```
    '3F83'
    '58BA'
    'D030'
    'F231'
```

Create samples of the preamble for use in a preamble detector to detect the preamble. Because the COSPAS-SARSAT specification does not specify the type of pulse shaping to be used, create the samples using QPSK at the symbol rate.

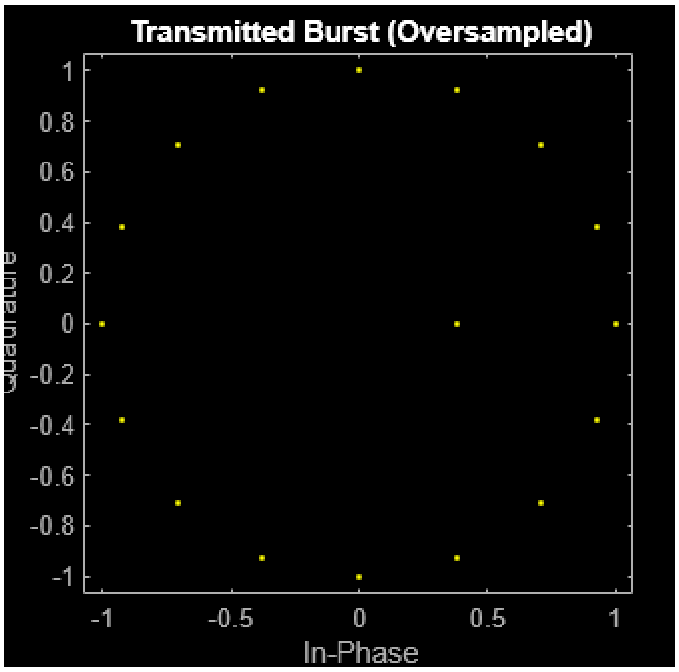
```
preambleQPSK = pskmod(reshape([DSSS.PRN_I(1:numPreambleChips/2)
DSSS.PRN_Q(1:numPreambleChips/2)].', ...
    1,numPreambleChips).',4,pi/4,InputType='bit');
```

Generate a transmit burst using the COSPAS-SARSAT specification and user-defined settings.

```
tx = helperDSSSTransmitter(system,sps,DSSS);
```

OQPSK modulation produces a signal with constant envelope. This type of signal minimizes out-of-band emissions from discontinuous phase and maximizes transmit power output from power amplifiers because it avoids the nonlinear amplifier region. Show that the transmitted burst is a constant envelope signal.

```
% Show the oversampled transmitted constellation
figure;
scatterplot(tx.Samples);
title('Transmitted Burst (Oversampled)');
```



Channel and Receiver Impairments

Create an over-the-air (OTA) buffer that contains all samples for the entire simulation. Add an offset to the transmission burst to insert a period of silence before the burst is transmitted. Apply frequency offset due to Doppler from the receiver speed.

```
txOffset = floor(settings.burstDelay * fs); % number of empty samples before the
transmission burst appears in the buffer
otaBuffer = complex(zeros(floor(settings.simTime*fs),1));
otaBuffer(txOffset:txOffset+numBurstSamples-1) = tx.Samples;

% Apply frequency offset due to Doppler from the receiver speed
otaBuffer = frequencyOffset(otaBuffer,fs,dopplerShift);
```

Path Loss Modeling

Thermal noise density at 25 degrees Celsius is -174 dBm/Hz. The bandwidth of COSPAS is 2*chipRate, or 76.8 kHz. The thermal noise for a 76.8 kHz receiver is -174 + 10*log10(76.8e3) = -125.1 dBm.

Using the default parameters, the receiver is at a distance of 300km. The free space path loss is `fspl`(300e3, 3e8 / 406.05e6) = 134.2 dB. Minimum equivalent isotropic radiated power (EIRP) is 33 dBm. Therefore, the received power level is 33 - 134.2 = -101.2 dBm. This gives a 24 dB SNR. When considering Eb/No, a spreading factor of 256 provides a spreading gain of 24.1 dB.

Calculate the SNR resulting from the given transmit power and signal power loss due to distance between the transmitter and receiver.

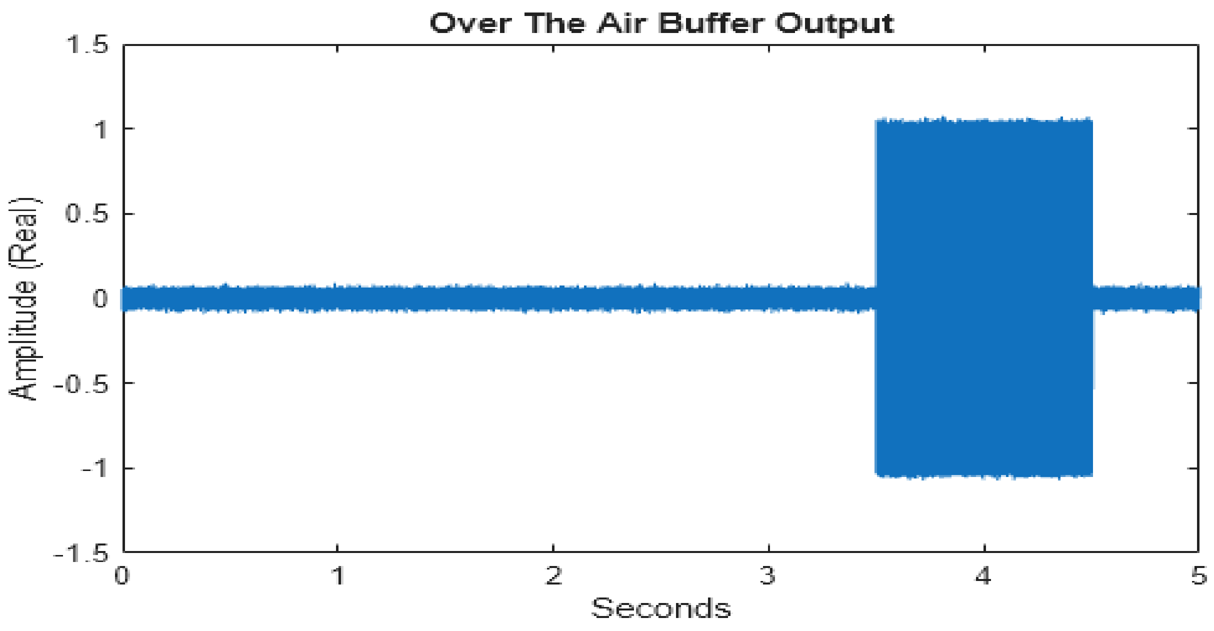
```
noiseFloorPow = -174 + pow2db(2*system.chipRate);  % noise floor (dBm)
pathLoss = fspl(settings.distance,physconst('LightSpeed')/(system.fc*1e6));
SNR = (settings.txPower - pathLoss) - noiseFloorPow;
fprintf('Simulation SNR = %4.1f dB\n',SNR);
```

```
Simulation SNR = 24.0 dB
```

Apply Receiver Impairments

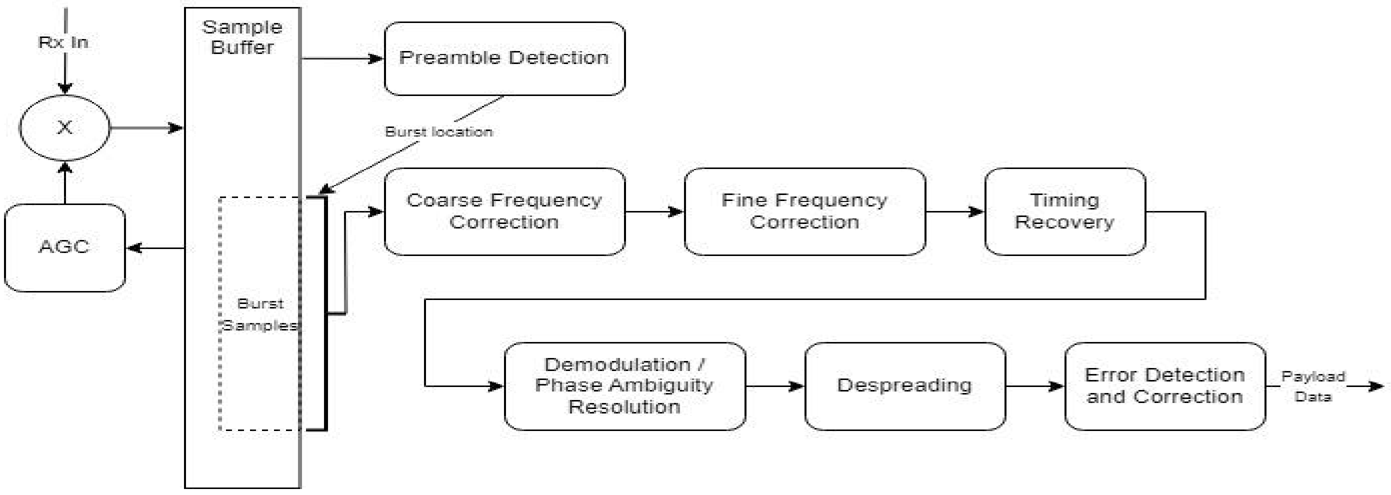
Apply frequency offset and AWGN impairments to the received samples. The mismatch of the transmitter and receiver oscillators causes frequency offset between the transmitted and received signal. The receiver front end adds AWGN to the received signal.

```
% Add frequency offset from receiver offset
otaBuffer = frequencyOffset(otaBuffer,fs,impairments.rxFreqOffset);

% Add AWGN according to the path loss calculation
otaBuffer = awgn(otaBuffer,SNR,'measured');

% Plot the received signal
figure;
plot(settings.simTime/length(otaBuffer):settings.simTime/
length(otaBuffer):settings.simTime,real(otaBuffer));
title('Over The Air Buffer Output');
xlabel('Seconds');
ylabel('Amplitude (Real)');
```



Receiver

Preamble detection finds and captures a complete burst in the sample buffer for processing. The receiver faces challenges of unknown transmission time, frequency offset from dissimilar radio oscillators and Doppler shift, phase offset from asynchronous sampling, timing drift from dissimilar clocks, and channel conditions from path loss and multipath. The receiver compensates for all these impairments by the end of the preamble so that it can synchronize the payload data for demodulation.

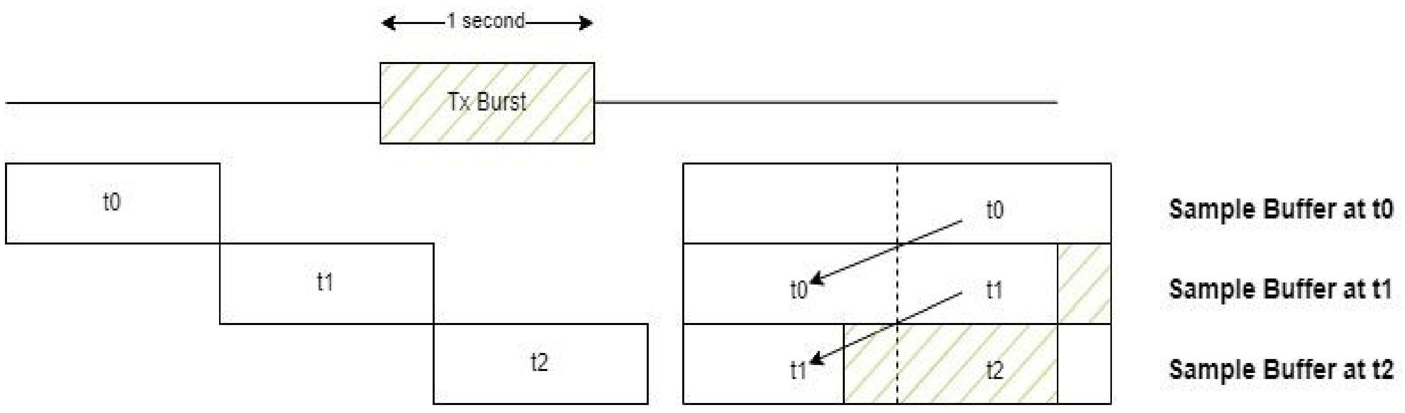

Rx In
X
AGC
Sample Buffer
Burst Samples
Preamble Detection
Burst location
Coarse Frequency Correction
Fine Frequency Correction
Timing Recovery
Demodulation / Phase Ambiguity Resolution
Despreading
Error Detection and Correction
Payload Data

The automatic gain control (AGC) adjusts the incoming signal amplitudes that are affected by path loss. Preamble detection continuously searches for the beginning of a COSPAS-SARSAT burst within the sample buffer. Once a burst is detected, the burst samples are extracted from the sample buffer and passed on to the coarse frequency correction block to estimate the frequency offset and correct the offset. Fine frequency correction is done next to get the frequency offset as close to zero as possible. Timing recovery takes the oversampled signal and resolves the proper phase such that the constellation aligns with the reference QPSK constellation. The constant phase alignment also compensates for timing drift. The signal is then demodulated several times with different phase and symbol offsets to resolve phase ambiguity resulting from the frequency correction and timing recovery algorithms. The QPSK symbols are despread and the decoded bits are passed through an error detection block to detect and correct errors within the payload data.

Sample Buffering

Since beacon transmissions are asynchronous, part of the burst might be captured in one buffer, while the latter part of the burst might be contained in the next buffer. To ensure capture of a burst within a single buffer, a double-buffering or "ping-pong" scheme is used where a sample buffer is created with a length that can store two complete transmission bursts. The sample buffer is partitioned into two sections: a previous sample buffer and a recent sample buffer. The second half of the sample buffer is filled with one second of the most recent samples, while the first half of the sample buffer contains the previous samples. When the most recent sample buffer is filled, a search for the preamble may be done across both sample buffers. If no preamble is detected, the recent sample buffer replaces the older sample buffer. The length of the sample buffer guarantees that a complete burst spans both sample buffers.

The sample buffer is shown on the right in the figure below. The right buffer is filled with new data, then shifted to the left after each time step. At time t0, the sample buffer does not capture any of the transmit burst. At time t1, a portion of the transmit burst is captured in the sample buffer. At time t2, the remaining portion of the transmit burst is captured, so that the entire burst is contained within the sample buffer at time t2.



Create a sample buffer with a length twice that needed to store a complete transmission burst. Begin feeding in one second of samples, apply automatic gain control, and search for the preamble.

```
numBuffers = floor(length(otaBuffer)/numBurstSamples);
sampleBuffer = complex(randn(numBurstSamples*2,1),randn(numBurstSamples*2,1)); % start
with empty buffer of AWGN

% Load in one second of samples then search for the preamble in the sample
% buffer
for n=1:numBuffers

    % Shift the last burst samples to front of buffer
    sampleBuffer(1:numBurstSamples) = sampleBuffer(numBurstSamples+1:end);

    % Read in next burst
    nextBurstIdx = (n-1)*numBurstSamples + 1;
    sampleBuffer(numBurstSamples+1:end) =
otaBuffer(nextBurstIdx:nextBurstIdx+numBurstSamples-1); % Rx in
```

Automatic Gain Control

The receiver's dynamic range must be able to detect faint signals from maximum distance up to strong signals received at close range. The bursty nature of ELT signals means that the AGC reaches maximum gain in between transmission bursts. The AGC then must quickly adjust the gain when the burst is received at the receiver front end to avoid signal saturation at the power amplifier and/or analog-to-digital converter (ADC).

Normalize the receiver input signals using a `comm.AGC` System object such that the signal power is unity. Configure the AGC to average over 10 symbols and set the maximum gain to be able to amplify down to the noise floor. This example does not model quantization noise effects from the ADC.

```
    % Configure the AGC and process the samples
    agc = comm.AGC( ...
        AdaptationStepSize = 0.01, ...
        AveragingLength =    10*sps, ...
        MaxPowerGain =       210);
    [rxAGCSamples,pwrLevel] = agc(sampleBuffer);

    % Saturate signals to prevent false preamble detection during correlation
    rxAGCSamples = saturate(rxAGCSamples,1.2);
```

Preamble Detection

The first cluster of samples in the preamble are assumed to be unusable while the AGC adjusts the input received signal power to the proper setpoint. Use only the latter part of the preamble for preamble detection by specifying an offset. Correlation with a long preamble results in lower probability of false alarm (false detection of preambles) in low SNR, but it is susceptible to low probability of detection when frequency offset is high. Shorten the length of the correlated preamble to combat frequency offset effects but maintain low false alarm probability. Frequency offset adversely affects correlator-based preamble detection, so different frequency offsets are applied to the reference preamble during the preamble search.