

# Lab 8 (Continued)

## PART5: 8-PSK signal with frequency offset

- Now instead of a QPSK, generate a 8-PSK signal and introduce a frequency offset of 10kHz as done in Part1.
- Implement the Viterbi-Viterbi algorithm after the polyphase clock sync.
- As we have to estimate the frequency offset, we have to use differential decoding of the argument( $\arg(s[n]s^*[n-1])$ ) as done in the FM demodulation).



The differential decoding is followed immediately after raising the signal to its 8th power.

- The argument is to be given to a VCO (complex) block to generate the appropriate error signal.



The sensitivity of the VCO should approximately be the sample rate. Why??

✓ Observe the output of the VCO using the FFT sink and check if you are getting back the frequency offset that was given.

✓ Multiply the output of the VCO with the signal obtained from the polyphase clock sync. Observe the output and show it to your TA. Are you getting back your original 8-PSK constellation? If not, why??

•(Refer Prelab 8 document for Viterbi-Viterbi algorithm)

# PART 6: Correcting the phase offset

- Now we need to correct for the phase offset obtained in Part 5. Just by changing the phase detector, we can use the same costas loop setup to achieve this.
- Implement the phase detector for the 8-PSK signals using the Viterbi-Viterbi algorithm and give the output of Part 5 as the input to the phase detector.
- Complete the costas loop as done in part 3 and part 4.
- ✓ Observe the output and show it to your TA. Are you getting back your original constellation?
- Compare the performance of the costas loop setup and the one using the Viterbi-Viterbi method. Which one is better?

# EE 340: Communications Lab

## Lab 9

### End To End Digital Communication

# Aim

So far, we have only looked at constellation diagrams in our experiments on digital communication. The goal of this experiment is to have you perform end-to-end data transmission and get a feel of the challenges involved.

Here are the steps we'll follow:

- End to end data transmission using BPSK (Binary Phase Shift keying)  
You have to transmit a text file and recover it at the receiver
- End to end data transmission using Differential BPSK

# Task 1

- Transmit the text file uploaded on moodle using BPSK, and demodulate to recover the file.

This involves expressing the file as a bit string first. The following slides mention some GNU Radio blocks you might find useful for this task.

## Useful blocks

**File Source** : Reads raw data values in binary format from the specified file.

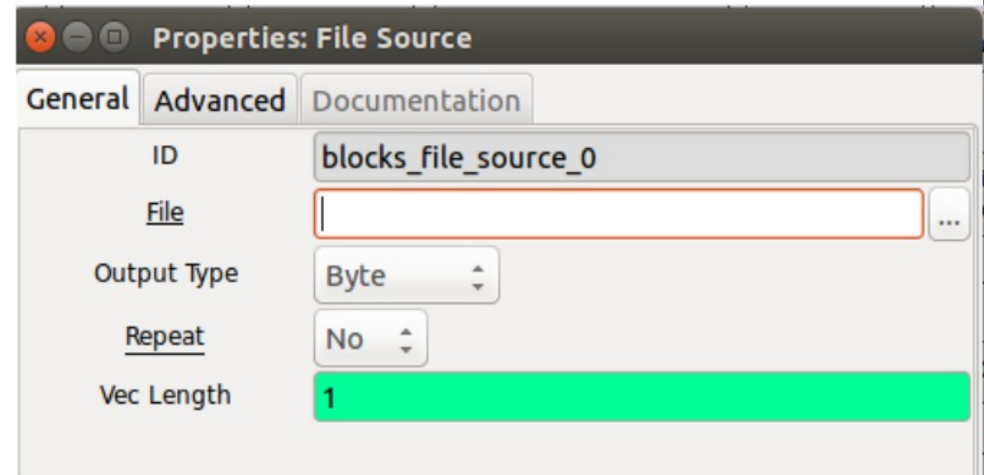
**File Source**  
**File:**  
**Repeat:** Yes

### **Note** :

Choose text file which was uploaded on Moodle.

Output Type = **Byte**.

Repeat = **No**.



The screenshot shows a window titled "Properties: File Source" with three tabs: "General", "Advanced", and "Documentation". The "General" tab is selected. It contains the following fields:

Property	Value
ID	blocks_file_source_0
File	[Empty text box with a browse button (...)]
Output Type	Byte
Repeat	No
Vec Length	1

# Contd.

**Unpack K Bits:** Converts a byte with k relevant bits to k output bytes with 1 bit in the LSB.

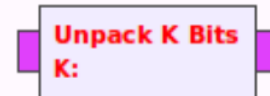
Since a text file consists of characters which are in Ascii format (each Ascii character is represented using 8 bits), we use this block to extract the bit-stream corresponding to the text file.

Eg: Ascii code for "A" is 1000001. This is packed format and output of file source block will be this as one byte. But, we need every bit of it such as 1,0,0,0,0,0,1.

If Input to Unpack K Bits is 10000001 then output will be 00000001 00000000 00000000 00000000 00000000 00000000 00000000 00000001. So every output of Unpack K Bit will be in byte format but will represent 1 bit.

The output of this block will be of the same format as that of random source and further processing to be done will be same as what we did in previous labs.

**Note:** K= 8.





# Contd.

## Float To UChar:

To change the format of data stream from float to Uchar (Byte format)



# Contd.

**Pack K Bits**; Converts a vector of bytes with 1 bit in the LSB to a byte with k relevant bits.

After Processing the data in bit format. To read whether we performed modulation and demodulation correctly or not we need to pack this bits format into a byte again. And for doing so we will be using **"Pack K Bits"**

## Note:

Input and Output of this block will be in Byte format only.

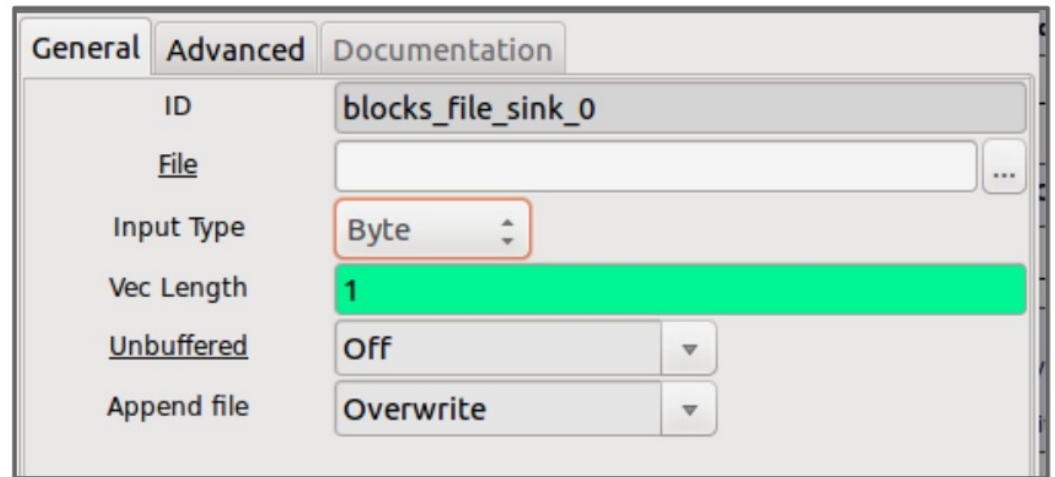
K=8.



# Contd.

**File Sink:** Outputs raw data values in binary format to the specified file.

**Note :**  
Save file with .txt format



# Contd.

## Skip Head:

Skips the first N items, from then on copies items to the output.

Useful when there are metadata or junk at the start.

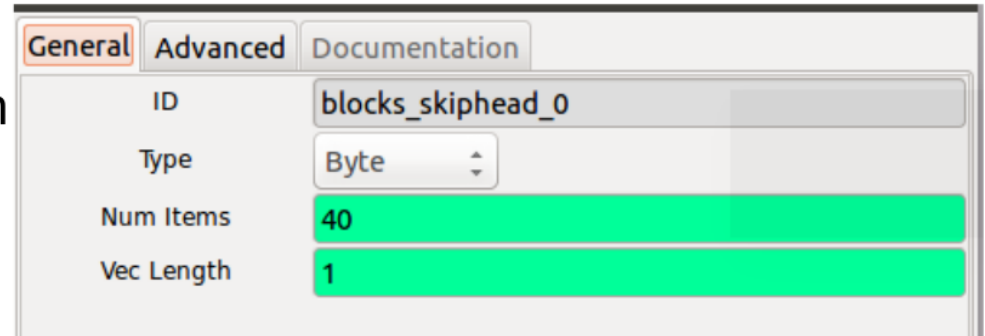
The Processing in GNU Radio adds some junk value at the start,

If we don't use this then packing bits in byte format will cause wrong packing of ASCII code.

## Note:

Since number of junk bits added depends on your flow and on modulation scheme so you need to do iterative trial to get the correct value.

Ask T.A's for Help.



# Task 2

- Task 1 assumed you have perfect phase sync between the transmitter and receiver. But what if you had a  $180^\circ$  phase shift between the txmitter and receiver? Your constellation diagram would look perfect, but your decoding would be wrong!
- One solution to this problem is differential BPSK – encode information not in the phase value of the output symbol, but in the phase differences
- Task 2 is to repeat the end-to-end transmission of the text file using differential BPSK signalling