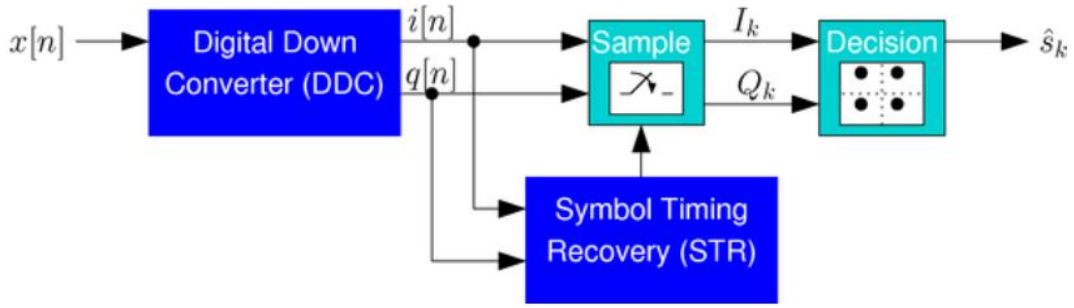
1 Introduction



Symbol timing recovery

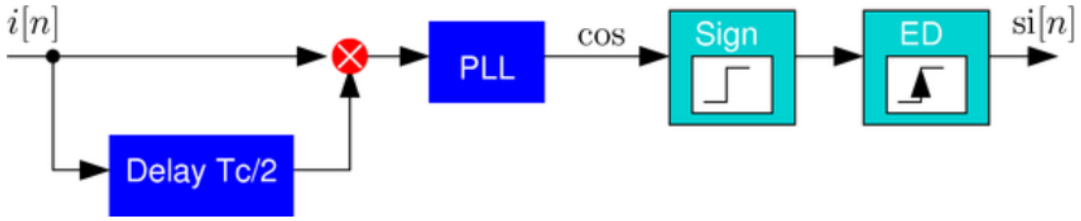


Figure 1: Symbol timing Recovery block module

A baseband waveform is taken to generate a sample indicator with stream si[n]. The si[n] indicates whether the sample should be skipped (si[n] = 0) or stored (si[n] = 1). The input is multiplied by a T/2 delayed version of itself which gives a signal at frequency of 1/T. The PLL is locked to this component and thresholded (Sign block) to obtain a sample clock. Then the rising edge detector (ED block) is used to identify optimal sample points.

Sampling and Decision Block

The output of symbol timing recovery block is sampled and made a decision to which symbol was transmitted. The decision block gives a list of estimated symbols for the current block.

Frame Synchronization

The synchronization needs to be framed to determine the starting point of the message. This is done by arranging several symbols into a frame and placing a known sequence at the beginning of the frame. By identifying the known sequence, the symbols can be decoded in a correct manner.

2 Execution/Evaluation

The system_param.m, delay, ddc codes were used from previous labs. The STR code is shown in the lab write up.

Prelab:

(1) What is the basic purpose of the symbol timing recovery (STR) block?

To sample the I/Q waveforms at optimal sampling points from the DDC and convert them back to symbols

(2) Is the input to the STR an IF or baseband waveform?

Baseband waveform.

(3) The STR works by multiplying the input signal with itself delayed by ***T/2*** *symbol periods.*

(4) The PLL that tracks the symbol timing should be tuned to a center frequency of ***1/T*** *.*

(5) How long should the sample indicator bit stream output be for a single block input into the STR?

It should be a vector as long as the block with si[n] = 0 meaning the sample should be skipped or with block si[n] = 1 meaning the sample should be stored.

(6) What do 0 and 1 indicate in the sample indicator stream?

0 indicates the DDC sample should be skipped and 1 indicates the sample should be stored.

(7) Where should the optimal sample points be relative to transitions of i[n] or q[n]?

The output of symbol timing PLL at each rising edge.

(8) What is the purpose of frame synchronization?

The purpose of frame synchronization is to determine the place to start reading the message.

(9) How does frame synchronization work?

Several symbols are arrange into a frame. Then, at the beginning of the frame a known sequence is placed. By identifying the known sequence, the beginning is found.

Check – off

Demonstrate to the TA or instructor that your STR module is working correctly. Be prepared to answer any questions about the operation of your code and how to tell if the output is correct.

Show that your complete receiver works by inputting a message signal and having the same message come out of your receiver.

Please refer to the lab write up to understand the functioning of STR and its different characteristics.

Demonstrate that adding noise eventually causes errors in the estimated symbols. You should experiment with this a little to know at what point the receiver starts to have trouble.

Original text:

ascii_test_recv
‘ .... Stand by .... Stand by .... In telecommunications and signal processing, frequency modulation (FM) is the encoding of information in a carrier wave by varying the instantaneous frequency>> ‘
>>

Received text when adding noise (Change: SNR=10):

ascii_test_recv
an` bi
. If delec/omunic`tions requency modulatikn (FM! is the encoding of information in c carridr wava by viry)n' }he an3tantanEOus f>>
>>’

Lab Write up

(1) A printout of your final MATLAB implementation of the STR and the top-level code that tests the complete receiver.

STR init function

```
function[str_state] = str_init(p)
 str_state.accum = 0;
 str_state.accum_out = 0;

 str_state.delay = delay_init(128, p.Tc/2);
 str_state.PLL = pll_init(1/p.Tc, p.T, p.xi, p.K);
end
```

STR function

```
function [si, str_state] = str(Ib,Qb, str_state)
state = str_state;

[state.delay, delay_Ib] = delay(state.delay,Ib);
Ib = Ib .* delay_Ib;

% Using PLL on the Ib and Qb
[ref accum_out state.PLL] = pll(Ib, state.PLL);
si = zeros(length(state.accum_out));

for i=2: length(accum_out)
    if accum_out(i-1) < 0.75 && accum_out(i) >= 0.75
        si(i)=1;
    else
        si(1) = 0;
    end
   str.accum = state.accum_out(end);
end
str_state = state;
```

STR testing code

```
% test_ddc.m
%   Tests the digital down-converter using a known test signal.
%% Block parameters
Ns = 100;   % Samples per block
n = [1:Ns];
%% System parameters
param = system_param;
%% Signal parameters
fs = param.fs; % Sample rate
f0 = param.f0; % Nominal Carrier frequency
ft = param.ft; % Tone frequency
fc = param.fc; % Symbol rate
cps = param.cps; % Cycles per symbol

% Amplitude of signal and tone
as = 1;
at = 1;

% Do repeated pattern over and over
Nframe = 100;
symb1 = [0 0 0 0 1 2 2 3 1 2 2 0 1 2 2 3 1 2 2 0 1 2 2 3 1 2 2 0 1 2 2 3];
% Replicate this frame over and Nframe times
symb = kron(ones(1, Nframe), symb1);

% Generate signal from symbols
[s s_debug] = make_signal_4psk(fs, f0, ft, cps, param.h_ps, as, at, symb);

% Compute number of blocks we have
Nb = floor(length(s)/Ns);

% Make signal into blocks
sb = reshape(s(1:Ns*Nb), Ns, Nb);

% DEBUG: Get components into blocks for debug.
% Tone signal alone
t_debug = reshape(s_debug.t(1:Ns*Nb), Ns, Nb);
% Carrier signal alone
c_debug = reshape(s_debug.c(1:Ns*Nb), Ns, Nb);
% Modulation signal alone
m_debug = reshape(s_debug.mod(1:Ns*Nb), Ns, Nb);

% DEBUG: Plot modulated signal
%for ii=1:Nb,
%  plot([1:Ns], real(m_debug(:, ii)), [1:Ns], imag(m_debug(:, ii)));
%  pause;
%end

% DEBUG: Reference design (ideal non-block operations)
```

```
%bb_ideal = conv(s.*conj(s_debug.c), param.ddc.h_lp);
%I_ideal = real(bb_ideal); I_ideal_b = reshape(I_ideal(1:Ns*Nb), Ns, Nb);
%Q_ideal = imag(bb_ideal); Q_ideal_b = reshape(Q_ideal(1:Ns*Nb), Ns, Nb);

% Process blocks
ddc_state = ddc_init(param.ddc);
str_state = str_init(param.str);

for ii=1:Nb,
  % Get a single block
  x = sb(:, ii);

  % Digital down converter
  [Ib Qb ddc_state ddc_debug] = ddc(x, ddc_state);
  [si str_state] = str(Ib,Qb, str_state);

  % DEBUG. Plot recovered carriers
  %plot(n, ddc_debug.cos1, n, real(s_debug.c((ii-1)*Ns+[1:Ns])));
  %plot(n, ddc_debug.sin1, n, imag(s_debug.c((ii-1)*Ns+[1:Ns])));

  % DEBUG. Plot I/Q Output vs. ideal conv operations
  %plot(n, Ib, n, I_ideal_b(:,ii));
  %plot(n, Qb, n, Q_ideal_b(:,ii));

  % DEBUG. Plot I/Q outputs
  sil = logical(si);
  idx = find(si);
  plot(n,Ib,'b', n, Qb, 'g', n(sil), Ib(sil), 'b*', n(sil), Qb(sil), 'g*');
  pause;
end
```

ASCII test code

```
% ascii_test_recv.m
%   Tests the complete receiver using an ASCII message.
% Set up standard parameters as you did before
%% System parameters
param = system_param;

% Ascii message to send (feel free to use your own message here!)
ascii_text = 'Stand by .... Stand by .... Stand by .... Four score and seven years ago our fathers
brought forth on this continent a new nation, conceived in Liberty, and dedicated to the proposition
that all men are created equal. Now we are engaged in a great civil war, testing whether that
nation, or any nation, so conceived and so dedicated, can long endure.';

% Convert the ascii message to 4PSK symbols
frm = ascii_to_symb_frame(2, param.frame.sync, param.frame.N, ascii_text);
symb = frm(:);
as = 1;
at = 1;
Ns = 192;
n = 1:Ns;
% Generate signal from symbols
[s s_debug] = make_signal_4psk(param.fs, param.f0, param.ft, param.cps, param.h_ps, as, at, symb);
Nb = floor(length(s)/Ns);
% Make signal into blocks
sb = reshape(s(1:Ns*Nb), Ns, Nb);

% Intialize state of frame synchronization code
frame_state = [];

% Initialize DDC and STR
ddc_state = ddc_init(param.ddc);
str_state = str_init(param.str);

% Process the waveform s as before ...
for ii=1:Nb,
  % Get a single block
  x = sb(:, ii);
  % Digital down converter
  [Ib Qb ddc_state ddc_debug] = ddc(x, ddc_state);
  % Symbol timing recovery
  [si str_state] = str(Ib, Qb, str_state);
  % Sample at optimal points
  si1 = find(si)
  % Decision block
  symb_out = decision(Ib(si1), Qb(si1));
  % Result should now be a vector of symbols (values 0-3) estimated from
  % the optimal sample points for the current block.  I assume here this vector is called
  % "symb_out"
```

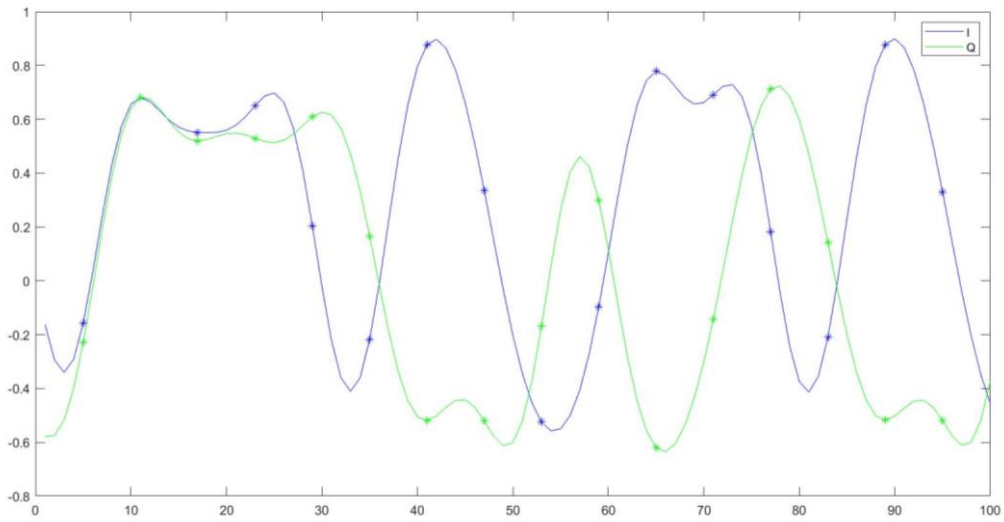
```
  % Recover frames.  This code only prints something when a complete frame is ready.
  [frame_out frame_state] = frame(symb_out, param.frame, frame_state);
  if ~isempty(frame_out),
    fprintf('%s', symb_to_ascii(2, frame_out));
  end
  pause;
end
```

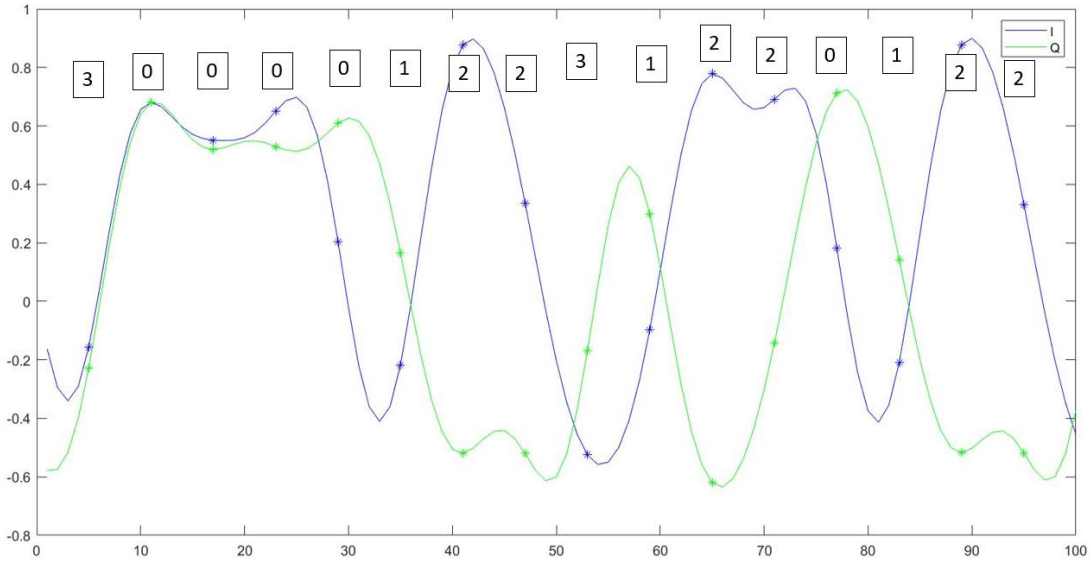
Decision function

```
function [symbol] = decision(Ib, Qb);

for i = 1:length(Ib)
    if Ib(i)>= 0
        if Qb(i) >= 0
            symbol(i) = 0;
        else
            symbol(i) = 2;
        end
    end
    if Ib(i) <= 0
        if Qb(i) <= 0
            symbol(i) = 3;
        else
            symbol(i) = 1;
        end
    end
end
```

(2) A plot showing the simulated performance of the STR, showing that the correct output is obtained. Please write a few sentences explaining how you can tell it is correct.





In the graph, 0, 1, 2, 3 symbols are shown.
The symbols can be read using the 4PSK constellation, which is indicated as:
0 symbol: positive I, positive Q
1 symbol: negative I, positive Q
2 symbol: positive I, negative Q
3 symbol: negative I, negative Q
Symbols 0 0 0 0 1 2 2 3 1 2 2 0 1 2 2 can be read according to the graph. The front part of the given symbol - symb1 = [0 0 0 0 1 2 2 3 1 2 2 0 1 2 2 3 1 2 2 0 1 2 2 3 1 2 2 0 1 2 2 3] – has the same symbol as the one in the graph, showing that the code is working well.

(3) A description of how you tested the complete receiver.

At the end of the last experiment, baseband I&Q waveforms were obtained at the output of the DDC. Then, each of them was multiplied by a T/2 shifted version of itself. The products were then tracked using PLL and their signum was taken to obtain a rectangular shape. As a last step, rising edge detector was used to identify sample points. At each step in the Matlab test script, comments were provided to explain what was done.

(4) An explanation of any problems you ran into implementing the STR or the complete receiver and how you found and fixed them.

There was a minor error in the PLL regarding the 'for' loop. The 'for' loop index was mixed with 'i' and 'k' giving wrong graphs. This minor mistake was fixed quickly. However there were still some troubles with the graph, and the no errors were found. This was fixed by moving the code to another folder. After fixing some errors, the code showed a proper graph.

3 **Conclusion**

In this experiment, a symbol timing recovery/full receiver was implemented in Matlab. Different steps of the reception process were analyzed, each of which was thoroughly explained in the receiver test description. At the end, the experiment was successful and the transmitted signal was recovered as shown in the second question of the lab write up. The effect of adding noise was also observed, showing a weird message at the receiver side. The main problems faced during the lab were a small error in the PLL and a mistake related to the file directory. Once the latter was changed, the code run and showed the expected results.

4 **References**

[1] http://dsp-fhu.user.jacobs-university.de/?page_id=234