 3e-6;
dopplerShift = 100;

impulseSignal = [1; zeros(numSamples-1, 1)];
rayleighChan = comm.RayleighChannel( ...
    'SampleRate', fs, ...
    'PathDelays', linspace(0, maxDelay, numPaths), ...
    'AveragePathGains', [-2 -3 -6 -8 -10], ...
    'MaximumDopplerShift', dopplerShift, ...
    'NormalizePathGains', true);
rxImpulseSignal = rayleighChan(impulseSignal);
timeAxis = (0:numSamples-1)/fs;

figure;
stem(timeAxis(1:100), 20*log10(abs(rxImpulseSignal(1:100))));
title('Impulse Response of Frequency-Selective Rayleigh Fading Channel');
xlabel('Time (s)');
ylabel('Gain (dB)');
grid on;

NFFT = 1024;
freqResponse = fft(rxImpulseSignal, NFFT);
freqAxis = linspace(-fs/2, fs/2, NFFT);
figure;
plot(freqAxis/1e6, 20*log10(abs(fftshift(freqResponse))));
title('Frequency Response of Frequency-Selective Rayleigh Fading Channel');
xlabel('Frequency (MHz)');
ylabel('Magnitude (dB)');
grid on;
```

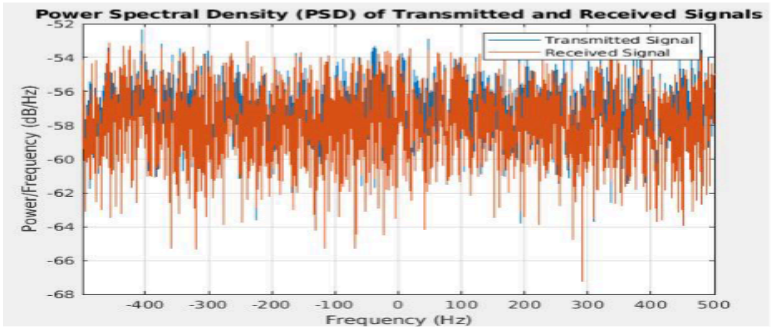
Frequency-non selective

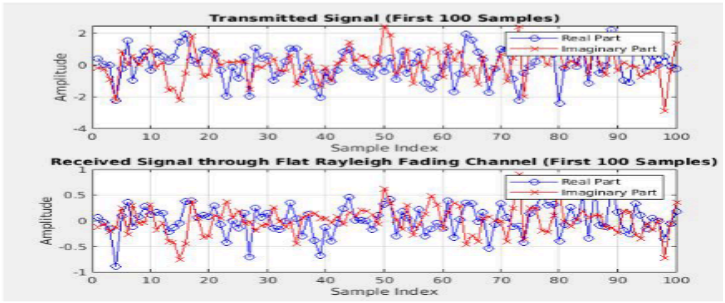
```
clc;clear all;close all
fs = 1e6;
numSamples = 10000;
maxDopplerShift = 100;

txSignal = (randn(numSamples, 1) + 1j*randn(numSamples, 1));
rayleighChan = comm.RayleighChannel( ...
    'SampleRate', fs, ...
    'MaximumDopplerShift', maxDopplerShift, ...
    'NormalizePathGains', true);
rxSignal = rayleighChan(txSignal);

figure;subplot(2, 1, 1);
plot(real(txSignal(1:100)), 'b-o');
hold on;
```

```
plot(imag(txSignal(1:100)), 'r-x');
title('Transmitted Signal (First 100 Samples)');
xlabel('Sample Index');
ylabel('Amplitude');
legend('Real Part', 'Imaginary Part');
grid on;
subplot(2, 1, 2);
plot(real(rxSignal(1:100)), 'b-o');
hold on;
plot(imag(rxSignal(1:100)), 'r-x');
title('Received Signal through Flat Rayleigh Fading Channel (First 100 Samples)');
xlabel('Sample Index');
ylabel('Amplitude');
legend('Real Part', 'Imaginary Part');
grid on;

figure;
pwelch(txSignal, [], [], [], fs, 'centered');
hold on;
pwelch(rxSignal, [], [], [], fs, 'centered');
title('Power Spectral Density (PSD) of Transmitted and Received Signals');
xlabel('Frequency (Hz)');
ylabel('Power/Frequency (dB/Hz)');
legend('Transmitted Signal', 'Received Signal');
grid on;
```









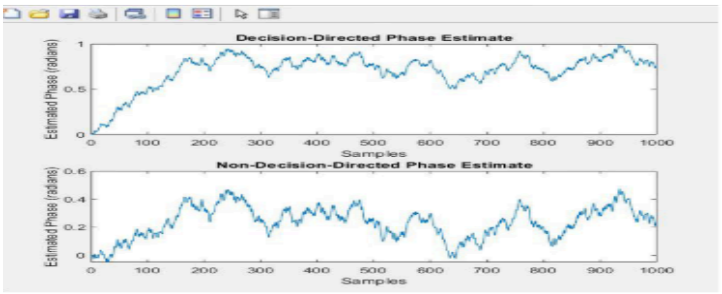
EXP-8

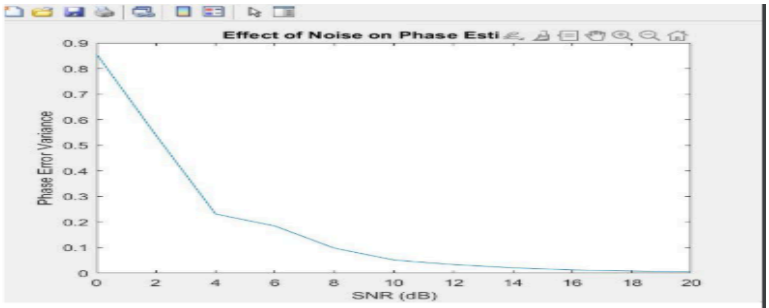
Carrier phase:Decision-Direct and Non-Decision-Directed loop

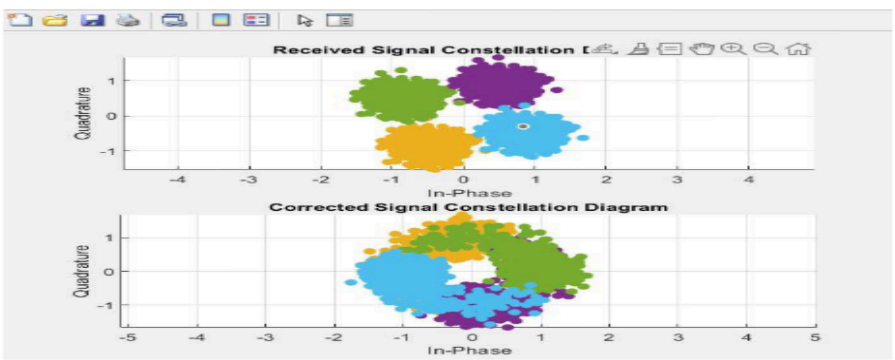
```
% MATLAB Code for Carrier Phase Estimation
clear all; clc;
N = 1000;
SNR_dB = 10;
M = 4;
loop_bandwidth = 0.01;
```

```
true_phase = pi / 3;

% Generate QPSK symbols
tx_symbols = exp(1j * (2 * pi * (0:M-1) / M));
tx_data = randi([0 M-1], N, 1);
tx_signal = tx_symbols(tx_data + 1);

noise = (1/sqrt(2*10^(SNR_dB/10))) * (randn(N, 1) + 1j * randn(N, 1));
rx_signal = tx_signal .* exp(1j * true_phase) + noise;
estimated_phase_ml = angle(sum(conj(tx_signal) .* rx_signal));
phase_error_pll = zeros(N, 1);
estimated_phase_pll = zeros(N, 1);
current_phase_estimate_pll = 0;
for n = 1:N
    phase_error_pll(n) = angle(rx_signal(n) * exp(-1j * current_phase_estimate_pll));
    current_phase_estimate_pll = current_phase_estimate_pll + loop_bandwidth * phase_error_pll(n);
    estimated_phase_pll(n) = current_phase_estimate_pll;
end
corrected_rx_signal = rx_signal .* exp(-1j * estimated_phase_pll);

figure;
subplot(2, 1, 1);
scatter(real(rx_signal), imag(rx_signal), 'filled');
title('Received Signal Constellation Diagram');
xlabel('In-Phase');
ylabel('Quadrature');
axis equal;
grid on;

subplot(2, 1, 2);
scatter(real(corrected_rx_signal), imag(corrected_rx_signal), 'filled');
title('Corrected Signal Constellation Diagram');
xlabel('In-Phase');
ylabel('Quadrature');
axis equal;
grid on;

SNR_dB_range = 0:2:20; % SNR range for variance calculation
phase_error_variance = zeros(length(SNR_dB_range), 1);
for idx = 1:length(SNR_dB_range)
    SNR_dB = SNR_dB_range(idx);
    noise_variance = 1/(2*10^(SNR_dB/10));
    noise = sqrt(noise_variance) * (randn(N, 1) + 1j * randn(N, 1));
    rx_signal = exp(1j * true_phase) + noise;
    estimated_phase = angle(sum(rx_signal));
    phase_error_variance(idx) = var(angle(rx_signal) - true_phase);
end

figure;
plot(SNR_dB_range, phase_error_variance);
xlabel('SNR (dB)');
ylabel('Phase Error Variance');
title('Effect of Noise on Phase Estimation');
```

```
phase_error_dd = zeros(N, 1);
phase_error_ndd = zeros(N, 1);
estimated_phase_dd = zeros(N, 1);
estimated_phase_ndd = zeros(N, 1);

current_phase_estimate_dd = 0;
current_phase_estimate_ndd = 0;
for n = 1:N
    noise = (1/sqrt(2*10^(SNR_dB))) * (randn + 1j * randn);
    rx_signal = tx_signal(n) * exp(1j * true_phase) + noise;

    % Decision-Directed (DD) Loop
    detected_symbol = exp(1j * round(angle(rx_signal) * M / (2 * pi)) * 2 * pi / M);
    phase_error_dd(n) = angle(detected_symbol * exp(-1j * current_phase_estimate_dd));
    current_phase_estimate_dd = current_phase_estimate_dd + loop_bandwidth * phase_error_dd(n);
    estimated_phase_dd(n) = current_phase_estimate_dd;

    % Non-Decision-Directed (NDD) Loop
    phase_error_ndd(n) = angle(rx_signal * exp(-1j * current_phase_estimate_ndd));
    current_phase_estimate_ndd = current_phase_estimate_ndd + loop_bandwidth * phase_error_ndd(n);
    estimated_phase_ndd(n) = current_phase_estimate_ndd;
end

figure;
subplot(2, 1, 1);
plot(1:N, estimated_phase_dd);
title('Decision-Directed Phase Estimate');
xlabel('Samples');
ylabel('Estimated Phase (radians)');

subplot(2, 1, 2);
plot(1:N, estimated_phase_ndd);
title('Non-Decision-Directed Phase Estimate');
xlabel('Samples');
ylabel('Estimated Phase (radians)');
```



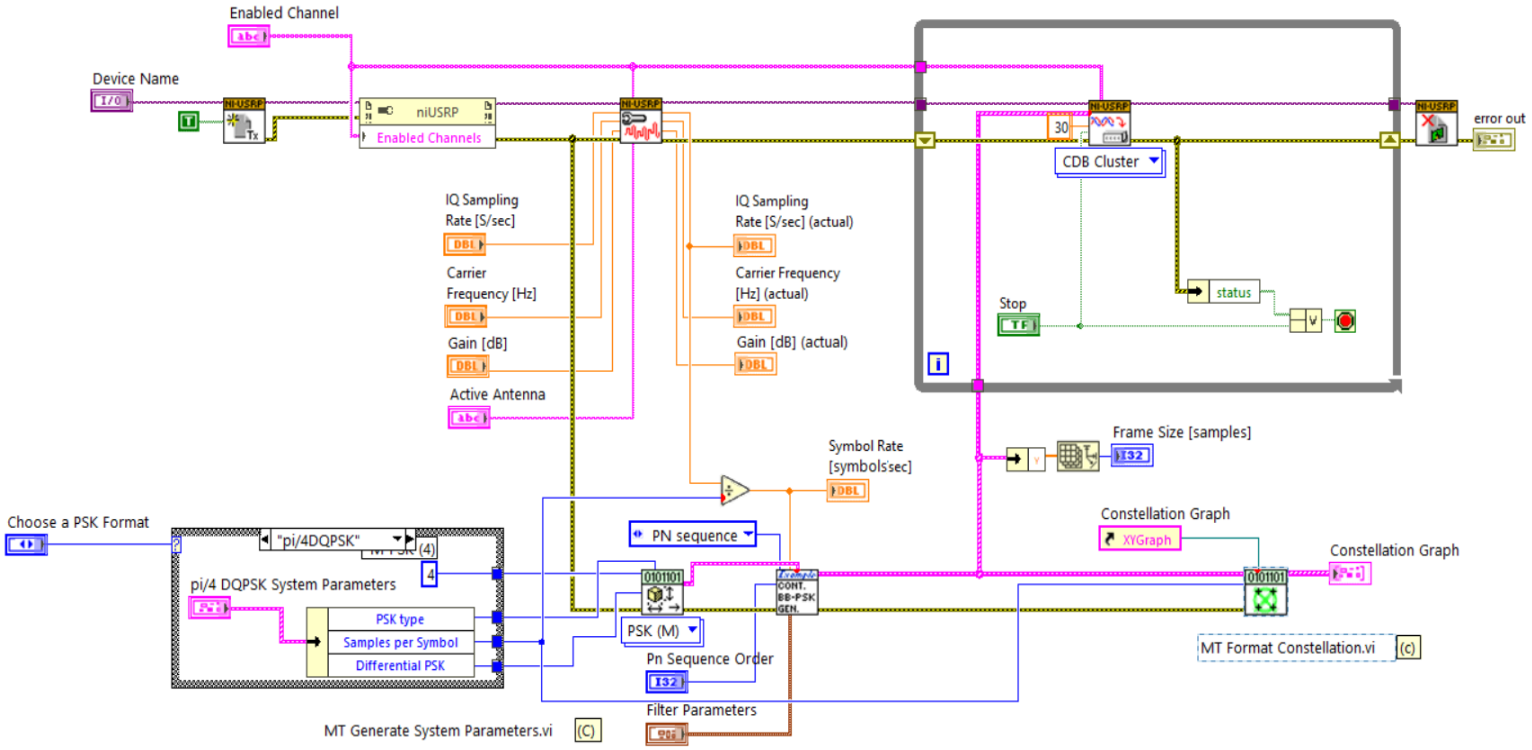




EXP-9

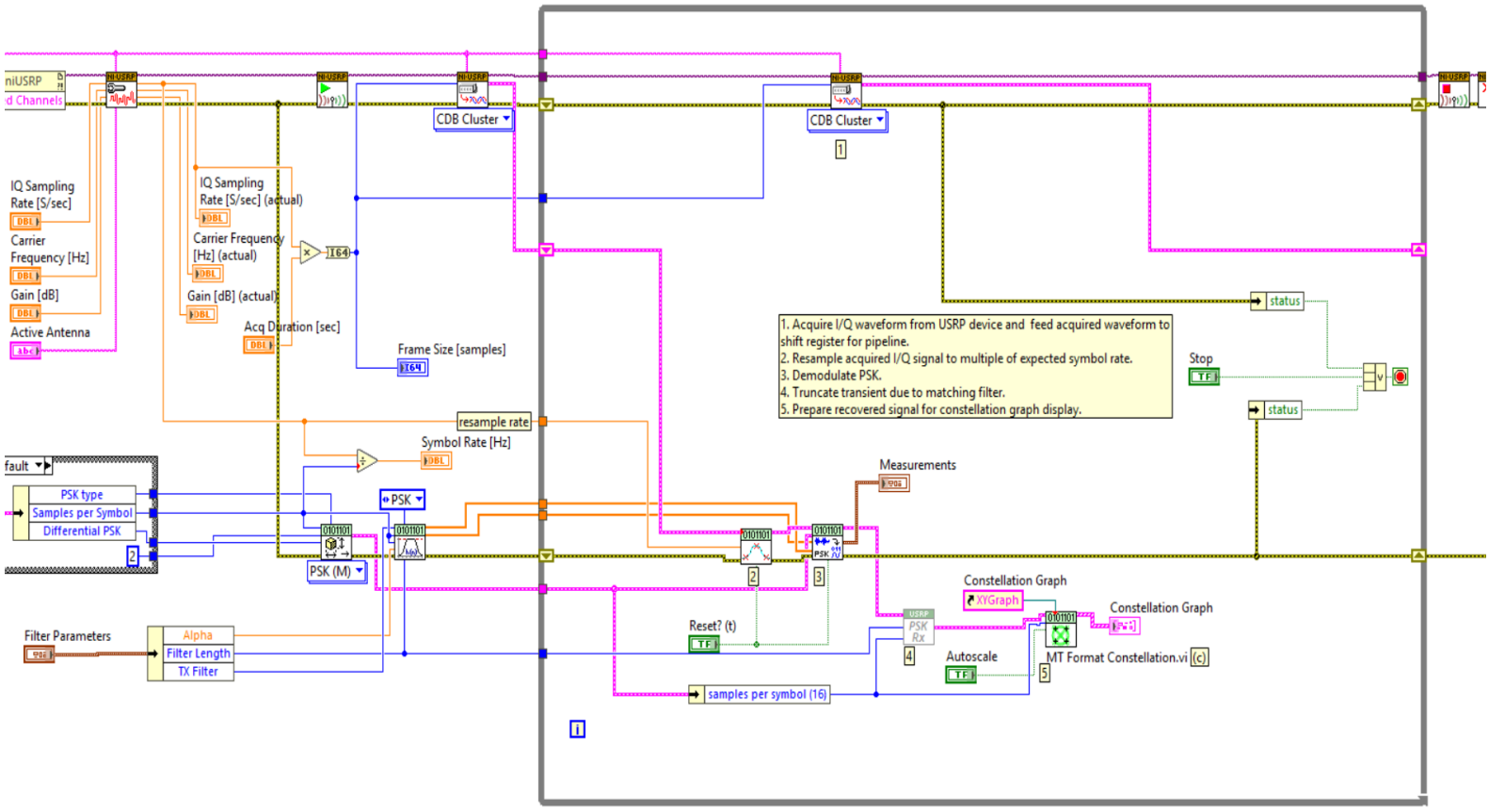
M-array Phase Shift Keying Modulation and Demodulation using USRP2920

MPSK Transmitter



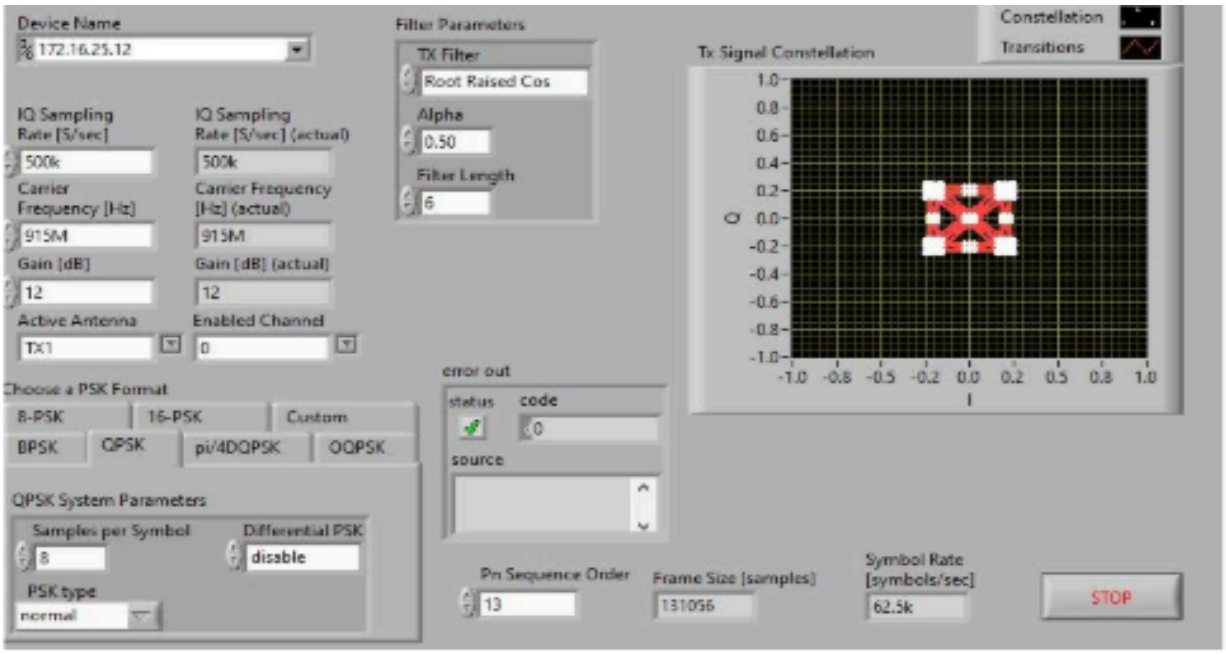
1. **MT Format Constellation.vi:** Available in search or Quick drop (ctrl + Spacebar) Path.
2. **MT Generate System Parameters.vi:** Available in search or Quick drop (ctrl + Spacebar) Path.

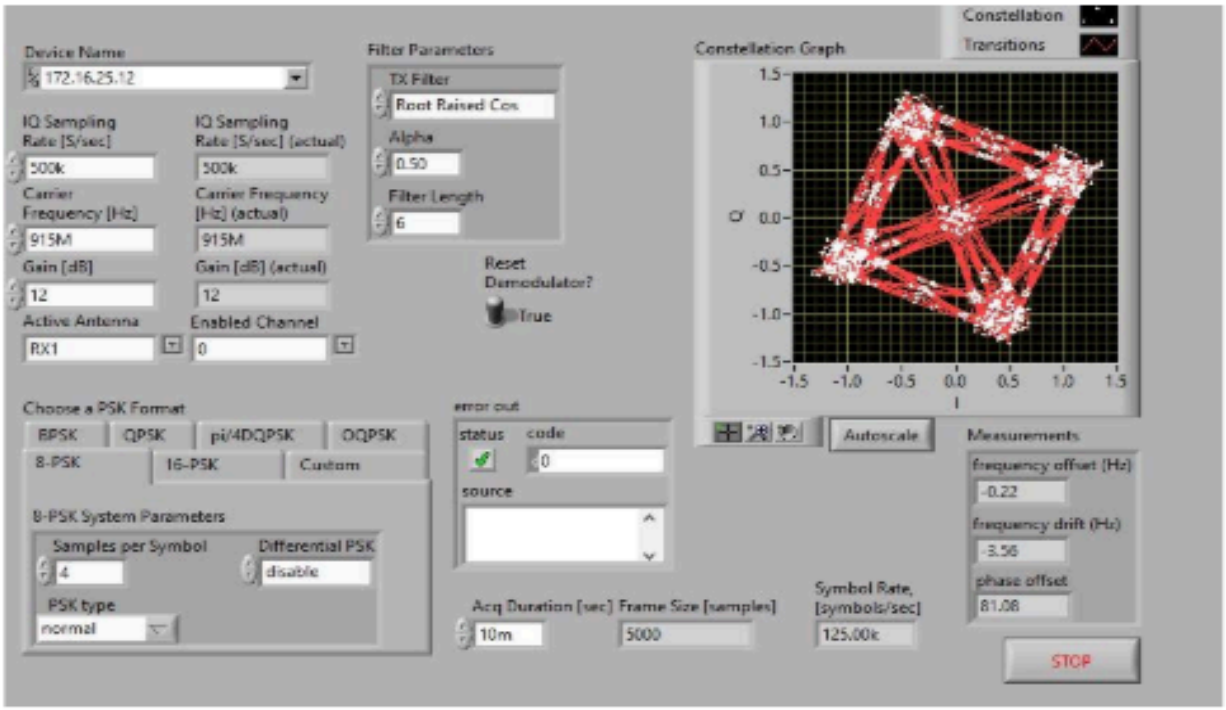
MPSK Receiver



1. **MT Format Constellation.vi:** Available in search or Quick drop (ctrl + Spacebar) Path

Output

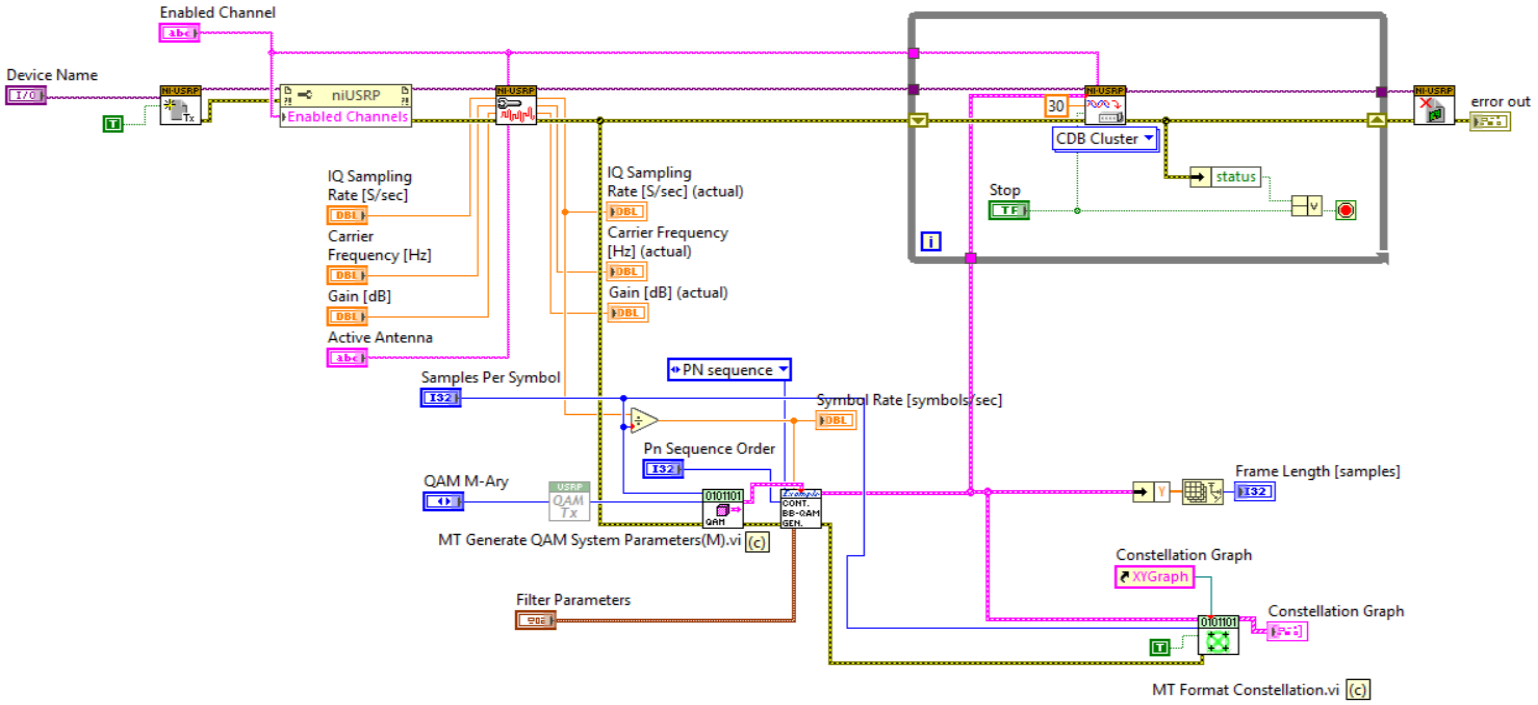

Device Name
172.16.25.12
Filter Parameters
TX Filter
Root Raised Cos
Alpha
0.50
Filter Length
6
Constellation
Transitions
Tx Signal Constellation
IQ Sampling Rate [S/sec]
500k
IQ Sampling Rate [S/sec] (actual)
500k
Carrier Frequency [Hz]
915M
Carrier Frequency [Hz] (actual)
915M
Gain [dB]
12
Gain [dB] (actual)
12
Active Antenna
TX1
Enabled Channel
0
error out
status
code
0
source
Choose a PSK Format
8-PSK
16-PSK
Custom
BPSK
QPSK
pi/4DQPSK
OQPSK
QPSK System Parameters
Samples per Symbol
8
Differential PSK
disable
PSK type
normal
Pn Sequence Order
13
Frame Size [samples]
131056
Symbol Rate [symbols/sec]
62.5k
STOP


Device Name
172.16.25.12
Filter Parameters
TX Filter
Root Raised Cos
Alpha
0.50
Filter Length
6
Constellation
Transitions
Constellation Graph
IQ Sampling Rate [S/sec]
500k
IQ Sampling Rate [S/sec] (actual)
500k
Carrier Frequency [Hz]
915M
Carrier Frequency [Hz] (actual)
915M
Gain [dB]
12
Gain [dB] (actual)
12
Reset Demodulator?
True
Active Antenna
RX1
Enabled Channel
0
error out
status
code
0
source
Choose a PSK Format
BPSK
QPSK
pi/4DQPSK
OQPSK
8-PSK
16-PSK
Custom
8-PSK System Parameters
Samples per Symbol
4
Differential PSK
disable
PSK type
normal
Autoscale
Measurements
frequency offset [Hz]
-0.22
frequency drift [Hz]
-3.56
phase offset
81.08
Acq Duration [sec]
10m
Frame Size [samples]
5000
Symbol Rate, [symbols/sec]
125.00k
STOP

EXP-10

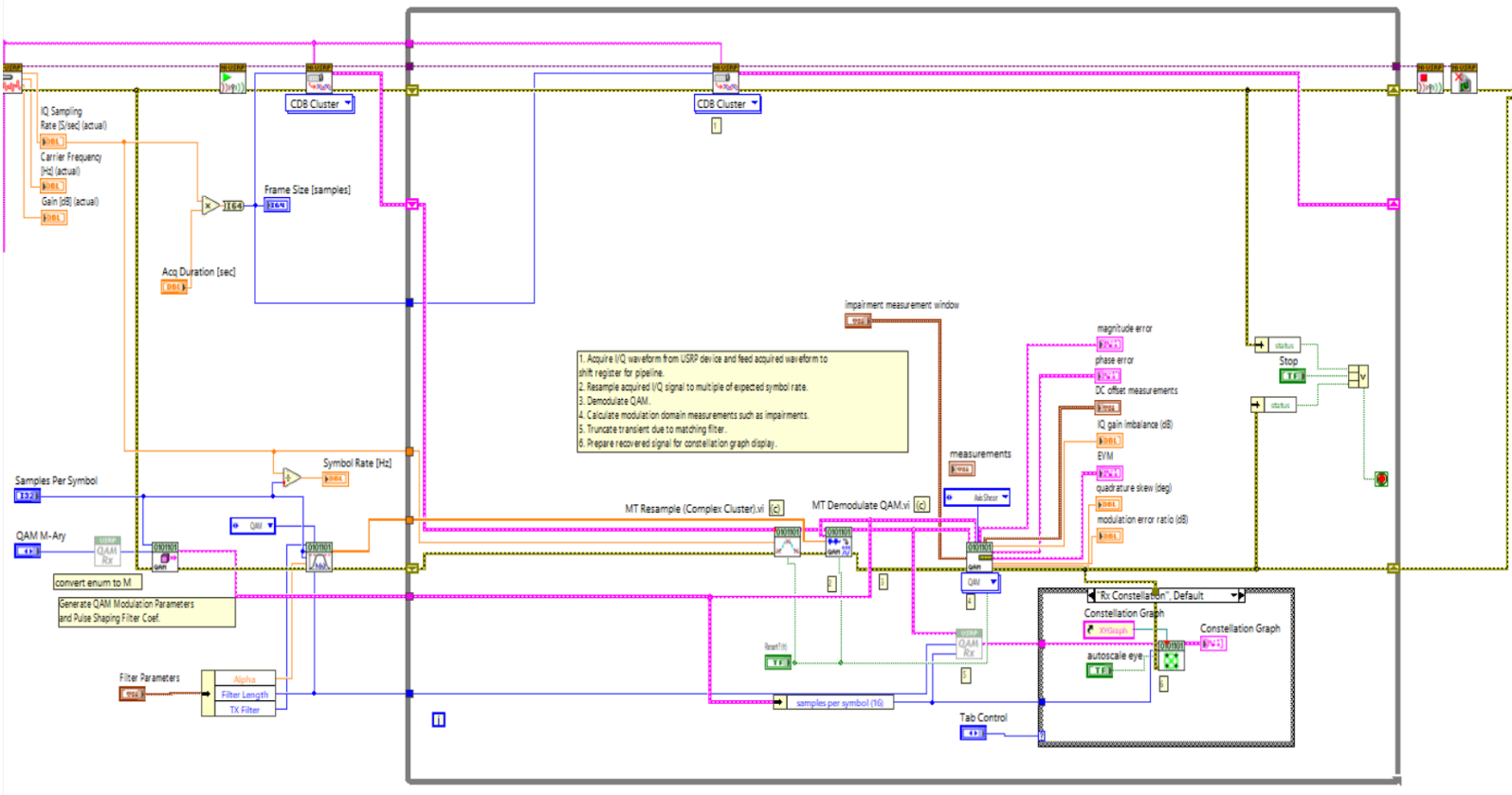
M-ary QAM Modulation and Demodulation using USRP2920

M-QAM Transmitter



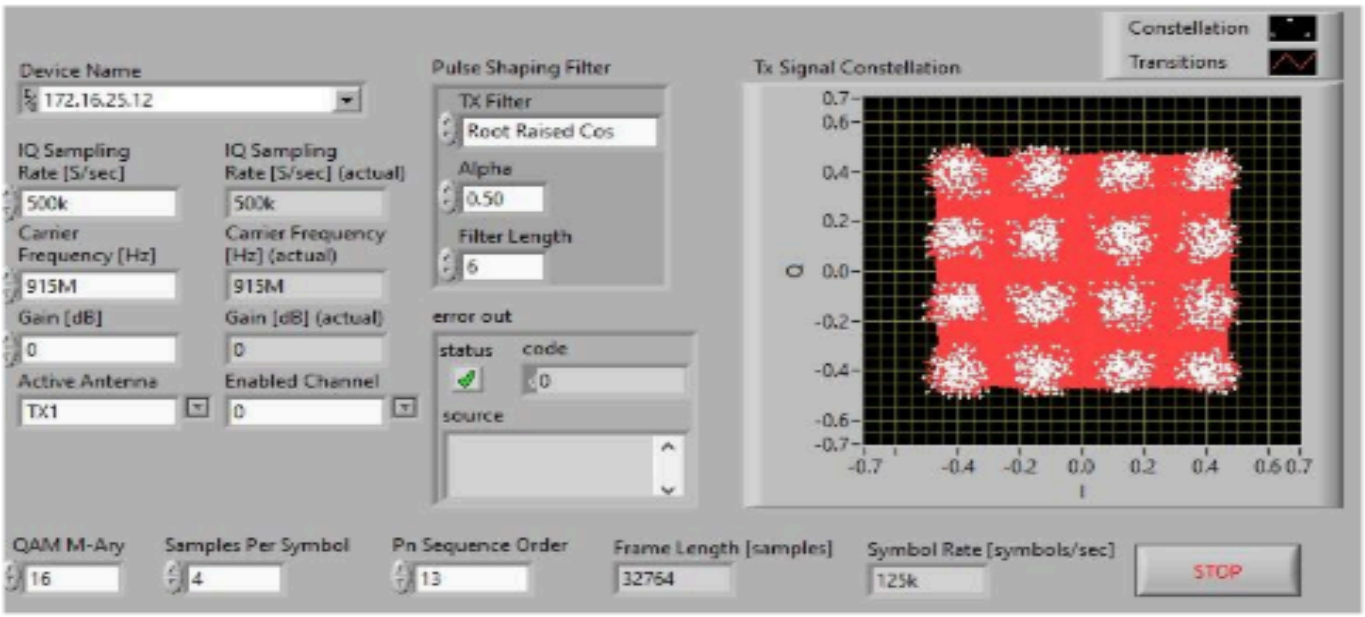
1. **MT Generate QAM System Parameters(M).vi:** Available in search or Quick drop (ctrl + Spacebar) Path
2. **MT Format Constellation.vi:** Available in search or Quick drop (ctrl + Spacebar) Path

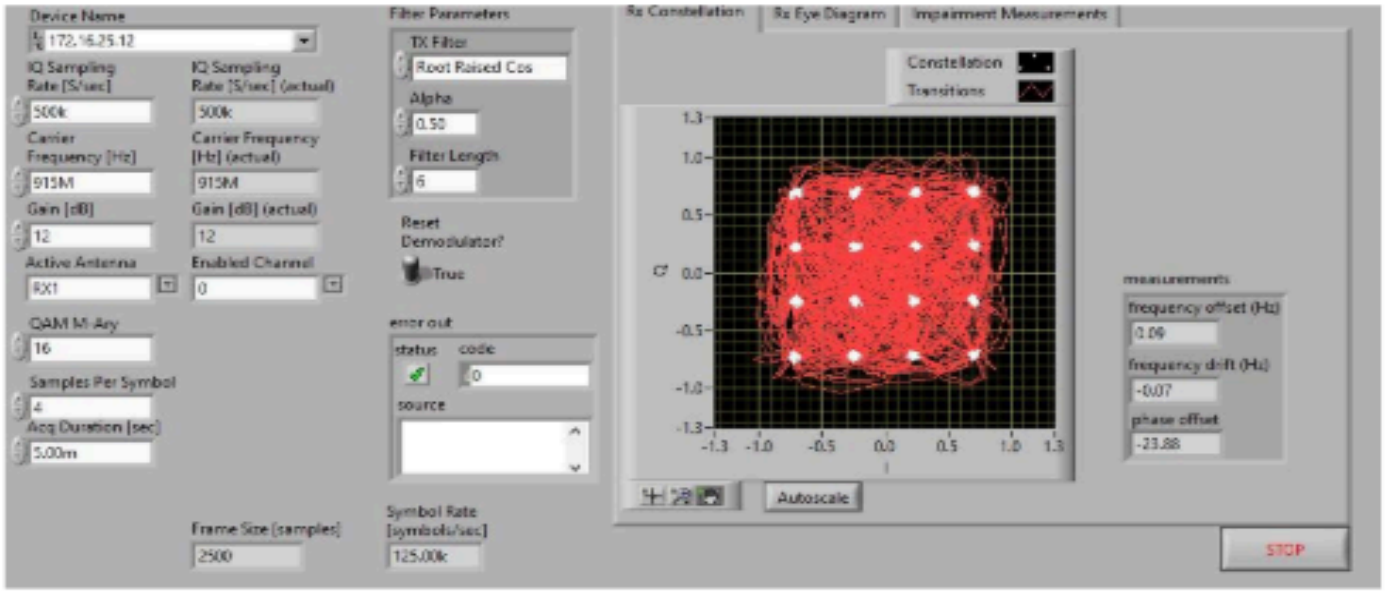
M-QAM Receiver



1. **MT Resample(complex cluster).vi:** Available in search or Quick drop (ctrl + Spacebar) Path
2. **MT Demodulate QAM.vi:** Available in search or Quick drop (ctrl + Spacebar) Path

Output


Device Name
172.16.25.12
IQ Sampling Rate [S/sec]
500k
IQ Sampling Rate [S/sec] (actual)
500k
Carrier Frequency [Hz]
915M
Carrier Frequency [Hz] (actual)
915M
Gain [dB]
0
Gain [dB] (actual)
0
Active Antenna
TX1
Enabled Channel
0
Pulse Shaping Filter
TX Filter
Root Raised Cos
Alpha
0.50
Filter Length
6
error out
status
code
0
source
Tx Signal Constellation
Constellation
Transitions
Q
I
QAM M-Ary
16
Samples Per Symbol
4
Pn Sequence Order
13
Frame Length [samples]
32764
Symbol Rate [symbols/sec]
125k
STOP


Device Name
172.16.25.12
IQ Sampling Rate [S/sec]
500k
IQ Sampling Rate [S/sec] (actual)
500k
Carrier Frequency [Hz]
915M
Carrier Frequency [Hz] (actual)
915M
Gain [dB]
12
Gain [dB] (actual)
12
Active Antenna
RX1
Enabled Channel
0
QAM M-Ary
16
Samples Per Symbol
4
Acq Duration [sec]
5.00m
Filter Parameters
TX Filter
Root Raised Cos
Alpha
0.50
Filter Length
6
Reset Demodulator?
True
error out
status
code
0
source
Frame Size [samples]
2500
Symbol Rate [symbols/sec]
125.00k
Rx Constellation
Rx Eye Diagram
Impairment Measurements
Constellation
Transitions
Q
I
Autoscale
measurements
frequency offset (Hz)
0.00
frequency drift (Hz)
-0.07
phase offset
-23.88
STOP