Introduction to Communication Systems Using National Instruments Universal Software Radio Peripheral Lab Manual

Authors: Anees Abrol and Eric Hamke
Contributors: Carlos Adrian Martinez Yero ,
Olajide Durosinmi, Jeffery Love
Editor: Dr. Ramiro Jordan

THE UNIVERSITY of NEW MEXICO

istec
Ibero-American Science & Technology Education Consortium

GINET
Global Innovation Network for Entrepreneurship and Technology

Draft: September 11, 2014

Table of Contents

List of Figures

List of Tables

1 Introduction

The students at the University of New Mexico Electrical and Computer Engineering Department are planning to use an integrated set of lectures and labs to better understand basic communications systems. The lectures are based on the textbook by Ziemer and Tranter, Principles of Communications - Systems, Modulation, and Noise. The labs are developed using the National Instruments *Universal Software Radio Peripheral (USRP). The choice of this radio provides 2 advantages from an instructional perspective: it minimizes the amount of lab equipment necessary for performing the labs, and its range of flexibility to support spectrum sensing, cognitive radio and alternate modulation schemes.*

The labs are written with the idea that the students not only need to make measurements, but also learn how to write their own Virtual Instruments (VIs) supporting the modulation schemes being studied. The first lab will include a demo where the USRPs are used to transmit and receive. The students will have a chance to setup the radios and gain experience with USRP interface Input/Output. The review will build on this experience to familiarize them with the standard lab interfaces used in the training. The review will culminate with having them modify and use the Spectrum Monitoring example provided with the LabVIEW USRP toolkit.

The other labs will build around a common interface and will have the students write their own modulation VIs. The students will then integrate their Modulation VIs into the standard interface that was developed or discussed in the review labs. The integration process will help the students become more advanced users as they need to plan and debug their application. The work involved in the resulting series of labs will be of increasing levels of difficulty. This approach is to ensure that the students do not spend excessive amounts of time on the labs and not learn the theory in class. The initial level will be structured such that the students are given an almost working AM modulation module and have to just debug it to get it to work. Essentially, using a working example, they would need to figure out what is missing and add it in. The next lab will have more elements missing and more discussion of what needs to be done. This will progress until the final lab where they will be given a description of what needs to be done, and they compose the final VI to accomplish the desire objective.

1.1 Syllabus

The following is an overview of the course and its labs. The Class Lecture Topics in the syllabus were developed by Dr. Sudharman K. Jayaweera, and are summarized in the following table. The supporting lab exercises and lectures are presented in the Lab Topics of the table. The main effort is synchronizing the material in the lab with the material being introduced in class. In later labs, the emphasis is based on reinforcing or making the mathematical abstractions discussed in lecture more concrete. Our experience is that the Engineering students learn by actually making things work (many students learn in different ways). This practical kind of knowledge is what we hope will make good the students into thinking and productive individuals capable of handling new and diverse assignments.

Table I – Training Course Overview

Week	Class Lecture Topic	Lab Topics
1.	Review and introduction of signal models and classifications	**Goal:** This is an introductory lab exercise to: • Ensure that students have a working installation of LabVIEW on their computers. • Train the students to setup the USRP and gain experience with Radio interface Input/Output. • *Gain* familiarity with the standard lab interfaces used in the training and the desired coding practices.
2.	Frequency-domain characterization of (periodic) signals (Fourier series analysis)	**Goals:** This lab will discuss fourier transform in communications and its relationship with to an observed spectrum. The students will: • Observe the amplitude spectrum of a given signal, and then modify the time domain signal parameters to observe the resulting impact in the frequency domain. • Learn how to use the USRP configured as a spectrum analyzer.
3.	Power spectral density (PSD), correlation Signals and linear systems	**Goal**: LabView training needed for the next set of weeks to include: • Learning to write and debug a simple LabVIEW Virtual Instrument in a group-lead exercise. • Familiarization with the USRP sub-VI functions for configuring/managing, transmitting data to, and receiving data from the radio.
4.	Bandwidth, sampling, DFT Hilbert transform, analytic signals, complex envelope representations of band-pass signals and band-pass systems	**Goal:** This lab will introduce the LabVIEW filter design toolkit and its use. The students will: • Observe the power spectrum of a given signal (or combination of signals) • Design a series of digital filters that will allow the isolatation and analysis of a specific tone from a linear combination of tones.
5.	Linear modulation (DSB, AM, SSB) Linear modulation (continued). (DSB, AM, SSB)	**Goal:** This lab focuses on: • Learning to write a VI that uses the USRP as an amplitude modulation (AM) transmitter and a receiver. • Designing and implementing envelope detectors for AM signals using LabVIEW Signal Processing Filter sub-VIs.
6.	Angle modulation (PM and FM), Narrowband angle modulation, power efficiency Angle modulation (PM and FM), Narrowband angle modulation, Demodulation of angle-modulated signals, PLL.	**Goal:** This lab focuses on: • Learning to write a VI that uses the USRP as an frequency modulation (FM) transmitter and a receiver. • Investigating demodulation of an FM signal in software which is much simpler than demodulation procedures in the traditional hardware approach.
7.	Analog Pulse modulation (PAM, PWM, PPM), digital pulse	**Goal:** This lab will explore the use of Pulse-Position Modulation (PPM) by learning to implement a PPM

Table I – Training Course Overview

Week	Class Lecture Topic	Lab Topics
	modulation (PCM, Delta modulation), multiplexing	transmitter/receiver.
	Probability, random variables, PDF's.	
8.	Random processes, correlation, PSD	**Goal:** This lab provides an example use of Cross-Correlation and Power Spectral Density (PSD) to predict the distance of an object (or target) by comparing the transmitted and reflected signal in the context of an ideal and a noisy RADAR environment.
9.	Semester Break	
10.	Review for Mid=Term	
11.	Noise in linear systems, narrowband noise, quadrature components of narrowband noise.	**Goal:** This laboratory exercise has two objectives: • Implementing an Additive Gaussian White Noise source in LabView. • Investigating the effects of noise on AM signal envelope detection.
	Noise sources, noise figure, narrowband noise, quadrature components of narrowband noise	
	SNR, AM and noise (coherent and envelope demodulations)	
12.	Noise in Angle modulation systems, FM and noise (above threshold operation)	**Goal:** This laboratory exercise has two objectives: • Investigating the effects of Additive Gaussian White Noise on FM signal envelope detection. • Comparison of FM to AM results from previous lab.
	Digital communication systems, Analog-to-digital conversion, sampling, quantization, compression, binary digital modulation and demodulation, probability of error analysis of integrate-and-dump receiver in AWGN	
13.	Binary data transmission, binary signal detection, Likelihood ratio (LR) detector, MAP and ML detectors	**Goal:** In this lab, the students will design a serial interface by developing a Universal Asynchronous Receive Transmit (UART) receiver VI. The UART will accept a source string: • Encoded using the American Standard Code for Information Interchange (ASCII). • With replicated bits used to minimize bit error rates. Additionally, the lab will include designing state machines using LabVIEW.
	Binary data transmission with arbitrary signal shapes, matched filter receiver, optimality of matched filter in non-Gaussian noise	
14.	Coherent detection of binary signals in digital communication systems: BPSK, OOK, BFSK, and probability of error analysis	**Goal:** In this lab, a bit stream is transmitted and received using Binary Phase Key Shifting (BPSK) as the modulation technique by: • Using the UART transmitter and receiver from the previous lab to generate the bit stream. • Using a binary phase mapping to convert bits into phase values for an FM transmitter.
	Non-coherent modulation schemes and non-coherent detection of binary digital signals: DPSK and non-coherent BFSK,	

Table I – Training Course Overview

Week	Class Lecture Topic	Lab Topics
	probability of error analysis	
15.	Digital communication over band-limited channels, Baseband modulation, Line-codes and their spectra Inter-symbol interference, Nyquist criterion for zero-ISI, Nyquist bandwidth, raised cosine pulses, Equalization, zero-forcing equalizer	**Goal:** This lab to develop an understanding of entropy coding by investigating Huffman coding algorithm: • Understanding the mechanisms and importance of source coding and data compression. • Investigating the efficiency of Huffman Coding • Implementation of Huffman Coding algorithm in LabVIEW
16.	M-ary modulation, signal space concept Spread Spectrum, Multicarrier modulation, and OFDM	**Goal:** The purpose of this lab is to: • Introduce the concept of frequency-division multiplexing. • Modulating two messages on separate sub-carriers. • Explore the concept of intermediate-frequency filtering in the receiver.
17.		Project Presentations: Best projects selected as new labs for next year. All projects will be added to course content library.

1.2 Structure of Labs

1.2.1 Lab Equiopment

A typical lab configuration will include:

1) One desktop computer with LabVIEW Student Developer License and access to the USRP driver and Modulation Toolkit.
2) One data switch to support interfacing with the radios at 1 gigabit per second and an Ethernet interface running at 100 megabit per second.
3) 2 USRP radios.

1.2.2 Format of Labs

1. Summary

This section should be a paragraph describing learning objectives and a list of learning tasks and activities. The paragraph is meant to introduce the learning objectives. These are meant to introduce the tasks or activities that the students are intended to accomplish in the lab. The list is composed of simple sentences or phrases.

2. Background

This section should briefly summarize material presented during lecture with additional material needed to support making the observations needed to support the **Objectives** section's tasks or activities. Note that the observation techniques are introduced here but they are explained in more detail in the **Lab Procedure** section.

Additional subsections deal with configuring the USRP. With a detailed description of creation of the transmitter/receiver VI needed to perform the lab.

3. Pre-Lab

This would include any preliminary work that you might expect the students to have accomplished prior to the lab. The preliminary work should include any programming assignments needed to support the lab. It is important that you stress that this needs to be completed prior to the lab since no time has been allocated to do this in the lab.

Transmitter
A template for the transmitter will be provided. This template contains the four interface VIs described in the **Background** section along with a "message generator" that is set to produce a message signal. Your task is to add blocks as needed to produce the modulated signal needed for the lab, and then to pass the modulated signal into the *while* loop to the Write Tx Data block. The modulation index is to be user-settable in the range $\boldsymbol{0 \leq \mu \leq 1}$, and a front-panel control will be provided.

Receiver
A template for the receiver will also be provided. This template contains the six interface VIs described in the **Background** section above along with a waveform graph on which to display your demodulated output signal.

4. Lab Procedure

This section provides instructions for making specific observations needed to complete the tasks or activities introduced in the **Objectives** section. These instructions need to be very specific, providing details as to how to configure the VIs used in the observations, and also instructions for recording observations such as getting a screen capture, saving data off to a file for post processing, and lab notebook entries. The Observation step should have simple direct questions that when answered will document the lab's execution.

5. Lab Write-up

This section should include directions for preparing the Lab write-up, including a secondary set of questions that ask the student:

1) To describe what he/she learned in terms of the tasks or activities introduced in the **Objectives** section.
2) To explain what could have possibly gone wrong and the consequences of these problems.
3) How to improve the lab.

2 An Introduction to Digital Communications Lab

2.1 Summary

The course will focus on applying the theory discussed in class. You will be using a software defined radio (SDR) that implements the algorithms necessary for digital communications. In this lab, you will be designing and implementing SDR applications using National Instruments' *Universal Software Radio Peripheral (*USRP) hardware and National Instruments' LabVIEW. Through the implementation of this SDR you will investigate practical design issues in digital communications.

In this particular lab, you will setup the USRPs and run the provided lab Visual Instruments (2.2.3.1). It is important that you follow lab safety procedures to avoid damaging the equipment. The radios are sensitive and can burn out if not used correctly.

This part of the lab will answer the following questions:

- How does USRP work?
- What is NI LabVIEW and why is it used in this lab?
- Is there such a thing as good code style in LabVIEW?

2.2 Background

Overall, this lab is meant to introduce the basic functioning of the USRP, and LabVIEW as a tool to design digital communication systems in this course.

2.2.1 Software and Other Materials for the Course

Throughout the course you will be required to do some preparatory work (e.g., pre-labs) before entering the lab itself. In order to complete those assignments, you must have access to the following National Instruments software tools and packages outside of class.

- NI LabVIEW 2012 or later
- NI LabVIEW Modulation Toolkit 4.3.1 or later

Note: Your instructor will provide information about how to access LabVIEW at your University.

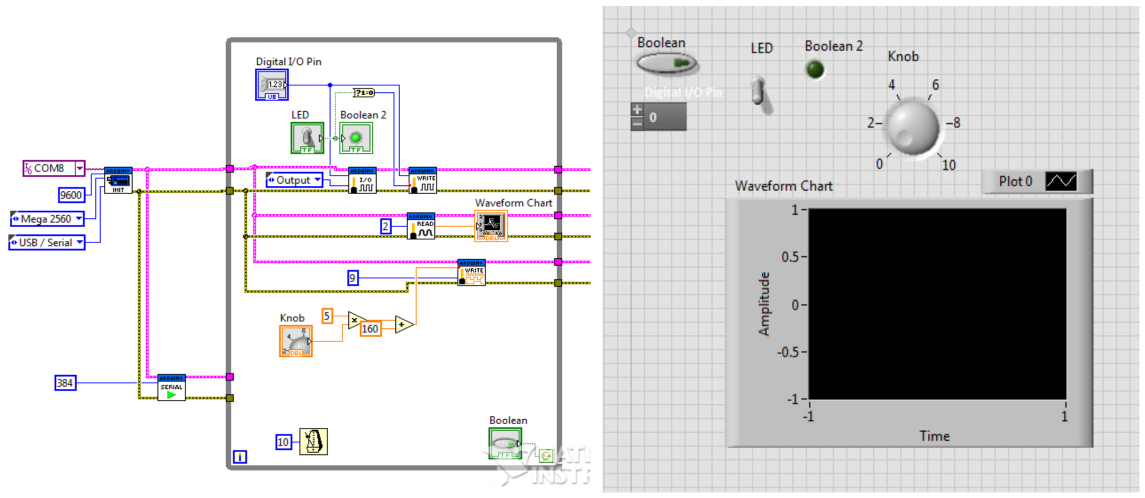
2.2.2 Introduction to NI USRP-2920

NI USRP (Universal Software Radio Peripheral) is a flexible software defined radio that turns a standard personal computer into a high-performance wireless prototyping platform. Paired with NI LabVIEW software, NI USRP transceivers provide a powerful system to help you learn and program quickly. The NI USRP-292x transceivers used in this course labs are adequate for hands-on laboratory learning in the field of RF and communications. With the USRPs and LabVIEW software, you will have the opportunity of experimenting with real-world signals in digital communications laboratories. With this solution, you may focus on the actual implementation of algorithms and related real-world impairments. More product information and specifications can be found on the National Instruments website.

2.2.3 Introduction to National Instruments LabVIEW

2.2.3.1 What Is LabVIEW?

LabVIEW is a graphical programming language developed by National Instruments. The basic building block of LabVIEW is the virtual instrument (VI). Conceptually, a VI is analogous to a procedure or function in conventional programming languages. Each VI consists of a *block diagram* and a *front panel* (Fig. 1.). The block diagram describes the functionality of the VI, while the front panel is a top level interface to the VI.



Block Diagram *Front Panel*

Fig. 1: LabVIEW VI Interfaces

2.2.3.2 LabVIEW Environment VIs

Many of the algorithms implemented in this lab (and in digital communications in general) use linear algebra. LabVIEW provides support for matrix and vector manipulation, and linear algebra, with VIs for functions like matrix inversion (*Inverse Matrix.vi*), matrix multiplication (*A x B.vi*), and reshaping arrays (*Reshape Array.vi*). LabVIEW also has many built-in signal processing functions, such as the fast Fourier transform (*FFT.vi*), inverse fast Fourier transform (*Inverse FFT.vi*), and convolution (*Convolution.vi*). Additionally, the Modulation Toolkit is a toolset of common digital communication algorithms which will also be leveraged in the lab. Note that many of the functions you will implement in this lab are already available in the Modulation Toolkit in some form. The objective of this course is to understand the principles of wireless digital communication by implementing the physical layer in as much detail as possible. Once familiar with these concepts, you will be able to decide when to use existing VIs and when to write your own. It is recommended that you explore tool palettes such as the (1) Signal Processing palette, (2) Digital palette (part of the Modulation Toolkit palette), (3) Structures palette, (4) Complex palette (part of the Numeric palette), and (5) Array palette in order to acquaint yourself the VIs likely to be used in this course.

Appendix B enumerates some common VIs and highlights how to access them.

For this introductory lab, you will use only existing VIs, tailored specifically to the course material. The intention is to allow students to focus on the lab theory. These VIs have been rigorously tested checked out and should not be a source of error when checking lab results. The VIs for this lab are divided into two categories- Transmitters and Receivers- and have been included in Appendix B for ease of reference.

2.2.3.3 Coding Style in LabVIEW

Pay close attention to the code used to create the VIs for this lab. In future labs you will be asked to write your own VIs, and the provided algorithms are excellent examples of the preferred coding styles. Remember that the coding style should exhibit the following qualities:

1. Consistency - All code should follow the same coding style at all times.
2. Readability -
    - The code should be organized in a modular fashion that promotes reuse and maintenance.
    - Connections (signals or lines) should be laid out with the least number of overlapping lines, and a signal should appear to enter a block at only one point.

- Connections should appear to enter or leave only the blocks being used. A connection should not be overlapped by a block that is not the starting point or ending point.

3. Documentation - Documentation should not be few and far between. Good documentation includes not only free text labels (see the Decorations palette, which is part of the Structures palette), but also descriptive variable, block, and sub-VI names."

The design of a digital communication system is modular and sequential in nature. It therefore makes sense to adopt a coding style which makes use of modularity and hierarchy. Since many algorithms in digital communications can often be boiled down to a recipe (*e.g., A then B then C*), it is critical that you use sub-VIs whenever possible.

2.3 Pre-Lab

Prior to beginning the in-class portion of this lab, you will be expected to install LabVIEW on your laptop and familiarize yourself with its features using the information provided in Appendix A. The following tutorials and reference material will help guide students through the process of learning LabVIEW:

- *LabVIEW 101* [5];
- *LabVIEW Fundamentals* from National Instruments [6];
- Online LabVIEW tutorials from NI[3], [4].

New LabVIEW programmers should carefully review all of the material in [5] and [4]. Please remember to refer to [6] and [4] often as they are excellent references for all basic LabVIEW questions. Bring any questions or concerns regarding LabVIEW or these tutorials to your instructor's attention.

2.4 Lab Procedure

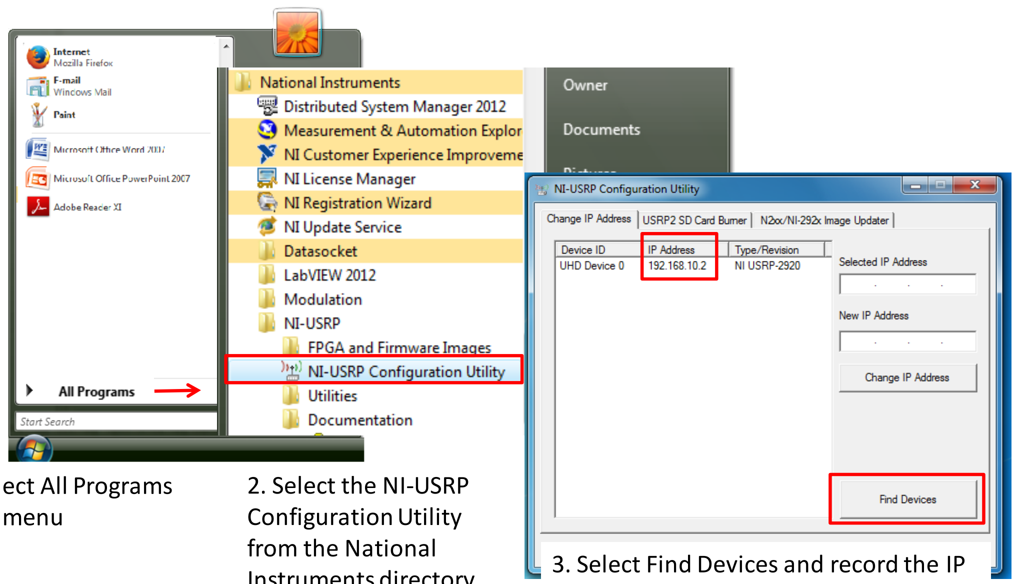
2.4.1 Global set-up

Each workstation will have two USRP radios, one for transmission and the other one for reception. There are two possible configurations:

a) Both radios connected to single computer (running both the transmitter and receiver VIs) with a dual-port Gigabit Ethernet interface, or using a data switch.
b) Each radio connected to a different computer; one running the transmitter VI, and the other, the receiver VI.

2.5 Set-up steps

1. Connect the computer to the USRP using an Ethernet cable.
2. Open the NI-USRP Configuration Utility found in the National Instruments directory under programs files as shown in Fig. 2. Be sure to record the IP addresses since you will need them to configure your software.



Device ID	IP Address	Type/Revision
UHD Device 0	192.168.10.2	NI USRP-2920

1. Select All Programs from menu

2. Select the NI-USRP Configuration Utility from the National Instruments directory

3. Select Find Devices and record the IP address of the radio or radios since you will need them to configure the software in the lab.

Fig. 2: Finding the IP Address Radio Connectivity Test

If the IP address does not appear in the window then check your connections and ask the Teaching Assistant (TA) to verify that the LAN card has been configured correctly.

2.5.1 VIs to be used in the lab:

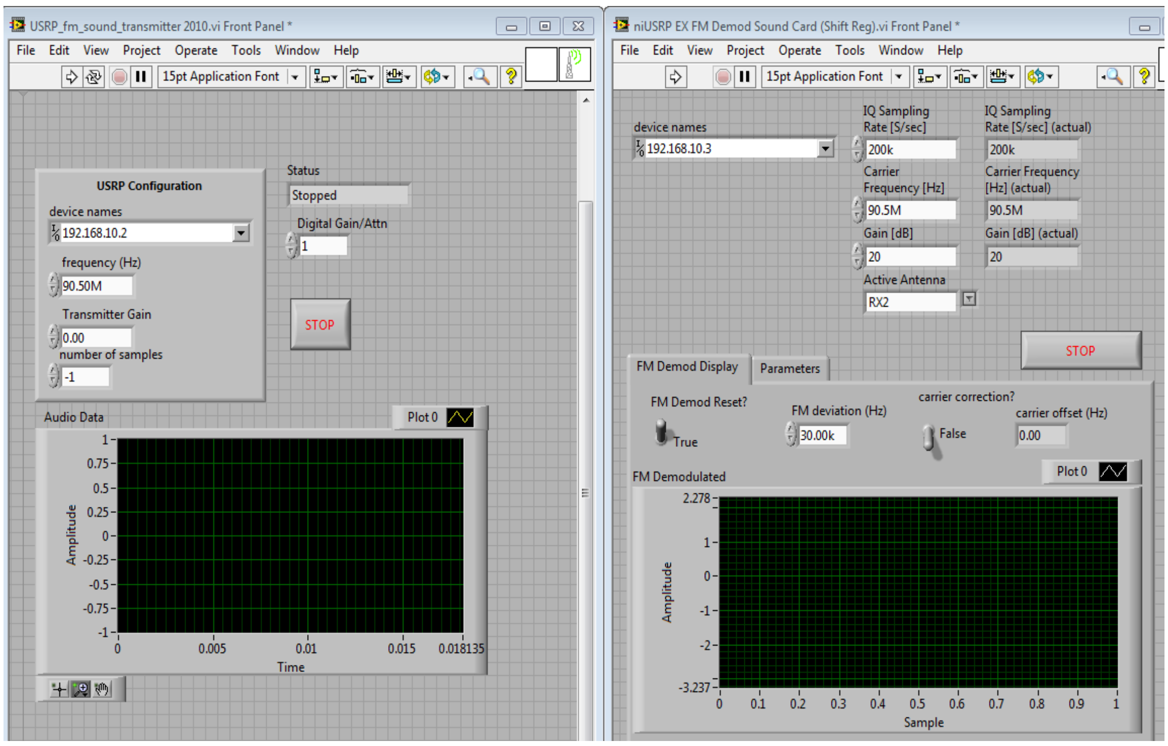
The following VIs, transmitter (Tx) and receiver (Rx), will be used on each of the lab's exercises.

2.5.2 Tx VI:

The "USRP_fm_sound_transmitter 2010.vi" (Tx VI) provided with the Lab1 material (also available at https://decibel.ni.com/content/docs/DOC-25893) which implements an FM transmitter that will transmit the contents of a wav file will be used as the Tx VI (Fig. 3.).

2.5.3 Rx VI:

Choose one from the two FM Demod VI examples provided with LabVIEW to be used as the Rx VI for this lab by following the path: Start Menu → All Programs → National Instruments → NI-USRP → Examples → LabVIEW → ModulationToolkitExamples.


USRP_fm_sound_transmitter 2010.vi Front Panel *

USRP Configuration
device names: 192.168.10.2
frequency (Hz): 90.50M
Transmitter Gain: 0.00
number of samples: -1
Status: Stopped
Digital Gain/Attn: 1

niUSRP EX FM Demod Sound Card (Shift Reg).vi Front Panel *

device names: 192.168.10.3
IQ Sampling Rate [S/sec]: 200k
IQ Sampling Rate [S/sec] (actual): 200k
Carrier Frequency [Hz]: 90.5M
Carrier Frequency [Hz] (actual): 90.5M
Gain [dB]: 20
Gain [dB] (actual): 20
Active Antenna: RX2
FM Demod Reset?: True
FM deviation (Hz): 30.00k
carrier correction?: False
carrier offset (Hz): 0.00

Fig. 3: Appropriate Field Setup In TX And RX VIs

Important set-up notes:

- ✓ Determine the IP addresses for your radios using the NI-USRP configuration utility.
- ✓ Make sure the Tx and Rx VIs are always set to the same carrier frequency whenever you pair them up to communicate (see Fig. 3.).
- ✓ Transmission should start only after receiving workstations are ready to receive.
- ✓ Verify that device name fields in both Tx and Rx VIs are set to the IP address of the URSP in use (see fig 3).
- ✓ Make sure to connect the provided attenuator between the receiver USRP’s Rx input and the antenna/loopback-cable. The attenuator is used to decrease the power level of the transmitted signal in order to avoid a high power signal at the receiver’s end, due to Rx and Tx inputs’ proximity to each other.

2.5.4 Broadcast Mode

The TA will be leading/moderating this exercise, and will set one of the USRPs in the classroom as the transmitter. All the students in their workstation will set one of their respective radios to receive at a given frequency, which will be announced by the TA.

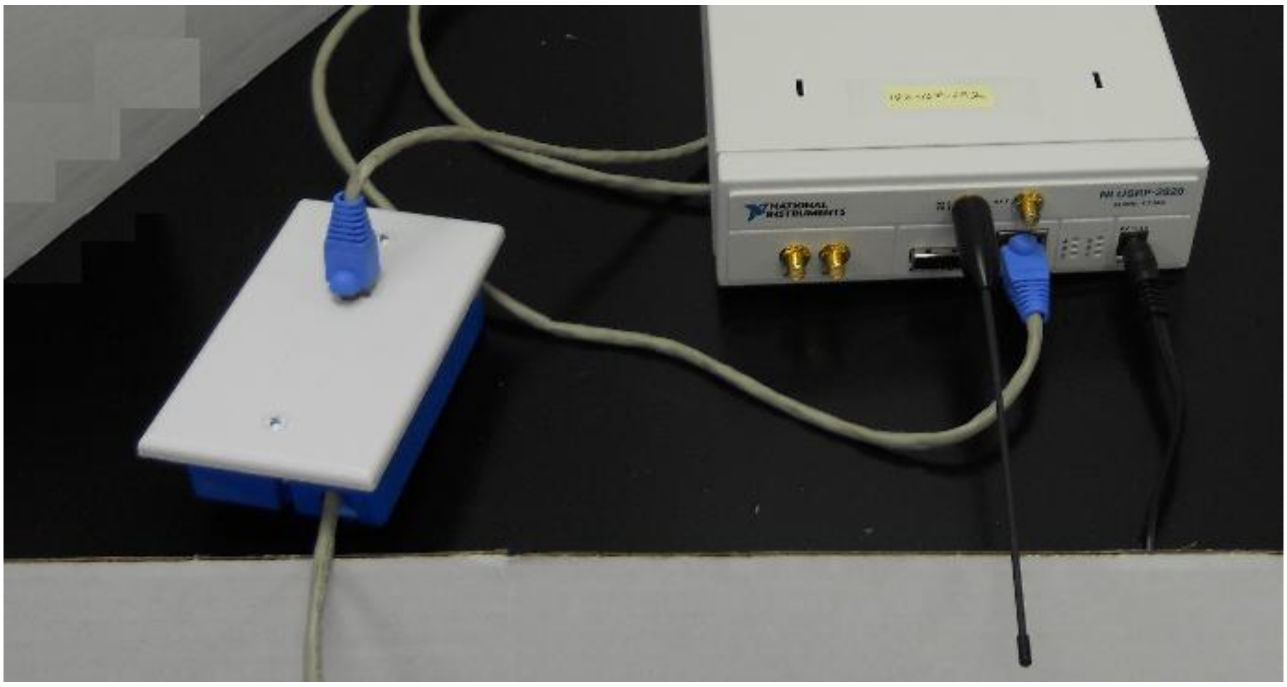


Fig. 4: Broadcast Setup

2.5.5 Students' receiver set-up

1. Connect the VERT400 Antenna (Fig. 5) to the RX1/TX1 (Fig. 6) or RX2 input (Fig. 6) on one of the two radios connected to the workstation. Remember to connect the provided attenuator between USRP's Rx input and antenna.

Fig. 5: VERT400 Antenna

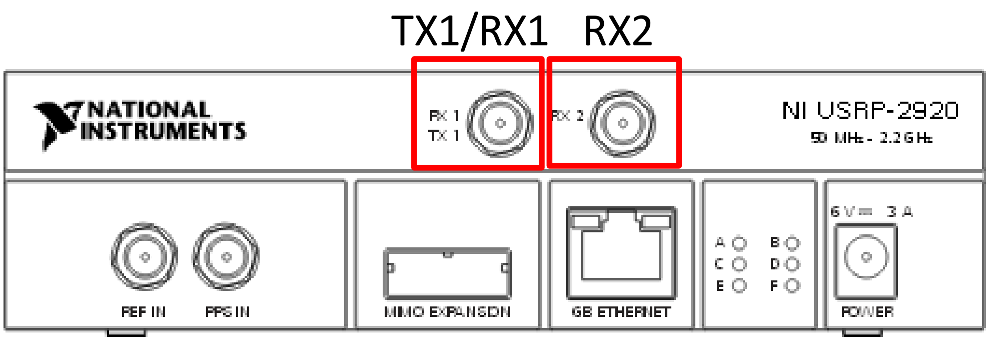


Fig. 6: USRP Front Panel

2. Connect a set of speakers to the computer to listen to the audio file transmitted from the TA's station.
3. Open the Rx VI.
4. Set the '*Carrier Frequency'* on front panel to match TA's Tx frequency.
5. Set '*Active Antenna*' on the front panel to the right input according to the actual USRP Rx being used (connected to the antenna).
6. Save this modified VI.

7. Run the Rx VI when the TA indicates that the transmitter is broadcasting. Once the TA starts transmitting, the contents of the wav file should be audible on the computer's speakers.

2.5.6 Point to point Wireless (P2PW)

In this section we will get to check the effect of channel interference by broadcasting over a frequency that is already in use. You will use both USRPs, one for transmission and the other one for reception.

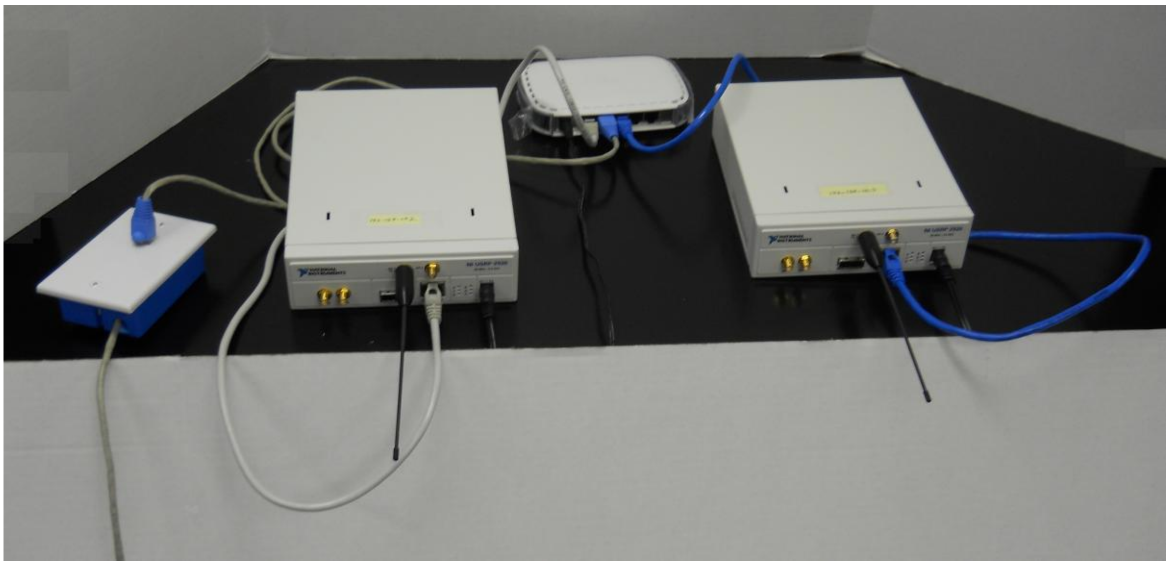


Fig. 7: P2PW Wireless Setup

1. Connect the VERT400 Antenna to the RX1 (or RX2) terminal (Fig. 6) on one radio and TX1 terminal (Fig. 6) on the other. Remember to connect the provided attenuator between the USRP's Rx input and the antenna.
2. Connect a set of speakers to the computer.
3. Set the '*Carrier frequency'* on the front panel to the frequency of a local radio station (e.g. 101.3 FM). Note: For this section to work you must have an USRP 292x series radio.
4. Set '*Active Antenna'* on the front panel according to the actual USRP Rx input being used (connected to the antenna).
5. Run the Rx VI to start listening to the tuned radio station.
6. Open the Tx VI.
7. Set the frequency on this VI's front panel to the same frequency you are receiving.
8. Run the Tx VI with an appropriate wav file.

The transmitted wav file should be heard over the radio channel. This is because the signal from the radio station is being overpowered by the stronger signal transmitted from the USRP over the same frequency channel.

2.5.7 Loop-back (Cable Carrier Example)

In this section, instead of the antennas, we use a loopback cable (SMA-M-to-SMA-M cable) as the transmission media.

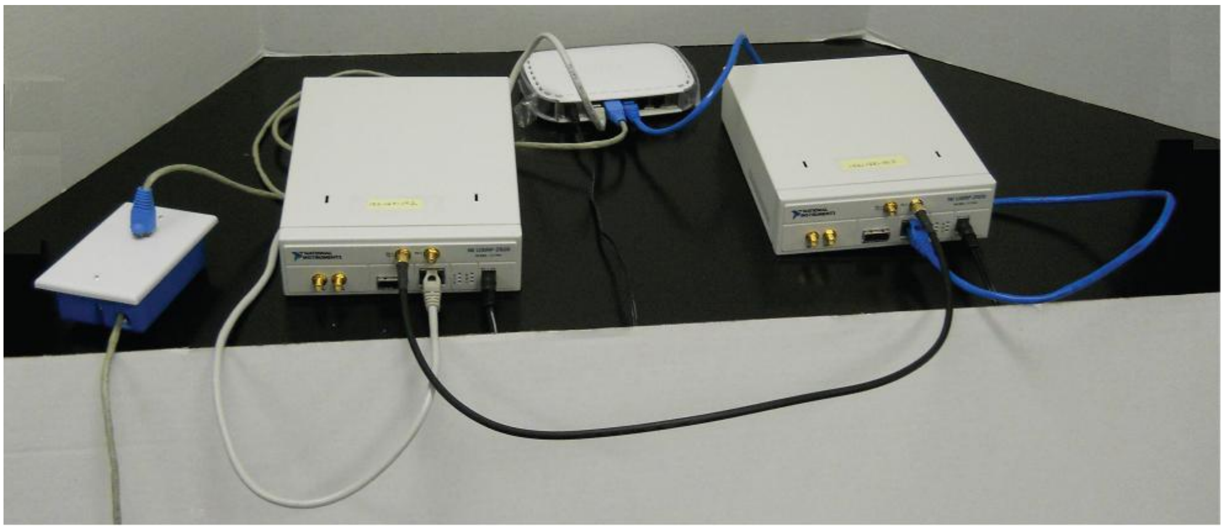


Fig. 8: Loopback Setup Picture

1. If still attached, detach the VERT400 vertical antennas from the USRPs.
2. Connect one end of the loop-back cable to the RX1/TX1 (or RX2) terminal (Fig. 6) on one radio and RX1/TX1 terminal (Fig. 6) on the other. Remember to connect the provided attenuator between the USRP's Rx input and the loop-back cable.
3. Connect a set of speakers to the computer.
4. Verify that both the transmitter and receiver are set to the same carrier frequency.
5. Set '*Active Antenna*' on the front panel according to the actual USRP Rx input being used (connected to the loopback cable).
6. Open and run the Rx VI.
7. Set the frequency on this VI's front panel to the same frequency you are receiving.
8. Open and run the Tx VI with an appropriate wav file.

The transmitted wav file should be audible on the computer's speakers.

2.6 TA Notes for Section 3.1 TA's Transmitter Set-up

1. Connect the VERT400 Antenna (provided with the USRP radios) to the "RX1/TX1" terminal on the radio.
2. Open the Tx VI.
3. Set the frequency on the VI's front panel to a suitable value according to the used antenna (VERT400 Tri-band omni-directional vertical antenna provided with the USRP).
4. Announce this frequency to the students.
5. Wait till all students run their Rx VIs.
6. The TA will run the VI with an appropriate wav file.

2.7 Lab Writeup

Performance Checklist

Introduction to Digital Communications

Short Answer Questions

1. What is the USRP? What is LabView?

2. What is a VI? Why do we use two VIs in this lab?

3. Explain "Broadcast", "Point to Point (P2P) wireless", and "Loop-back cable" transmissions.

Performance Measures

Task	Standards	Sat/Unsat
Hardware Setup	Working setup for all 3 configurations: Broadcast, P2P wireless, and Loop-back cable.	
Running VIs	Successful transmission and reception of audio files for all 3 configurations: Broadcast, Point to point wireless, and Loopback-cable.	
QOS in P2P wireless configuration	Quality of received signal heard over the radio channel in point to point wireless configuration.	

Discussion

Did all configurations perform as expected?
Did you have any difficulties completing the lab?
Did your TA provide enough guidance?
Do you have any recommendations to improve the lab?

2.8 References

[1] National Instruments, NI USRP-2920, retrieved August 23, 2014, from http://sine.ni.com/nips/cds/view/p/lang/en/nid/209948.
[2] National Instruments, Device Specifications: NI USRP-2920, retrieved August 23, 2014, from http://www.ni.com/pdf/manuals/375839a.pdf.
[3] National Instruments, Getting Started with NI LabVIEW Student Training, retrieved August 23, 2014, from http://zone.ni.com/devzone/cda/tut/p/id/7466.
[4] National Instruments, How Can I Learn LabVIEW?, retrieved August 23, 2014, from http://www.ni.com/getting-started/labview-basics/.
[5] National Instruments, LabVIEW 101, retrieved August 23, 2014, from http://www.ni.com/lv101.
[6] National Instruments, LabVIEW Fundamentals, retrieved August 23, 2014, from http://www.ni.com/pdf/manuals/374029c.pdf.

3 Frequency-domain Characterization of Signals: A Look at the Fourier Transform

3.1 Summary

This lab will discuss the importance of Fourier Transform and its use in digital communications. You will also receive some additional training in the use of LabVIEW.

The LabVIEW training will consist of a set of lectures focused on writing and debugging an example code. The example will be provided at the start of the lecture and you and your partner are expected to follow along with the lecture. There is no Pre-Lab assignment and the Lab Procedure will be to follow along with the lecture.

The Fourier Transform portion of this lab is divided into two parts:

- The first part of this lab will introduce students to the Fourier Transform. You will write a VI to observe the amplitude spectrum of a given signal, and then modify the time domain signal parameters to observe the resulting impact in the frequency domain.
- The second part will show you how to connect to the USRP radio and use it as a spectrum analyzer. You will use “niUSRP EX Spectral Monitoring (Interactive).vi” to monitor the spectrum, and use it to identify the different frequency components.

3.2 Background

3.2.1 Software and Other Materials for the lab

To complete this lab you will need the following National Instruments software and radio equipment.

- NI LabVIEW 2012 or later
- NI LabVIEW USRP demonstration file “niUSRP EX Spectral Monitoring (Interactive).vi”
- NI LabVIEW USRP radio
- VERT2450 Antenna
- a data switch (optional manages transition from 1 Gigabit to 100 Megabit Ethernet interface)

3.2.2 Introduction to the Fourier Transform

Note: *For a more in-depth discussion see “Signals and Linear Systems Analysis” [1]. You should read his chapter, specifically the FT section, as it will help you to answer some of the write-up questions.*

3.2.2.1 What Is the Fourier Transform?

All waveforms, no matter what you describe or observe in the universe, are actually just the sum of weighted sinusoid waveforms (both sines and cosines) of different frequencies. The representation of a periodic function, or of a function that is defined only on a finite interval as the linear combination of sines and cosines, is known as the Fourier series expansion of the function. Essentially, the Fourier series decomposes such functions into a sum of sinusoids. The Fourier Transform is the extension of this idea, and it is used for obtaining the amplitude and phase information in the frequency domain for sequences and functions which are not periodic in the time domain.

The Fourier Transform is the mathematical tool (1) that deconstructs the waveform into an equivalent representation using sinusoidal components. In other words, the Fourier transform is the frequency-domain representation of a signal $\mathrm{x(t)}$, and it is denoted as $X(f)$. The Fourier transform is a function of the frequency f. $X(f)$is often called the spectrum of $x(t)$. The amplitude spectrum of

X(f), defined as a plot of |X(f)| vs. f, gives how much power $x(t)$ contains at the frequency f, while the phase spectrum, defined as the plot of ∡X(f) vs. f, will give the phase at f.

$$\mathcal{F}\{x(t)\} = X(f) = \int_{-\infty}^{\infty} x(t)e^{-j2\pi ft}dt \tag{1}$$

Conversion back from the frequency domain to the time-domain representation of the signal is then called the Inverse Fourier Transform (2).

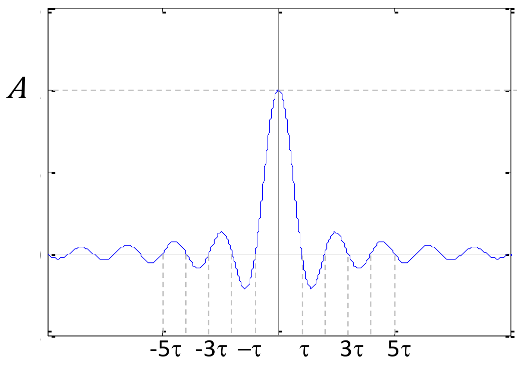
$$\mathcal{F}^{-1}\{X(f)\} = x(t) = \int_{-\infty}^{\infty} X(f)e^{j2\pi ft}df \tag{2}$$

The previous two equations are referred to as the Fourier transform pair. One of the most important of these pairs (Fig. 9) is the rectangle function (Π(f)) and the sinc(t) function. Nyquist in his work identified these as the ideal functions for representing a frequency band-limited/time-sampled signal. The frequency band limiting is obvious from observing the rectangle function. The sinc(t) function has a unique property, its value at t=0 zero is 1, and zero at any non-zero multiple of the sampling period (T)). As discussed in the next section, a signal sampled at regular intervals or sampling period (T) results in a finite sequence of data that can be represented as

$$x(t) = \sum_{n=-\infty}^{\infty} x(nT)sinc\left(\frac{t-nT}{T}\right) \tag{3}$$

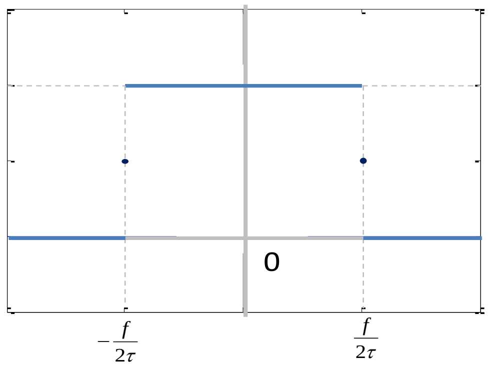
and, in the frequency domain, as

$$X(f) = \Pi(fT)X_s(f) \tag{4}$$



$$A\,\mathrm{sinc}(t) = \frac{A\tau\sin(\pi\tau t)}{\pi\tau t}$$

(a)



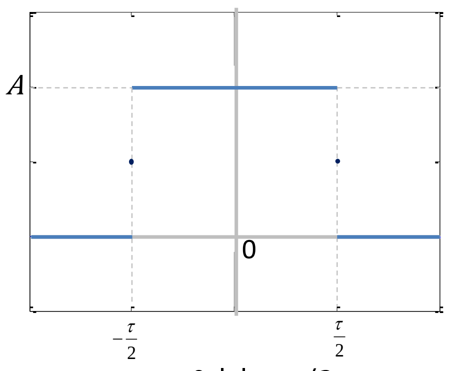
$$\Pi(f) = \begin{cases} 0, |f| > f/2\tau \\ \frac{A}{2}, |f| = f/2\tau \\ A,\ \ |f| < f/2\tau \end{cases}$$

where τ is the pulse width/duration

(b).

Fig. 9: sinc(t) and Rectangle Functions

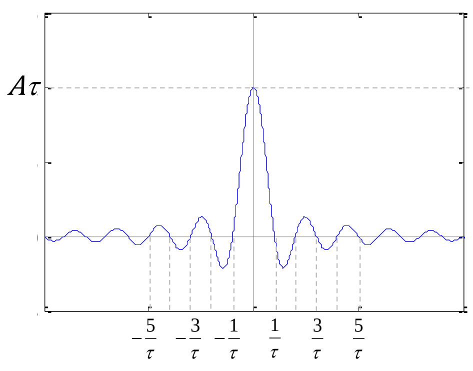
In the lab you will also be looking at the same transform pair, but in a different order. This alternative way of looking at the transform pair is used in the in Orthogonal Frequency Division Multiplexing (OFDM) modulation technique. This allows for a signal to be multiplexed in the time domain, and for each time interval to have its own orthogonal carrier frequency. The orthogonality guarantees that the signals will not interfere with each other.



$$\Pi(t) = \begin{cases} 0, |t| > \tau/2 \\ \frac{A}{2}, |t| = \tau/2 \\ A, \quad |t| < \tau/2 \end{cases}$$

where τ is the pulse width/duration

(a)



$$A\,\tau\,\mathrm{sinc}\,(\pi f \tau) = \frac{A\sin(\pi f \tau)}{\pi f \tau}$$

(b)

Fig. 10: OFDM sinc and Rectangle Functions

The basic idea behind OFDM is the ability to map a window of time in the frequency domain to a specific carrier frequency. In practice it is possible to break a signal up into a sequence of windows the time domain, each having a unique orthogonal carrier frequency. The windows in time are represented as rectangle functions. The time-domain windowing function and its resulting frequency domain function are given by equations (5) and (6), respectively.

$$A\,\tau\,\mathrm{sinc}\,(\pi f \tau) = \frac{A\sin(\pi f \tau)}{\pi f \tau} \tag{5}$$

$$X_m(f) = x(t)\,\tau\,\mathrm{sinc}\,(\pi f \tau)\, e^{j2\pi f m} \tag{6}$$

3.2.2.2 Discrete Fourier Transform

The Discrete Fourier Transform (DFT) is the equivalent of the continuous Fourier Transform for signals sampled at regular intervals or sampling period (T), resulting in a finite sequence of data. The DFT gives an approximation of the Fourier spectrum of the original signal at $\mathrm{f_k} = \mathrm{k/NT}$, where $k = 0, 1, 2, \dots, N-1$. The original signal is truncated to an N number of samples.[3]

A given discrete sequence $x(kT) = x_k$, has the DFT transform:

$$X(f_k) = X_k = \sum_{n=0}^{N-1} x_n e^{-j\frac{2\pi nk}{N}}, \qquad where\ k = 0, 1, 2, \dots, N-1 \tag{7}$$

$$X(f_k) = X_k = \sum_{n=0}^{N-1} x_n W_N^{kn},\ \ where\ W_N^{kn} = e^{-j\frac{2\pi nk}{N}} \tag{8}$$

The inverse DFT is then defined as:

$$x_n = \frac{1}{N}\sum_{k=0}^{N-1} X(k) e^{\frac{j2\pi nk}{N}}\ where\ n = 0,1,2,\dots,N-1 \tag{9}$$

It should be noted that W_N^{kn} for $k = 0, 1, 2, \dots, N-1$ are called the N^th^ *roots of unity*. They are called this because, in complex arithmetic, $\left(W_N^{kn}\right)^N = 1$ for all k. TheW_N^{kn} can be thought of as the vertices of a regular polygon inscribed in the unit circle of the complex plane (Fig. 11), with one vertex at (1; 0). Below are roots of unity for $N = 2$, $N = 4$ and $N = 8$, graphed in the complex plane.[2]

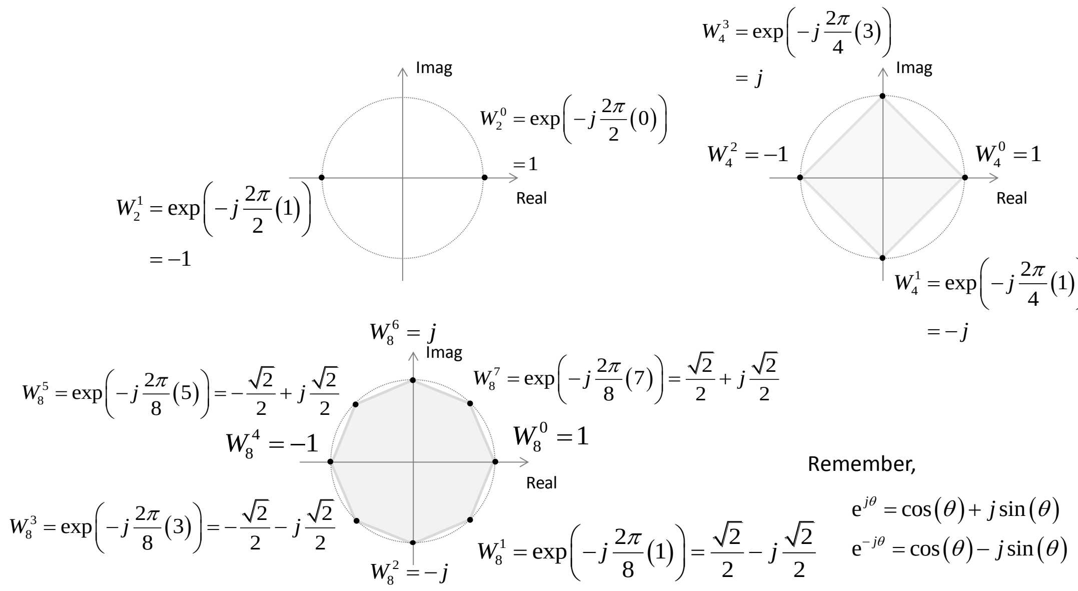

$W_2^0 = \exp\left(-j\frac{2\pi}{2}(0)\right) = 1$
$W_2^1 = \exp\left(-j\frac{2\pi}{2}(1)\right) = -1$
Imag
Real
$W_4^3 = \exp\left(-j\frac{2\pi}{4}(3)\right) = j$
$W_4^2 = -1$
$W_4^0 = 1$
$W_4^1 = \exp\left(-j\frac{2\pi}{4}(1)\right) = -j$
$W_8^6 = j$
$W_8^5 = \exp\left(-j\frac{2\pi}{8}(5)\right) = -\frac{\sqrt{2}}{2} + j\frac{\sqrt{2}}{2}$
$W_8^7 = \exp\left(-j\frac{2\pi}{8}(7)\right) = \frac{\sqrt{2}}{2} + j\frac{\sqrt{2}}{2}$
$W_8^4 = -1$
$W_8^0 = 1$
$W_8^3 = \exp\left(-j\frac{2\pi}{8}(3)\right) = -\frac{\sqrt{2}}{2} - j\frac{\sqrt{2}}{2}$
$W_8^2 = -j$
$W_8^1 = \exp\left(-j\frac{2\pi}{8}(1)\right) = \frac{\sqrt{2}}{2} - j\frac{\sqrt{2}}{2}$
Remember,
$e^{j\theta} = \cos(\theta) + j\sin(\theta)$
$e^{-j\theta} = \cos(\theta) - j\sin(\theta)$

Fig. 11: N^th^ Roots of Unity

Powers of roots of unity are periodic with period N, since the N^th^ roots of unity are points on the complex unit circle every $2\pi/N$ radians apart, and multiplying by W_N is equivalent to rotation clockwise by this angle. Multiplication by W_N^N is rotation by 2π radians, that is, no rotation at all. In general, $W_N^k = W_N^{k+jN}$ for all integer j. Thus, when raising W_N to a power, the exponent can be taken modulo N.[2]

3.2.2.3 Fast Fourier Transform

Before discussing the Fast Fourier Transform (FFT) algorithm, let us explore what is involved in just using the definition give in the preceding section. Suppose we wish to compute the DFT for $N = 2$. So starting with equation (8), we get the following

$$\begin{aligned} X_k &= \sum_{n=0}^{N-1} x_n(-1)^{nk}, \ \ where\ W_2 = e^{-j\pi} = -1 \\ &= x_0(-1)^{0\cdot k} + x_1(-1)^{1\cdot k} \\ &= x_0 + (-1)^k x_1 \end{aligned} \tag{10}$$

$$\begin{aligned} X_0 &= x_0 + x_1 \\ X_1 &= x_0 - x_1 \end{aligned}$$

This is fairly easy, so let us move on to $N = 4$,

$$\begin{aligned} X_k &= \sum_{n=0}^{N-1} x_n(-j)^{nk}, \ \ where\ W_4 = e^{-j\frac{\pi}{2}} = -j \\ &= x_0(-j)^{0\cdot k} + x_1(-j)^{1\cdot k} + x_2(-j)^{2\cdot k} + x_3(-j)^{3\cdot k} \\ &= x_0 + (-j)^k x_1 + (-j)^{2k} x_2 + (-j)^{3k} x_3 \end{aligned} \tag{11}$$

$$X_0 = x_0 + x_1 + x_2 + x_3 \quad \rightarrow \quad X_0 = (x_0 + x_2) + (x_1 + x_3)$$

$$X_1 = x_0 - jx_1 + x_2 + jx_3 \rightarrow X_1 = (x_0 - x_2) - j(x_1 - x_3)$$
$$X_2 = x_0 - x_1 + x_2 - x_3 \rightarrow X_2 = (x_0 + x_2) - (x_1 + x_3)$$
$$X_3 = x_0 + jx_1 - x_2 + jx_3 \rightarrow X_3 = (x_0 - x_2) + j(x_1 - x_3)$$

Notice that in the second part of the above calculations that we can perform the calculations by first determining $(x_0 + x_2)$, $(x_0 - x_2)$, $(x_1 + x_3)$, and $(x_1 - x_3)$. Next we can combine these as shown to solve for X_0, X_1, X_2 and X_4. This process is faster than evaluating the formulas one at a time because we would compute $(x_0 + x_2)$, $(x_0 - x_2)$, $(x_1 + x_3)$ and $(x_1 - x_3)$ twice.

The Fast Fourier Transform (FFT) is an algorithm to compute the DFT and its inverse by taking advantage of this computational savings. The computation is faster than computing the DFT using the definition, especially for the large data sets with which you will be working with in this lab. The FFT utilizes properties of the DFT summation to reduce the overall computations by computing elements of the summation only once and then reusing them as needed. The key to the FFT involves realizing that what can be broken into two parts: a sum over the even-numbered indices $(2m)$ and a sum over the odd-numbered indices $(2m + 1)$, and further realizing that the even summation and the odd summation have roughly the same form as shown in (12).

$$X_k = \sum_{m=0}^{N/2} x_{2m} W_N^{(2m)k} + W_N^k \sum_{m=0}^{N/2+1} x_{2m+1} W_N^{(2m)k} \tag{12}$$

The inverse FFT is given as

$$x_n = \frac{1}{N}\left[\sum_{m=0}^{N/2} X(2m) W_N^{-(2m)k} + W_N^k \sum_{m=0}^{N/2+1} X(2m+1) W_N^{-(2m)k}\right] \tag{13}$$

So using the algorithm computes the exponential terms only once and then uses them as needed. The data flow for the FFT is shown in Fig. 12. The E [ ] terms are the even terms and the O [ ] terms are the odd terms from above.

The FFT will result in a spectrum of N points. The spectrum sampling interval is given as

$$\Delta f = \frac{2f_s}{N} \tag{14}$$

The range of frequency values is $[-f_s,\ f_s]$. So the center frequency (f_0) is located at the $[N/2]$ index of the FFT output.

3.3 Pre-Lab

In addition to reviewing the theory behind the Fourier Transform, prior to this lab you should look at "The Fourier Transform" **Error! Reference source not found.**, and get familiarized with the concepts escribed there.

3.4 Lab Procedure

3.4.1 FFT

In this activity, you we will use LabVIEW to compute the FFT of a given signal, e.g. a square window function. You'll modify the time domain parameters, in particular the time duration (or signal width) of a pulse waveform, and observe the impact of this change in the frequency domain representation of the signal.

For this exercise you will be given an existing VI to modify for the second part of the exercise.

Note: *The FFT is more efficient with a size (number of samples taken from the original time domain signal) power of 2, hence, you are going to use such a value in your LabVIEW VIs.*

3.4.2 Worksheet: The Effect of Varying the Pulse Duration

1. Open "***fft_demo.vi***".
2. Set the FFT size of 512 samples (as shown in Fig. 12).
3. Set the pulse width (τ) (Fig. 12) to the first value in Table II.

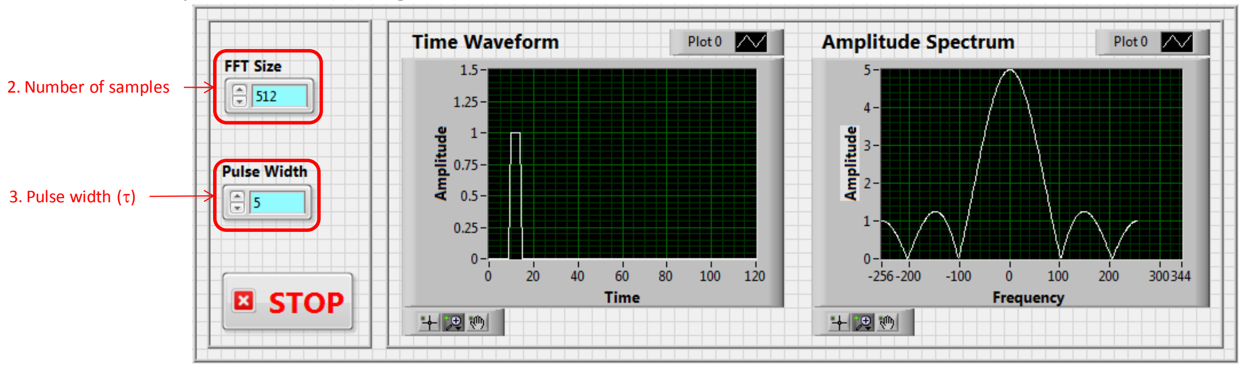
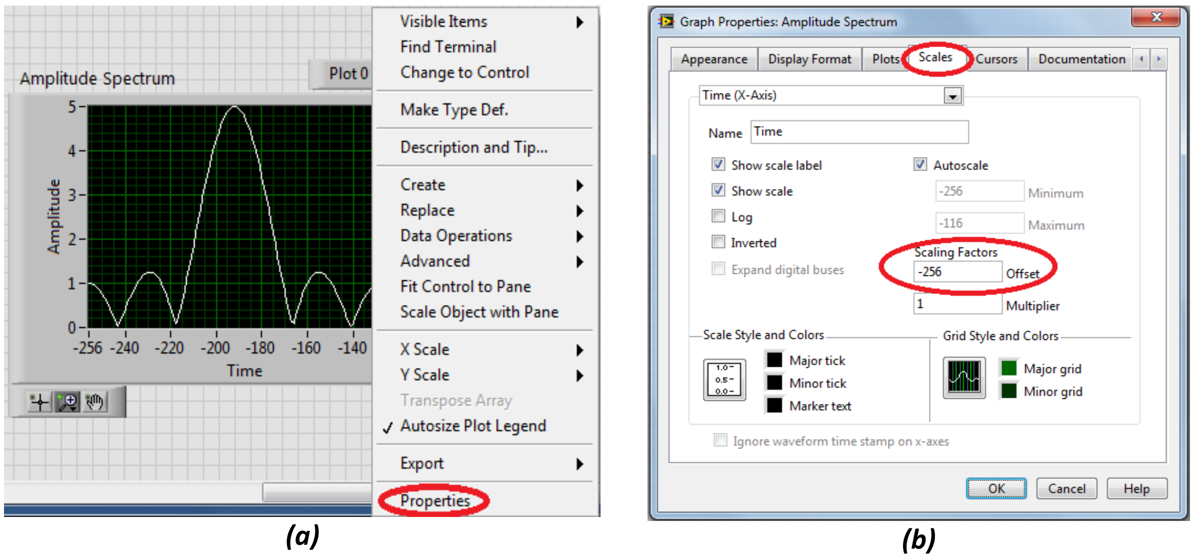


Fig. 12: FFT demo VI Block diagram (top) and Front panel (bottom)

4. Before running the VI, modify the amplitude spectrum display by changing the display's center point value from 0 to 256 as follows: right click the amplitude spectrum graph and then select Properties (Fig. 13a). Using the "Scales" tab in the "Properties" window, change the "Scaling Factor Offset" to -256 (Fig. 13b).

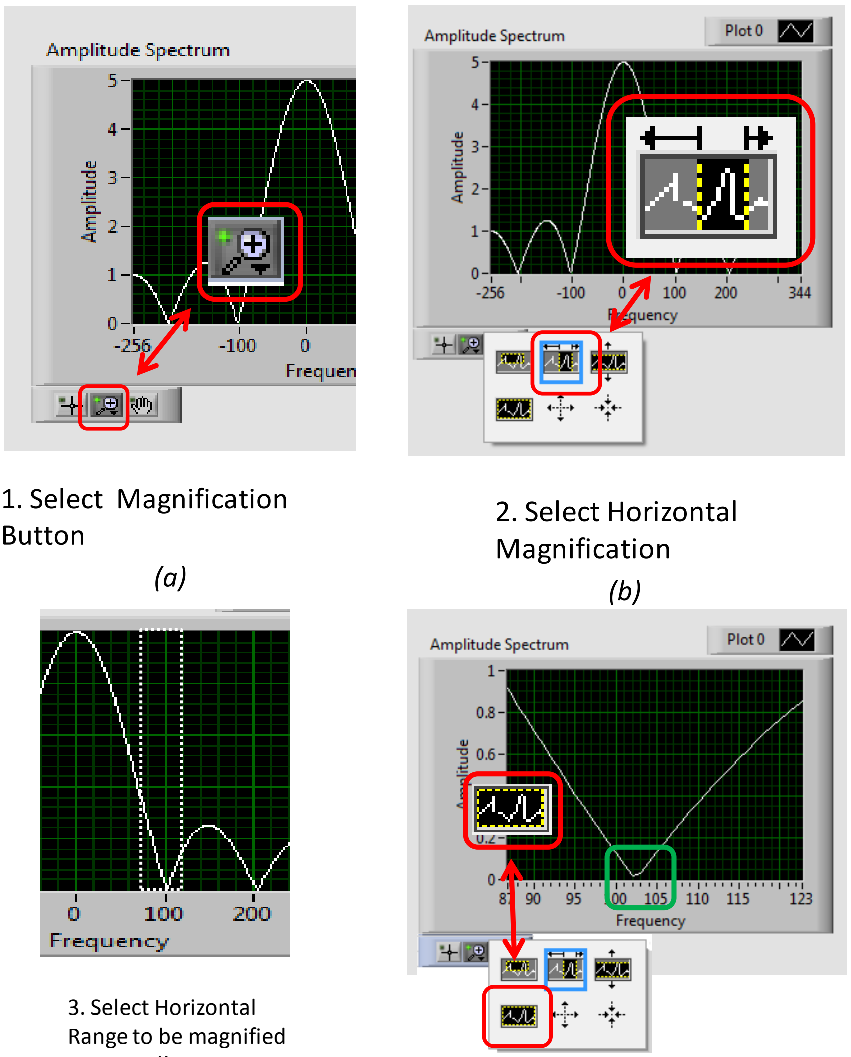


Time (X-Axis)
Name: Time
☑ Show scale label
☑ Show scale
☐ Log
☐ Inverted
☐ Expand digital buses
☑ Autoscale
-256 Minimum
-116 Maximum
Scaling Factors
-256 Offset
1 Multiplier
Scale Style and Colors: Major tick, Minor tick, Marker text
Grid Style and Colors: Major grid, Minor grid
☐ Ignore waveform time stamp on x-axes

Fig. 13: Changing the scaling factor on the Amplitude Spectrum Display

5. Start/Run ***"fft_demo.vi"***.
6. Observe the waveform on **Amplitude Spectrum** waveform chart and record the amplitude and bandwidth as indicated in Table II.
7. Update the pulse width (τ) (Fig. 12) to the next value in Table II and repeat steps 6 until the table is complete.

Note: You will have to use the plot's magnification tool to make the observations. The waveform chart in the demo VI will have the Graph palette visible as shown in Fig. 14(a) in the lower left hand corner. When you click on the middle button, you will see the Graph Magnification Tools Palette. The please select the Horizontal magnification tool Fig. 14(b). This tool will allow you to zoom in along the x-axis without changing the y-axis. This will allow you to focus in on the point of contact on the x-axis (Fig. 14(c)). The cursor will show up as a pair of vertical lines delineating the range of x-values of interest. Finally, you can see the point of contact (Fig. 14(d) green box) and record your observations. After observing the data, you should use the return to non-magnified mode using the return to original scaling tool (Fig. 14(d) red box).

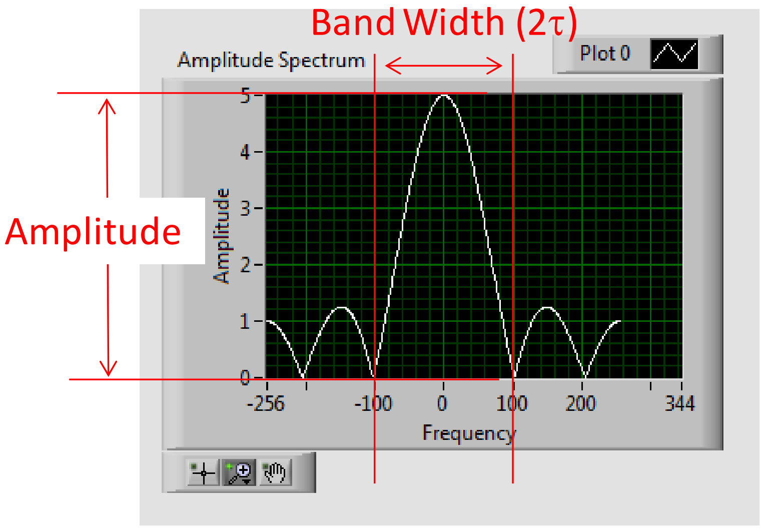

Amplitude Spectrum
Amplitude
Frequency
Plot 0
1. Select Magnification Button
(a)
2. Select Horizontal Magnification
(b)
3. Select Horizontal Range to be magnified using tool's cursor
(c)

4. Observe data and return to non-magnified mode for next observation.

(d)

Fig. 14: Graph Palette Magnification Tools

Table II – Pulse Duration Observations

Pulse Width	***Bandwidth (Hz)***	***Amplitude***	
1			
3			
5			
7			
9			

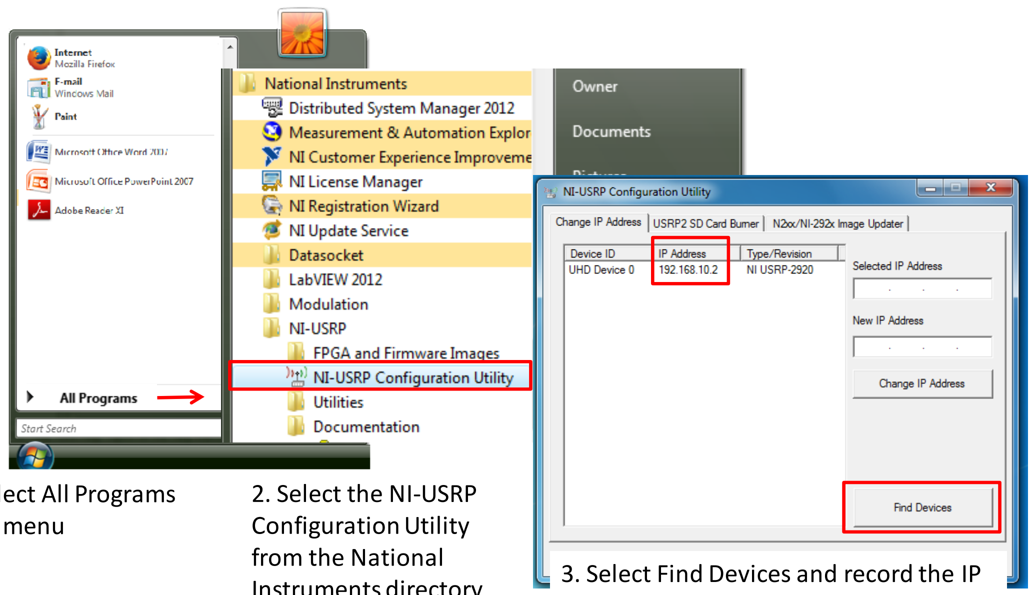
3.4.3 Monitoring the spectrum with NI USRP and LabVIEW spectrum monitoring example VI

Spectrum monitoring involves examining the frequency content of a spectral band as a function of time. This display uses the FFT to obtain a spectrum from observed data.

Important set-up notes:

- ✓ Make sure the global set-up configuration (discussed in lab 1) has been performed before interfacing with the USRPs.
- ✓ Verify that device name fields in both Tx and Rx VIs are set to the IP address of the URSP in use (see figure 16 below).

1. Connect the computer to the USRP using an Ethernet cable.
2. Open the NI-USRP Configuration Utility found in the National Instruments directory under programs files as shown in Fig. 2. Be sure to record the IP addresses since you will need them to configure your software.



Device ID	IP Address	Type/Revision
UHD Device 0	192.168.10.2	NI USRP-2920

1. Select All Programs from menu

2. Select the NI-USRP Configuration Utility from the National Instruments directory

3. Select Find Devices and record the IP address of the radio or radios since you will need them to configure the software in the lab.

Fig. 15: Finding the IP Address: Radio Connectivity Test

If the IP address does not appear in the window then check your connections and ask the Teaching Assistant (TA) to verify that the LAN card has been configured correctly.

3. Connect the VERT2450 Antenna to the "RX2" input on one of the radios connected to your workstation (see Fig. 16). For this exercise you won't need to connect/use the attenuator between the USRP's Rx input and the antenna.

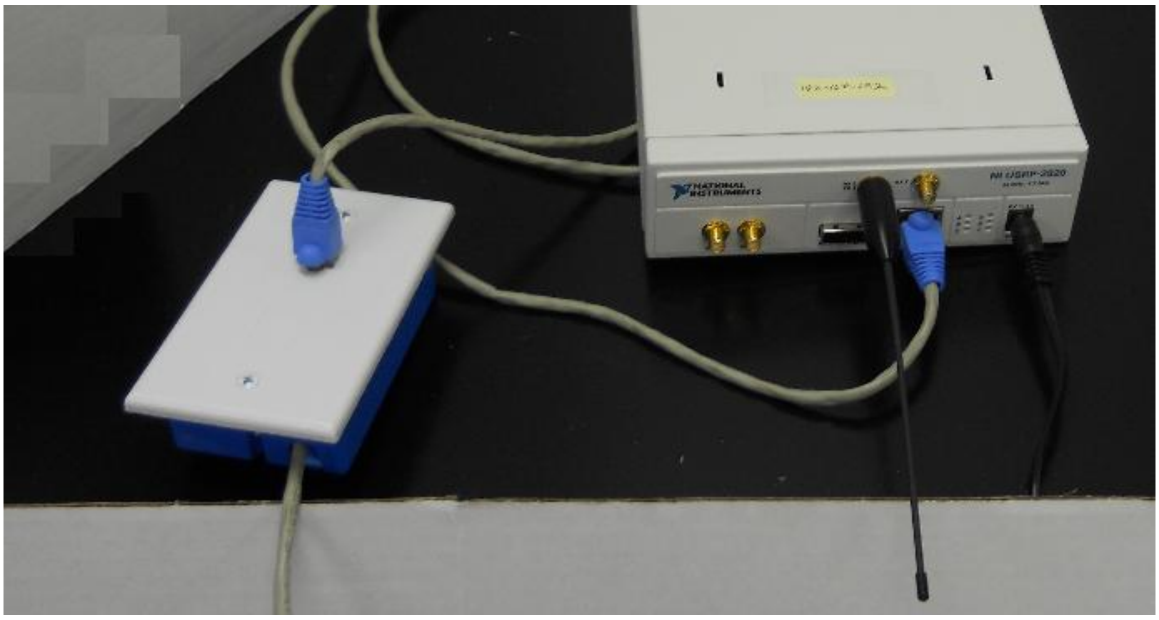
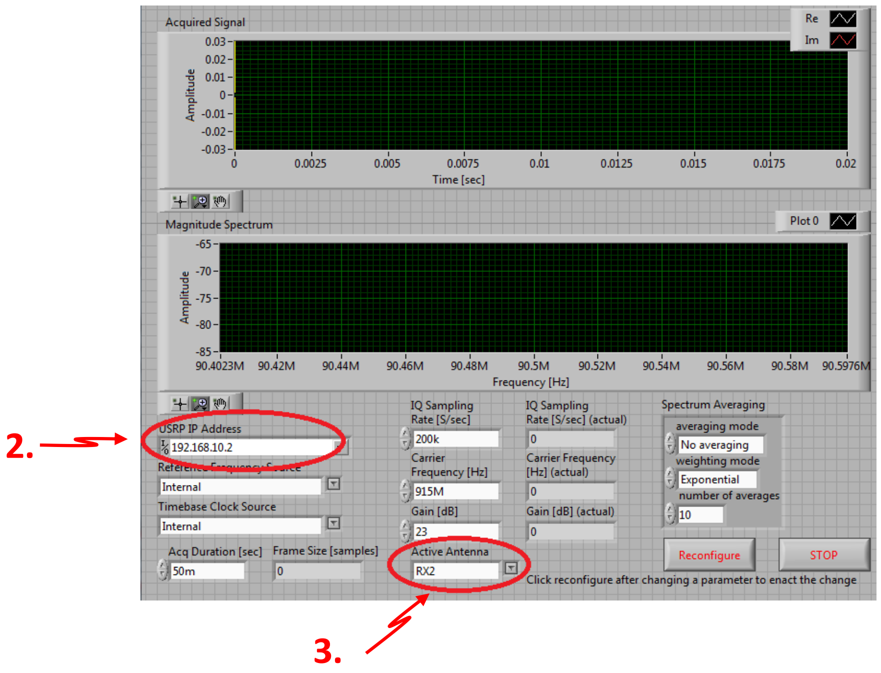


Fig. 16: USRP physical set-up: connecting the antenna to the RX2 input.

4. Locate and open the VI "niUSRP EX Spectral Monitoring (Interactive).vi". It can be found in C:\Program Files\National Instruments\LabVIEW 2012\examples\instr\niUSRP). When opened, you should see the front panel illustrated in Fig. 17.

5. Set the USRP IP Address (see Fig. 17) to the actual IP of the USRP you are using (each USRP should be labeled with its IP address).
6. Set the active antenna to "RX2" (see Fig. 17).



USRP IP Address: 192.168.10.2
Reference Frequency Source: Internal
Timebase Clock Source: Internal
Acq Duration [sec]: 50m
Frame Size [samples]: 0
IQ Sampling Rate [S/sec]: 200k
IQ Sampling Rate [S/sec] (actual): 0
Carrier Frequency [Hz]: 915M
Carrier Frequency [Hz] (actual): 0
Gain [dB]: 23
Gain [dB] (actual): 0
Active Antenna: RX2
Spectrum Averaging
averaging mode: No averaging
weighting mode: Exponential
number of averages: 10
Click reconfigure after changing a parameter to enact the change

Fig. 17: Front Panel of niUSRP EX Spectral Monitoring (Interactive).vi

7. Run the VI (for now don't change any of the other controls on the Front panel), and look at the resulting spectrum reading (see Fig. 18; the amplitude is shown in decibels: (Db)).

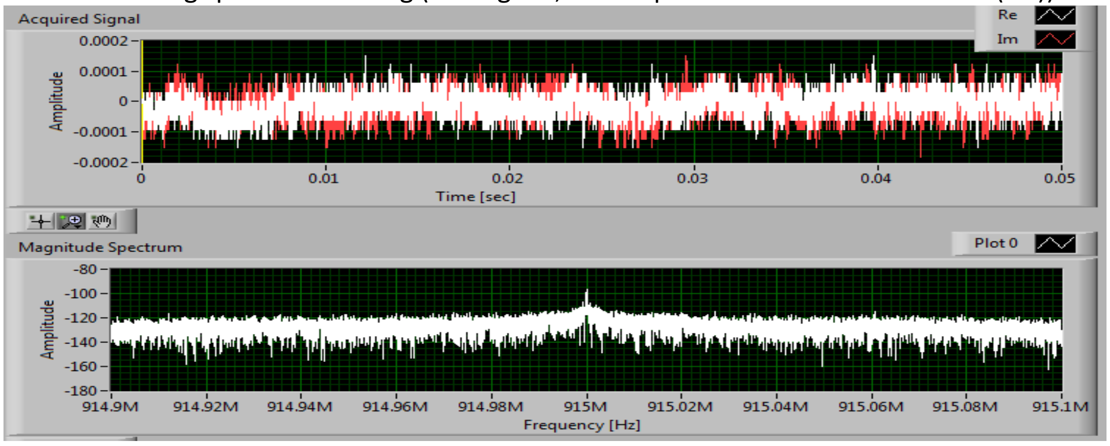


Fig. 18: Observed Spectrum

Note: *The top display in **Fig. 18** shows the quadrature signals (in-phase is shown in red, and out-of-phase in white) sensed by the radio. The USRP is designed use quadrature modulation and you will be using the radio's capability to adapt this modulation technique to support other modulation approaches. For now you will focus only on the magnitude spectrum.*

8. Leave all other parameters unchanged except changing the "Averaging mode" to either "RMS averaging" or "Peak hold" using the up/down button. This will give you a better picture, showing the existing frequency peaks.

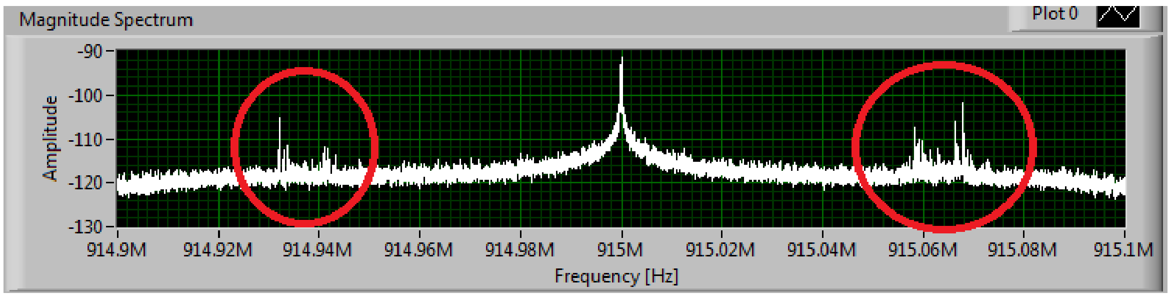
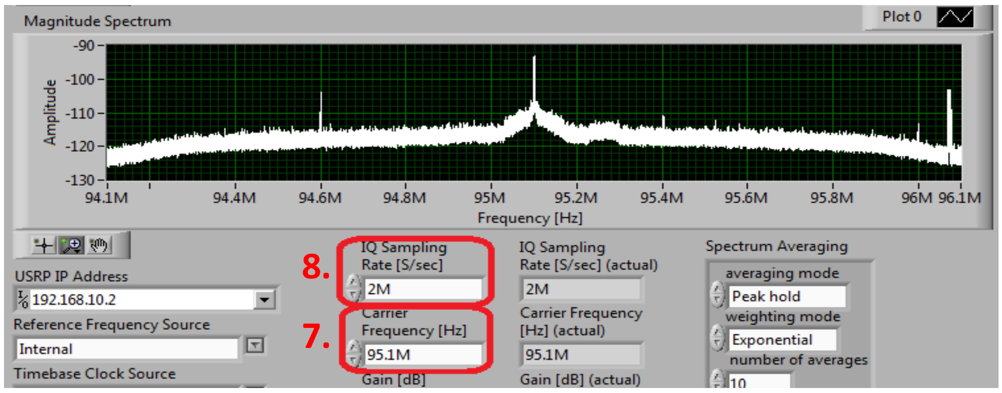


Fig. 19: Spectrum with either "*RMS averaging*" or "*Peak hold*"

9. Set the "Carrier Frequency [Hz]" control to the local radio station at 91.5 MHz (Fig. 20). What other radio stations do you see?
10. Changing the "IQ Sampling Rate [S/sec]" control to up to 2M samples per second will allow you to observe a wider band of spectrum. Fig. 20 shows the spectrum reading results of changing the earlier mentioned controls. What additional radio stations do you see?



USRP IP Address: 192.168.10.2
Reference Frequency Source: Internal
Timebase Clock Source
IQ Sampling Rate [S/sec]: 2M
Carrier Frequency [Hz]: 95.1M
Gain [dB]
IQ Sampling Rate [S/sec] (actual): 2M
Carrier Frequency [Hz] (actual): 95.1M
Gain [dB] (actual)
Spectrum Averaging
averaging mode: Peak hold
weighting mode: Exponential
number of averages: 10

Fig. 20: Observed Spectrum with "IQ Sampling Rate" and "Carrier Frequency" changes

3.5 Lab Write-up

Performance Checklist

Fast Fourier Transforms

1. Describe your observations about the pulse width. What happens to the amplitude spectrum of the signal when you change the pulse duration in the time domain?

2. Calculate the actual *frequency* values for the first two zero intersections of the Amplitude spectrum corresponding to the equation in Fig. 10b. Use $\tau = 10$ and 20.

3. Discuss the cause of the distortion in the spectrum analyzers at the center frequency (see Fig. 19).

4. "Given your answers to the previous questions, if you had a limited spectrum to work with, how would you select the size of the time window [symbols] in the OFDM windowing function given in equation (5)?"

5. Given your observations of the FM spectrum usage in the second part of the lab procedure, is it possible to set up an OFDM frequency band in the FM spectrum? What is the frequency spacing for FM radio stations?

Performance Measures

Task	Standards	Sat/Unsat
Hardware Setup	Working setup for all 3 configurations: Broadcast, P2P wireless, and Loop-back cable.	
Running VIs	Successful transmission and reception of audio files for all 3 configurations: Broadcast, Point to point wireless, and Loopback-cable.	
QOS in P2P wireless configuration	Quality of received signal heard over the radio channel in point to point wireless configuration.	

Discussion

Did all configurations perform as expected?
Did you have any difficulties completing the lab?
Did your TA provide enough guidance?
Do you have any recommendations to improve the lab?

3.6 References

[1] R.E. Ziemer and W.H. Tranter, Chapter 2: Signal and Linear System Analysis, In “Principles of Communications: Systems, Modulation and Noise”, Fifth Edition.

[2] Fourier Transforms and the Fast Fourier Transform (FFT) Algorithm, Paul Heckbert. “Notes 3, Computer Graphics 2, 15-463.” retrieved June 12, 2014, from http://www.cs.cmu.edu/afs/andrew/scs/cs/15-463/2001/pub/www/notes/fourier/fourier.pdf.

[3] Introduction to the Fourier Transform, “The Fourier Transform” retrieved June 12, 2014, from http://www.thefouriertransform.com/transform/fourier.php#introduction.

[4] Chapter 9: MIMO, Thomas Schwengler. “Wireless & Cellular Communications: Class Notes for TLEN-5510 - Fall 2014” retrieved August 23, 2014, from http://morse.colorado.edu/~tlen5510/text/classwebch9.html#x50-2020009.1.

[5] National Instruments, Spectrum Monitoring with NI USRP, retrieved August 23, 2014. http://www.ni.com/white-paper/13882/en/.

4 Digital Filter Design: Introduction to LabVIEW Filter Design Toolkit.

4.1 Summary

This lab will introduce you to the LabVIEW filter design toolkit. You will be asked to observe the power spectrum of a given signal (or combination of signals). You will be asked to observe the power spectrum of a given signal (or combination of signals) and design a series of digital filters that will allow you to isolate and analyze specific signal frequencies.
The NI software below will be used in the course:

- NI LabVIEW 2012 or later.
- NI LabVIEW Filter Design Toolkit

4.2 Background

You will be asked to observe the power spectrum of a given signal (or combination of signals) and design a series of digital filters that will allow you to isolate and analyze specific signal frequencies.. You have the option of designing both a Finite Impulse Response (FIR) and an Infinite Impulse Response (IIR) filters. You will need to determine a sampling frequency, filter specifications, and design method.

4.2.1 Finite Impulse Response Filters

A FIR or non-recursive filter has no feedback. The filter's output is a function of the input signal only. The difference equation is given by

$$y[n] = \sum_{k=0}^{N} b_k x[n-k] \tag{15}$$

where, $y[n]$ is the filter output, $x[n]$ is the signal being filtered, b_k are the filter coefficients, and where N is the filter order. The output signal $y[n]$ of the filter in response to an impulse is limited only the last N values of $x[n]$, so after $N+1$ samples the response returns to zero. For example, the response of a fifth order filter (Fig. 21) consists of a finite sequence of six $(N+1)$ samples.

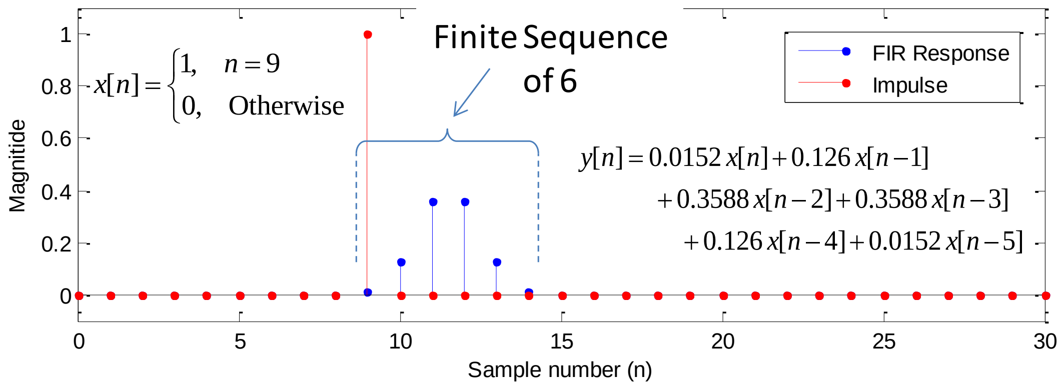


Fig. 21: Low-pass FIR Filter: Impulse Response

4.2.2 Infinite Impulse Response filters

Infinite Impulse Response (or recursive) filters are a more complex type of filter than a FIR filter, with an output at time n, given by:

$$y[n] = \sum_{k=0}^{N} b_k x[n-k] + \sum_{i=1}^{M} a_i y[n-i] \tag{16}$$

where, $y[n]$ is the filter output, $x[n]$ is the signal being filtered, b_k and a_i are the filter coefficients, and where$N$ previous inputs and M outputs. The output signal of the filter can be non-zero infinitely, even when the input signal has a value of zero. In theory, when a recursive filter is excited by an impulse, the output will persist forever.

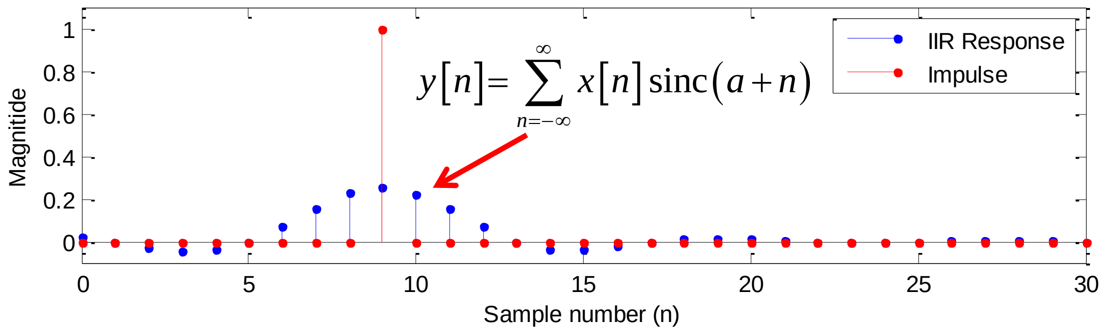


Fig. 22: Low-pass IIR Filter: Impulse Response

The corresponding transfer function for an IIR filter is given by equation.

$$H[z] = \frac{\sum_{k=0}^{N} b_k z^{-k}}{1 - \sum_{i=1}^{M} a_i z^{-i}} = \frac{N[z]}{D[z]} \tag{17}$$

This equation, in addition to having N zeros (as the FIR, the roots of $N[z]$), it also has M poles (the roots of $D[z]$), which for a stable filter, are required to be inside the unit circle in the z plane.

4.2.3 Sampling frequency

The symbol f_s denotes the sampling frequency, which is the expected rate at which you sample the input signal to the filter. In the LabVIEW Digital Filter Design Toolkit, the default sampling frequency is 1, which is the normalized sampling frequency.

4.2.4 Filter design specifications

In this lab, you will be asked to design low-pass, high-pass, band-pass, and band-stop filters, the characteristics of which are outlined below.

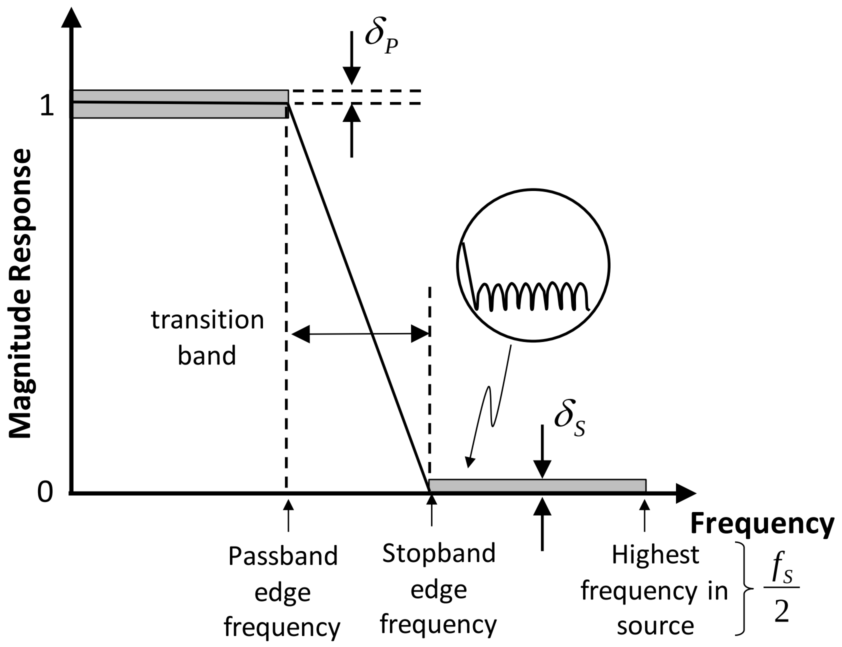


Fig. 23: Absolute Low-pass Filter: Magnitude Frequency Response

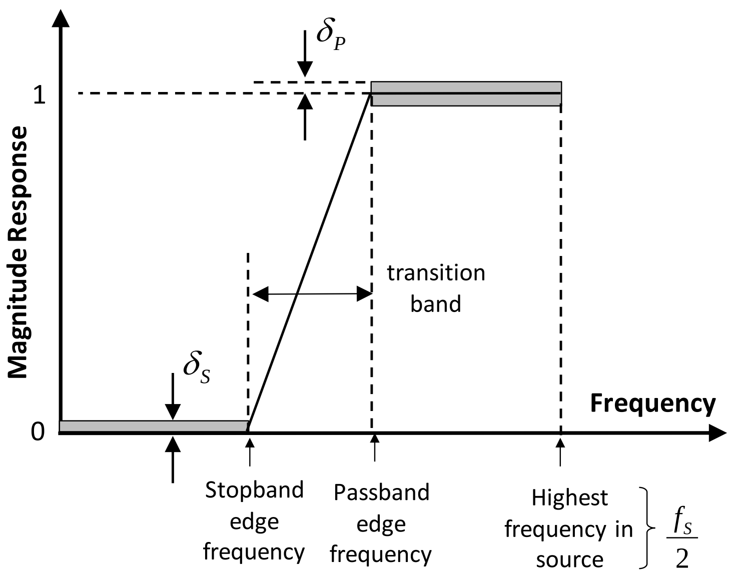


Fig. 24: Absolute High-pass Filter: Magnitude Frequency Response [1]

As illustrated in Fig. 23 and Fig. 24 (low-pass and high-pass), the frequency range from the pass-band edge frequency (cut-off frequency) to the stop-band edge frequency is the transition band, which has an unspecified frequency response. The filter pass-band and stop-band may contain oscillations, which are known as ripples. A typical example of a ripple appears in the circle (zoomed view) of the previous in Fig. 23. In the figure, δ_P indicates the magnitude of the pass-band ripple (or the maximum deviation from the unity) and δ_S indicates the magnitude of the response of the stop-band ripple (or the maximum deviation from zero).

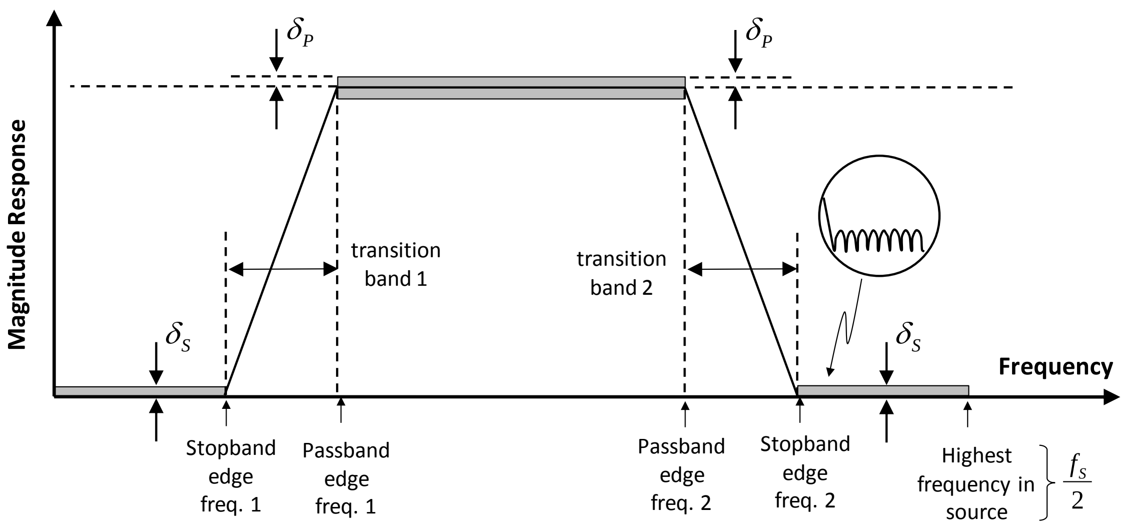


Fig. 25: Absolute Band-pass Filter: Magnitude Frequency Response [1]

For band-pass filters (Fig. 25), stop-band edge frequency 1 indicates the maximum frequency of the lower frequency range to be attenuated, and the stop-band edge frequency 2 indicates the minimum frequency of the higher frequency range to be attenuated. The frequency range between pass-band edge frequencies 1 and 2 indicates the range of frequencies that can pass through the filter.

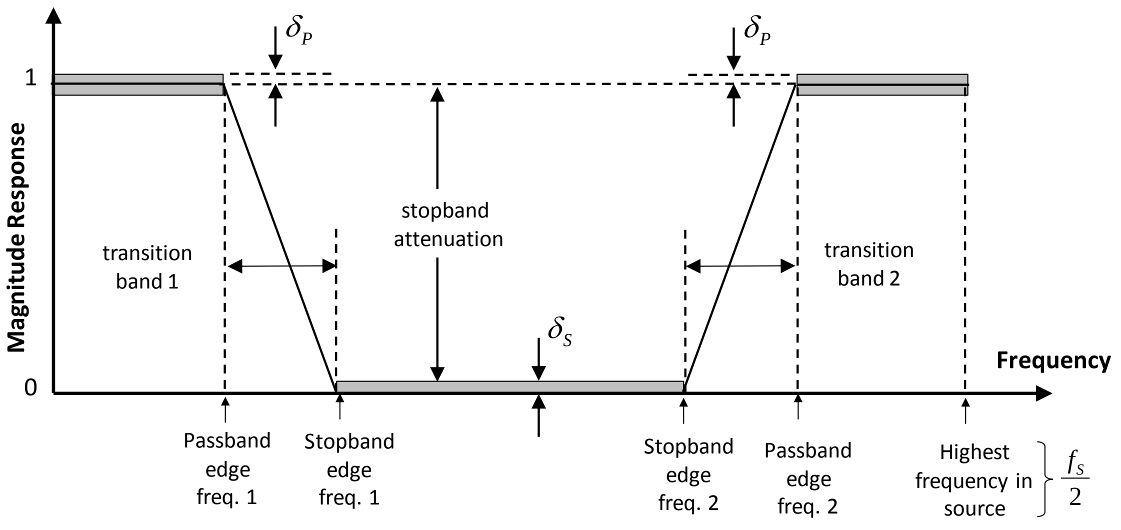


Fig. 26: Absolute Band-stop Filter: Magnitude Frequency Response [1]

For band-stop filters, (Fig. 26), pass-band edge frequency 1 indicates the maximum frequency of the lower frequency range that can pass through the filter, and pass-band edge frequency 2 indicates the minimum frequency of the higher frequency range that can pass through the filter. The frequency range between stop-band edge frequencies 1 and 2 indicates the range of frequencies to be attenuated.

4.2.5 Design methods

The LabVIEW Digital Filter Design Toolkit[1] provides the following finite impulse response (FIR) filter design methods:

- Kaiser Window
- Dolph-Chebyshev Window
- Equiripple FIR

The Kaiser Window method and the Dolph-Chebyshev Window method allow you to obtain the filter coefficients directly from the analytical equations, making them easier to use than the Equiripple FIR method. The Equiripple FIR method is more complicated because it uses a least square optimization to produce an optimal filter and is often the best solution for most FIR filter design problems.

In addition to the FIR-based methods, the Digital Filter Design Toolkit supports the following infinite impulse response (IIR) filter design methods:

- Butterworth
- Chebyshev
- Inverse Chebyshev
- Elliptic

The following table summarizes the main features of the four IIR-based design methods so you can determine the best IIR filter design method to use.

Table III – IIR Design Method Summary

IIR Filter	Ripple in Pass-band?	Ripple in Stop-band?	Transition Bandwidth for a Fixed Order	Order for Given Filter Specifications
Butterworth	No	No	Widest	Highest
Chebyshev	Yes	No	Narrower	Lower
Inverse Chebyshev	No	Yes	Narrower	Lower
Elliptic	Yes	Yes	Narrowest (Sharpest)	Lowest

4.3 Pre-Lab

In this lab, you will be designing four digital filters: low pass, high-pass, band-pass, and band-stop. For each filter type, you will analyze the power spectrum of a chosen signal and frame a set of specifications for a desired output. You will be using the Dual-Tone Multi-Frequency (DTMF) coding scheme. You have probably heard them when you dial a telephone number. Each key on the phone is assigned a pair of frequencies. Thus each key will two peaks in the power spectrum of the signal (Fig. 27).

	1209 Hz	**1334 Hz**	**1477 Hz**	**1633 Hz**
697 Hz	1	2	3	A
770 Hz	4	5	6	B
852 Hz	7	8	9	C
941 Hz	*	0	#	D

A key press of 4 is encoded as

$$\begin{aligned} y(t) &= \sin\left(770(2\pi)t\right) + \sin\left(1209(2\pi)t\right) \\ &= \sin\left(4838\,t\right) + \sin\left(7596\,t\right) \end{aligned}$$

Fig. 27: DTMF Keypad Tone Frequencies

Given a desired output, you can easily determine the frequency specifications of your filter design by using tones of the DTMF coding.

a) A Low-pass filter that isolates the lower tone in the DTMF for a key press of 4 (Fig. 28). Record your specifications in Table IV for future reference.
b) A High-pass filter that isolates the higher frequency in the DTMF for a key press of 4 (Fig. 28). Record your specifications in Table IV for future reference.

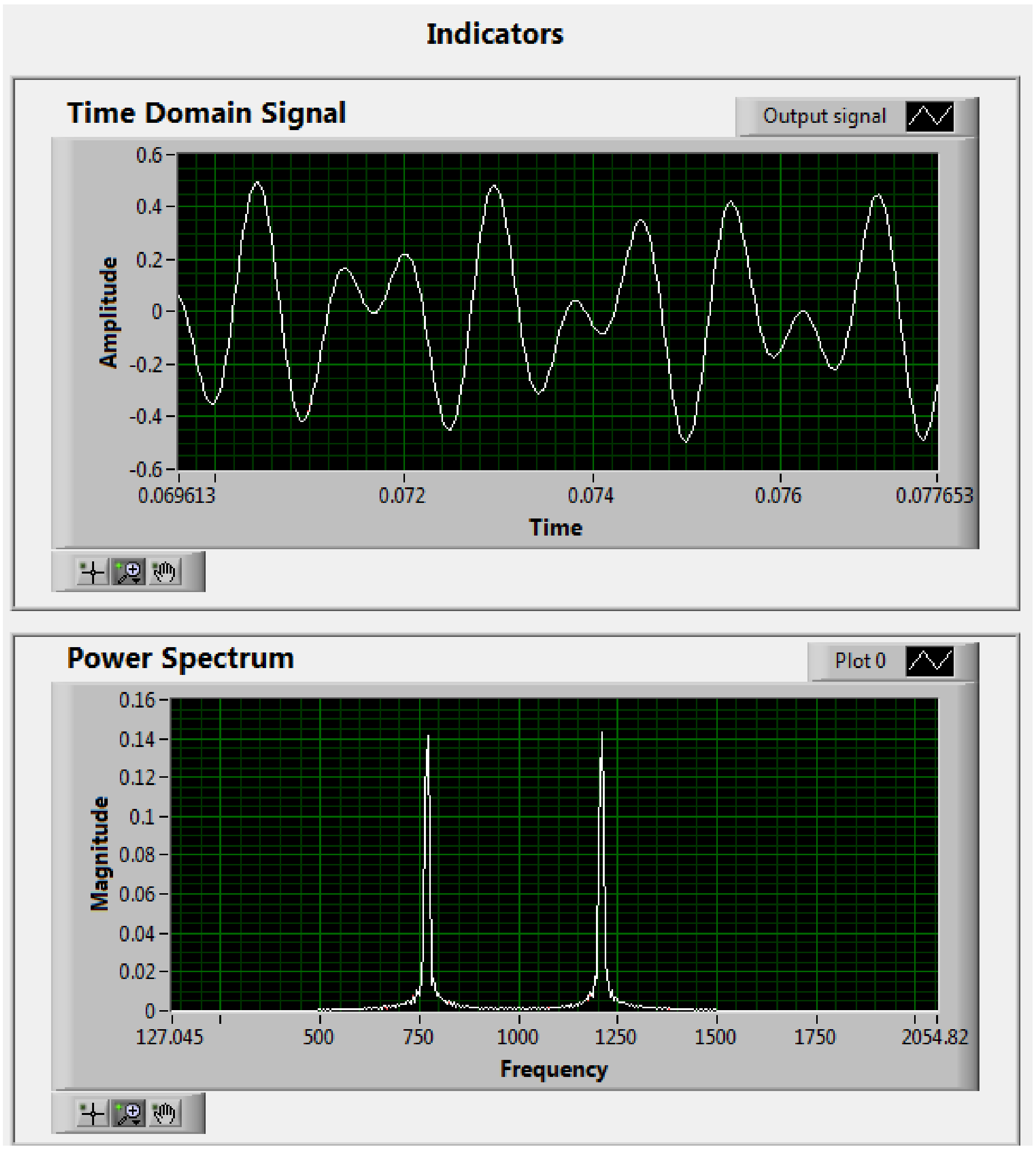


Fig. 28: Key Press of 4 Time Domain and Frequency Spectrum (dtmf-4.wav)

c) Band-pass filter to isolate a key press of 4 (Fig. 28) when both 3 and 4 keys are pressed at the same time (Fig. 29). Record your specifications in Table V for future reference.

d) Band-stop filter to isolate a key press of 3 (Fig. 28) when both 3 and 4 keys are pressed at the same time (Fig. 29). Record your specifications in Table V for future reference.

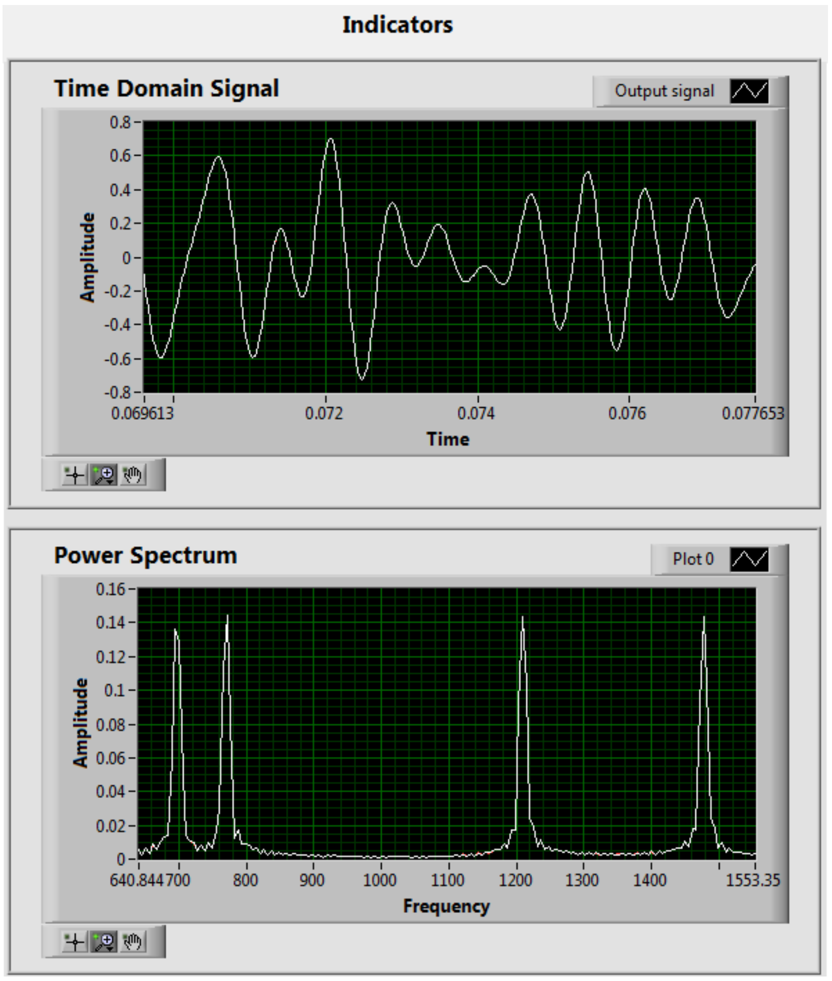


Fig. 29: Combination of both 3 and 4 Keys DTMF Tones (dtmf-3.wav and dtmf-4.wav)

Table IV - Specifications For Lowpass And Highpass Filters

Specification	**Low-pass**	**High-pass**
1. Pass-band edge frequency		
2. Pass-band Ripple		
3. Stop-band edge frequency		
4. Stop-band attenuation		

Table V - Specifications For Band-pass And Band-stop Filters

Specification	**Band-pass**	**Band-stop**
1. Pass-band edge frequency 1		
2. Pass-band edge frequency 2		
3. Pass-band ripple		
4. Stop-band edge frequency 1		
5. Stop-band edge frequency 2		
6. Stop-band attenuation		

4.4 Lab Procedure

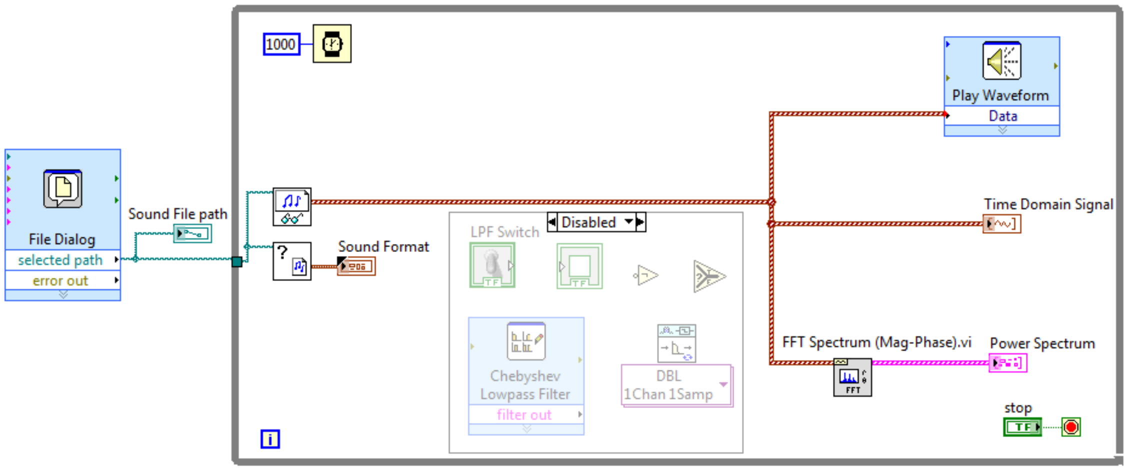
4.4.1 Description

You will be designing four VIs, each using a different filter type to filter a DTMF wav file or a combination of DTMF wav files. The specifications framed in the pre-lab are to be used for the first instance. However, in case your filters don't work well, you'll always have the option of tuning the specifications later.

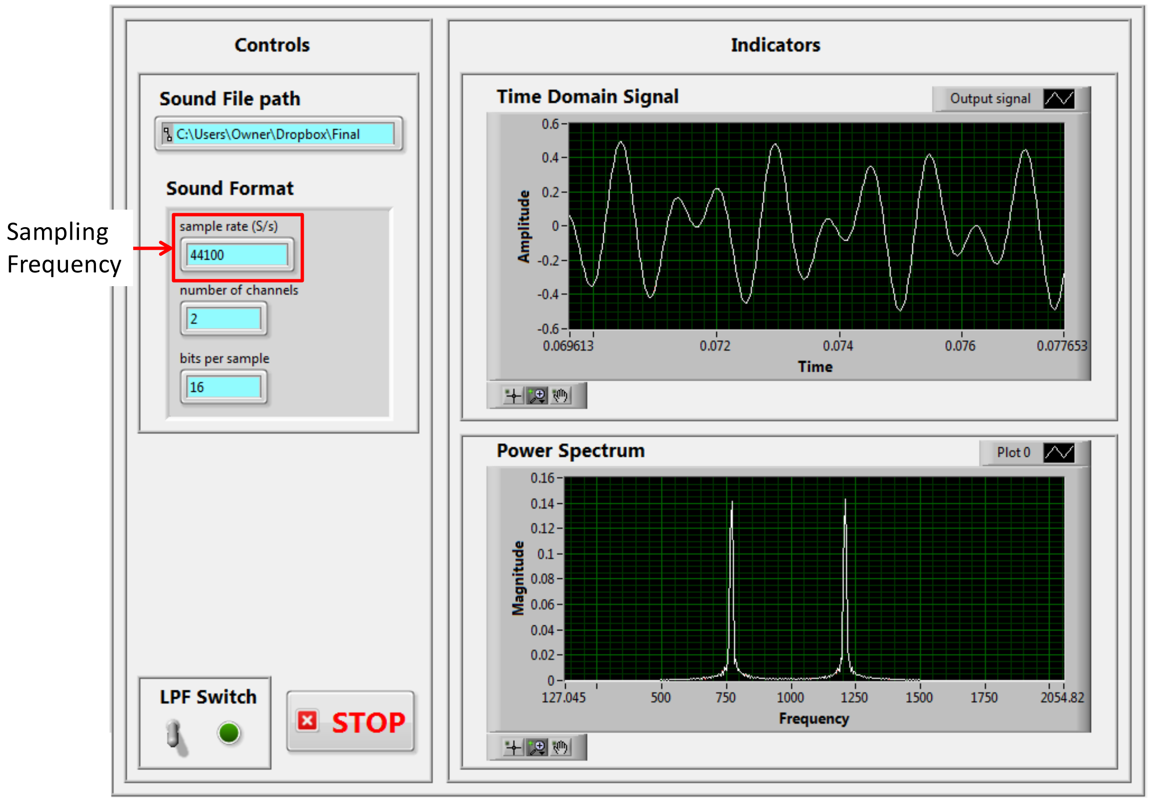
4.4.1.1 Low-pass filter design

1. Open "DTMF_Demo_LPF.vi" (Fig. 30a) and run it. Select "dtmf-4.wav" and then observe the signal in the time and frequency domains (Fig. 30b). Note the sampling frequency, you will need it to configure the filter block.
2. Navigate to the block diagram for the filter, and remove the blocks from the "disabled" structure as directed.
3. Drag and drop the "Classical Filter" VI from the "disabled" structure into the "while" loop structure.
4. Double click on the "Classical Filter" VI and a pop-up window (Fig. 31) will appear. This pop-up window is your design tool screen where you enter the respective filter design specifications (Table IV).

Note: This lab has been designed on LabVIEW 2012 edition. The signal processing/digital filter design palettes and sub-VIs could be modified in future LabVIEW releases. If you cannot find the necessary functions or controls, ask the TA for help.

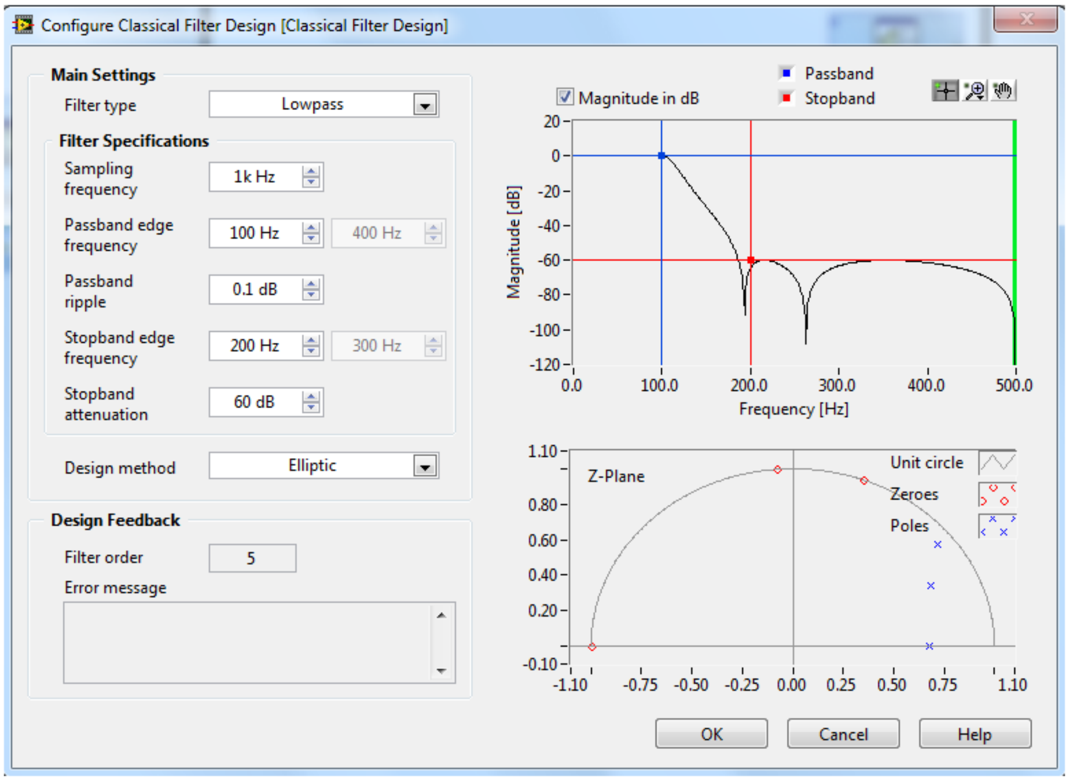


a) Block diagram



b) Front panel

Fig. 30: DTMF_Demo_LPF_Start VI



Configure Classical Filter Design [Classical Filter Design]

Main Settings
Filter type: Lowpass

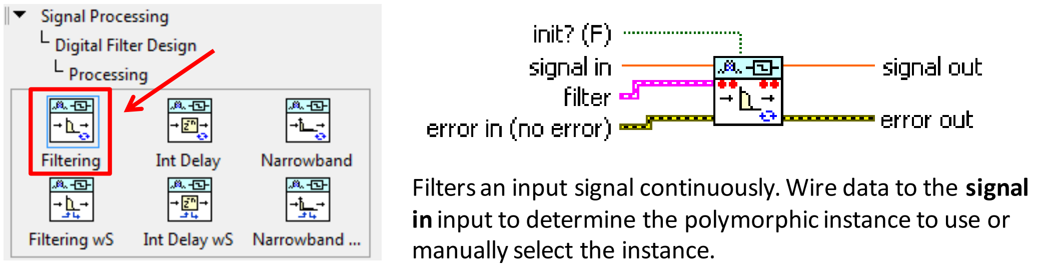
Filter Specifications
Sampling frequency: 1k Hz
Passband edge frequency: 100 Hz (400 Hz)
Passband ripple: 0.1 dB
Stopband edge frequency: 200 Hz (300 Hz)
Stopband attenuation: 60 dB

Design method: Elliptic

Design Feedback
Filter order: 5
Error message:

Fig. 31: Classical Filter Design Tool Screen

5. Select the appropriate filter type, sampling frequency (check the sound file details on front panel), and design method (choose "Butterworth" or "Chebyshev"). Next, enter the specifications of the low-pass filter you designed (Table IV).
6. To use the filter you designed above, drag and drop the "Filtering" sub-VI (Fig. 32) from the "disabled" structure into the "while" loop.



Filters an input signal continuously. Wire data to the **signal in** input to determine the polymorphic instance to use or manually select the instance.

Fig. 32: Inserting the "Filtering" Sub-VI

7. Wire the sound file and the designed filter to the filtering VI's "Signal In" and "Filter" terminals respectively, respectively.
8. To analyze the performance of your filter, connect the filtered signal to the "Play Waveform Data" VI to listen, and to the waveform charts to analyze plots in the time and frequency domains.
9. Save the modified VI as "DTMF_Demo_LPF_*YOUR_INITIALS*.vi".
10. Next, run your VI. You will see that your input signal has been filtered: the higher frequency peak is no longer present, and you have a clean sinusoid in the time domain (Fig. 33).

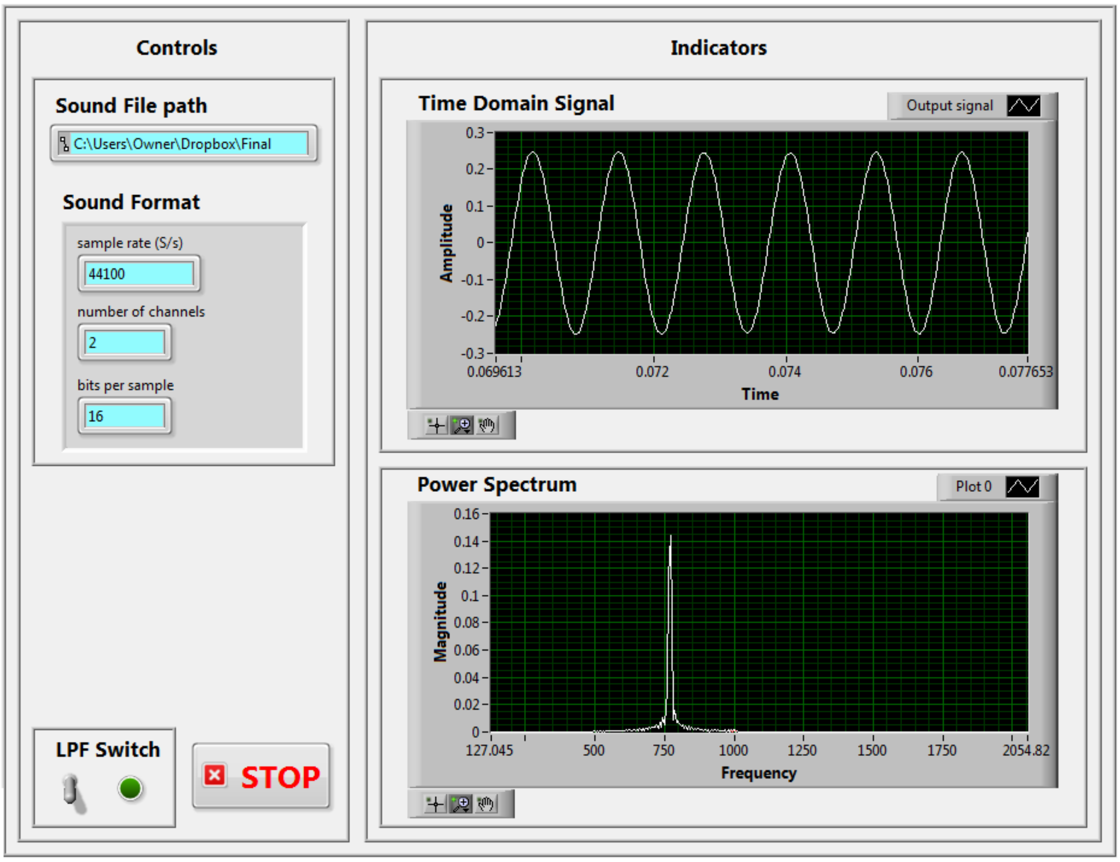
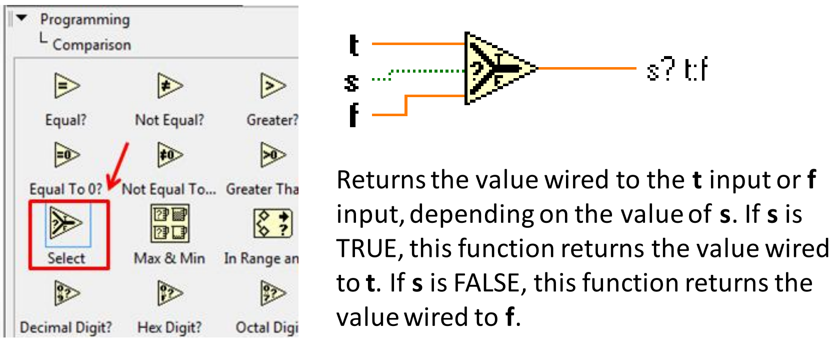


Fig. 33: Desired Low-pass Filter Result.

If you don't get the right result, right click on the "Classical Filter" VI, select "Properties", and change the filter specifications. Keep testing until you get it right! The next task is to apply some logic to the VI so that filtering is done only when the filter is turned on from the front panel.

11. Drag and drop the "Select" control (Fig. 34) from the "disabled" structure into the "while" loop structure.



Returns the value wired to the **t** input or **f** input, depending on the value of **s**. If **s** is TRUE, this function returns the value wired to **t**. If **s** is FALSE, this function returns the value wired to **f**.

Fig. 34: Inserting the "Select" Control.

Wire the unfiltered and the filtered data to the FALSE and TRUE case terminals respectively. This means that the filtered data would go through when the condition is TRUE. To the "s" terminal, you have to wire condition.

12. Drag and drop the "Vertical-Toggle" (switch) (Fig. 35a) and a "Round LED" (LED indicator) (Fig. 35b) from the "disabled" structure into the "while" loop structure. When the switch is

"on" (TRUE case), the indicator is turned on and the filtered data is passed to the waveform chart. When the switch is "off" (FALSE case), the indicator is turned off and the filtered data is not passed to the waveform chart.

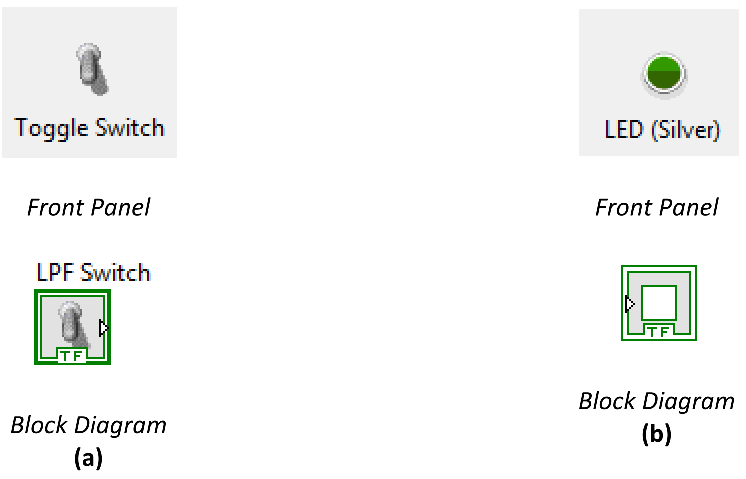
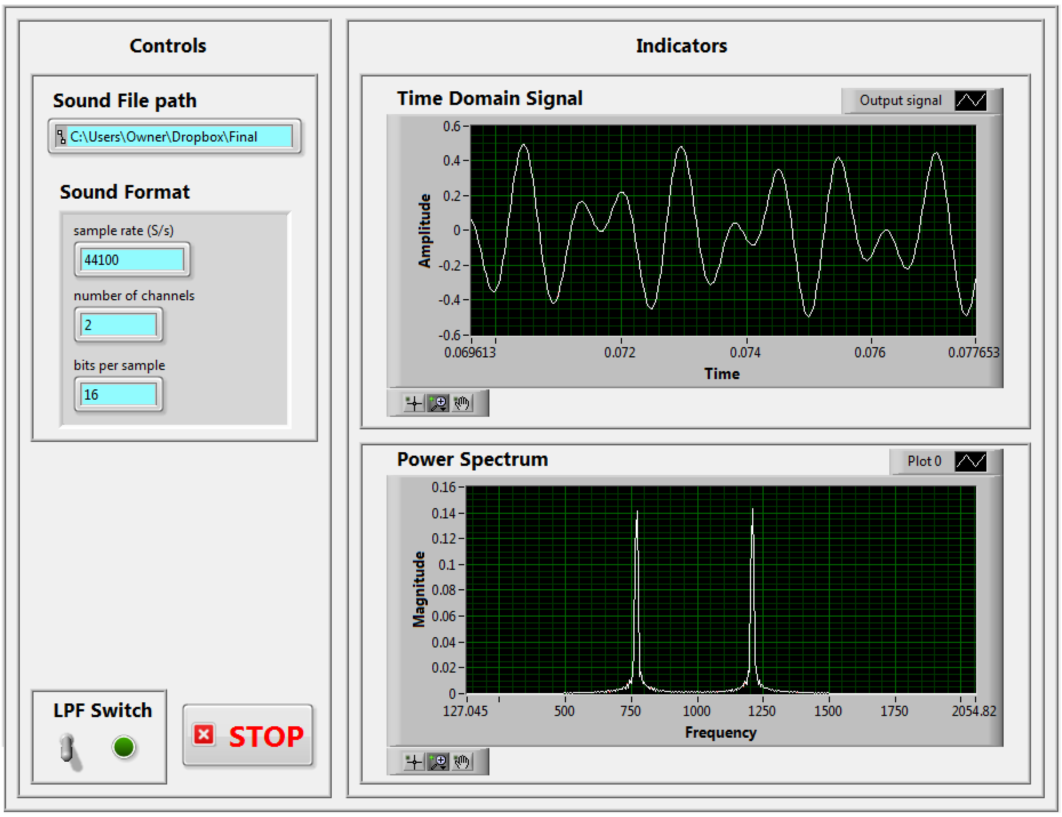
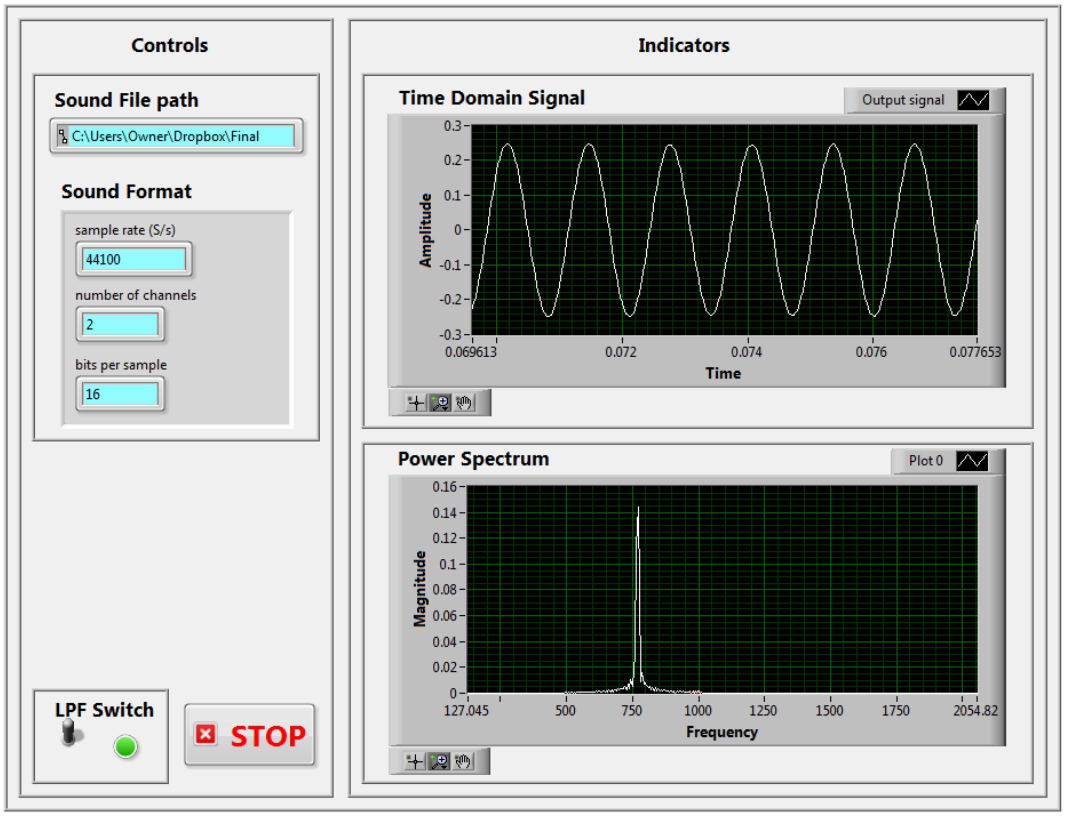


Fig. 35: Inserting the "Round LED" and "Vertical Toggle" Boolean Controls.

Verify your results for both cases (Fig. 36).



a) Filter off

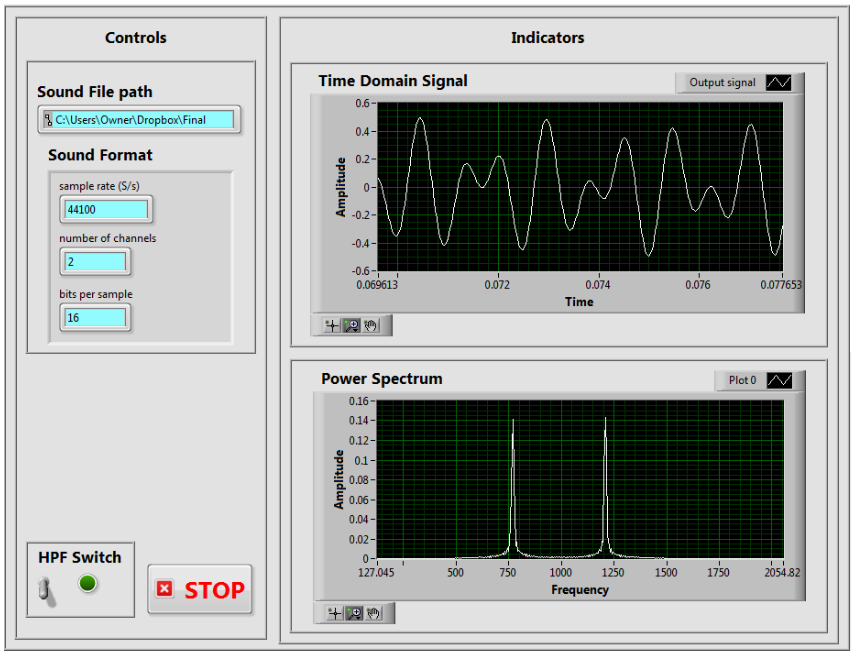


b) Filter on

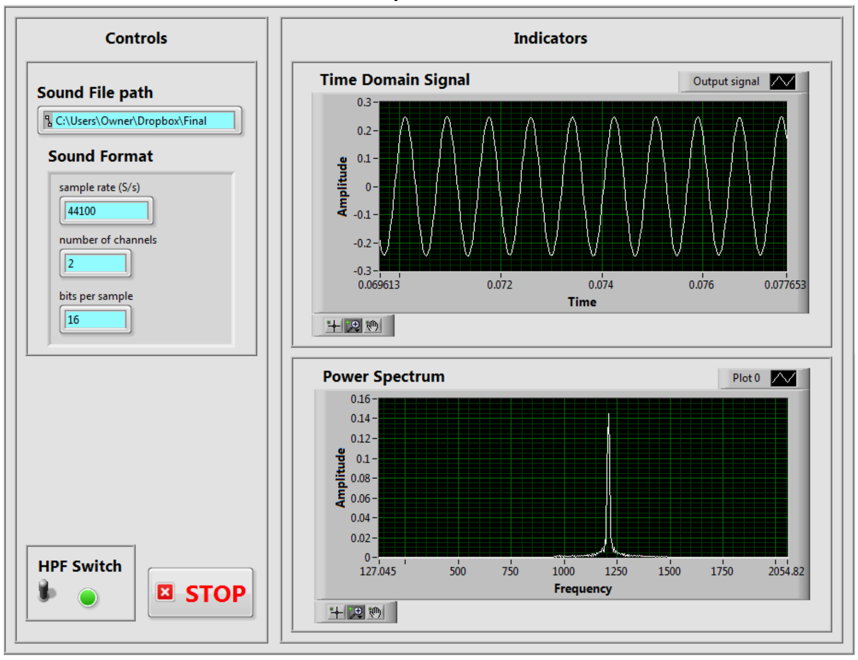
Fig. 36: Low-pass Filter

4.4.1.2 High-pass filter design

a) Copy and paste the low-pass filter design block diagram to a new VI.
b) Save the VI as "DTMF_Demo_HPF_*YOUR_INITIALS*.vi".
c) Double click on the filter you designed to open the properties window (filter design tool screen). Change the filter type to High-pass and modify the other specifications to the ones you designed in the pre-lab session (Table IV).
d) Run the VI and verify your results for both cases (Fig. 37). If you don't get the desired result, modify the filter specifications.


Sound File path: C:\Users\Owner\Dropbox\Final
sample rate (S/s): 44100
number of channels: 2
bits per sample: 16

a) Filter off

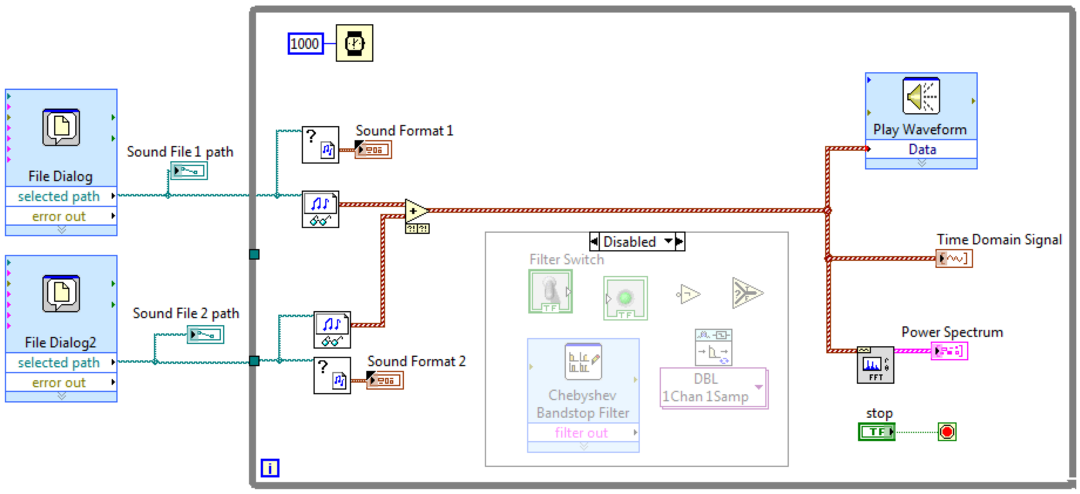

Sound File path: C:\Users\Owner\Dropbox\Final
sample rate (S/s): 44100
number of channels: 2
bits per sample: 16

b) Filter on

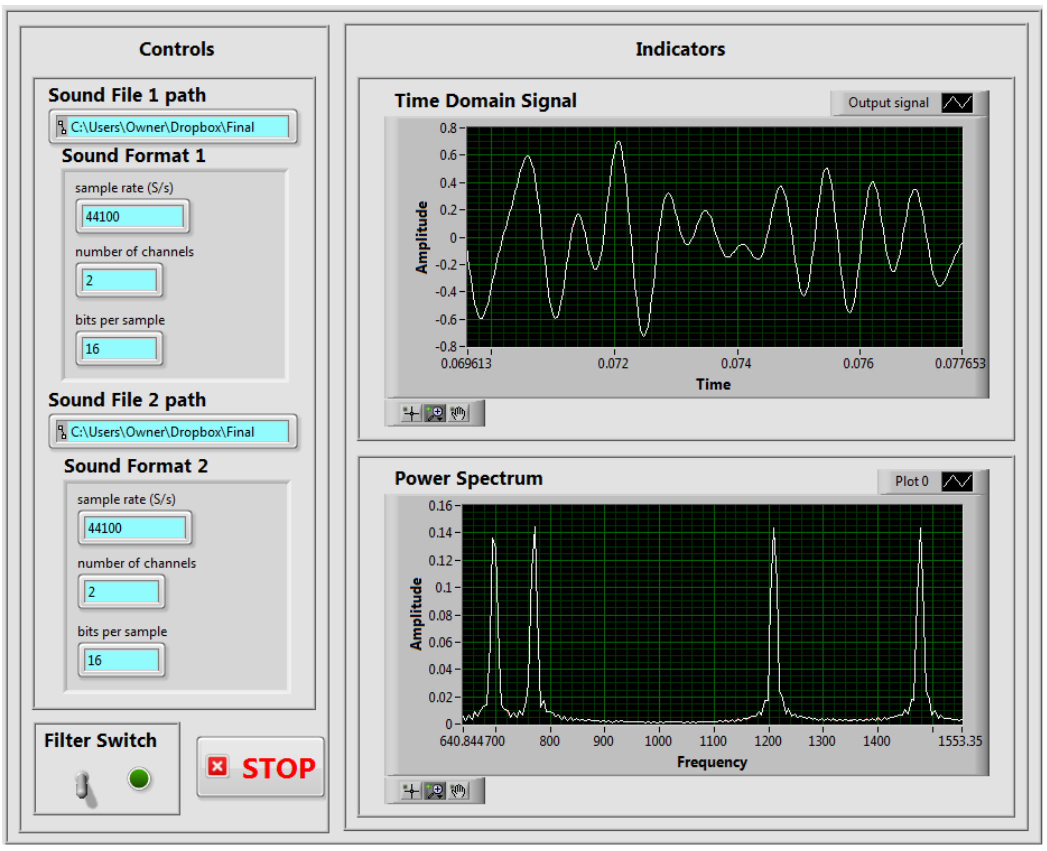
Fig. 37: High-pass Filter

4.4.1.3 Band-pass filter design

1. Open the given VI: DTMF_Demo_BPF.vi (Fig. 38a) and run it. Select "dtmf-3.wav" for the first file and "dtmf-4.wav" for the second. Observe the signal in in the time and frequency (Fig. 38b).



a) Block diagram

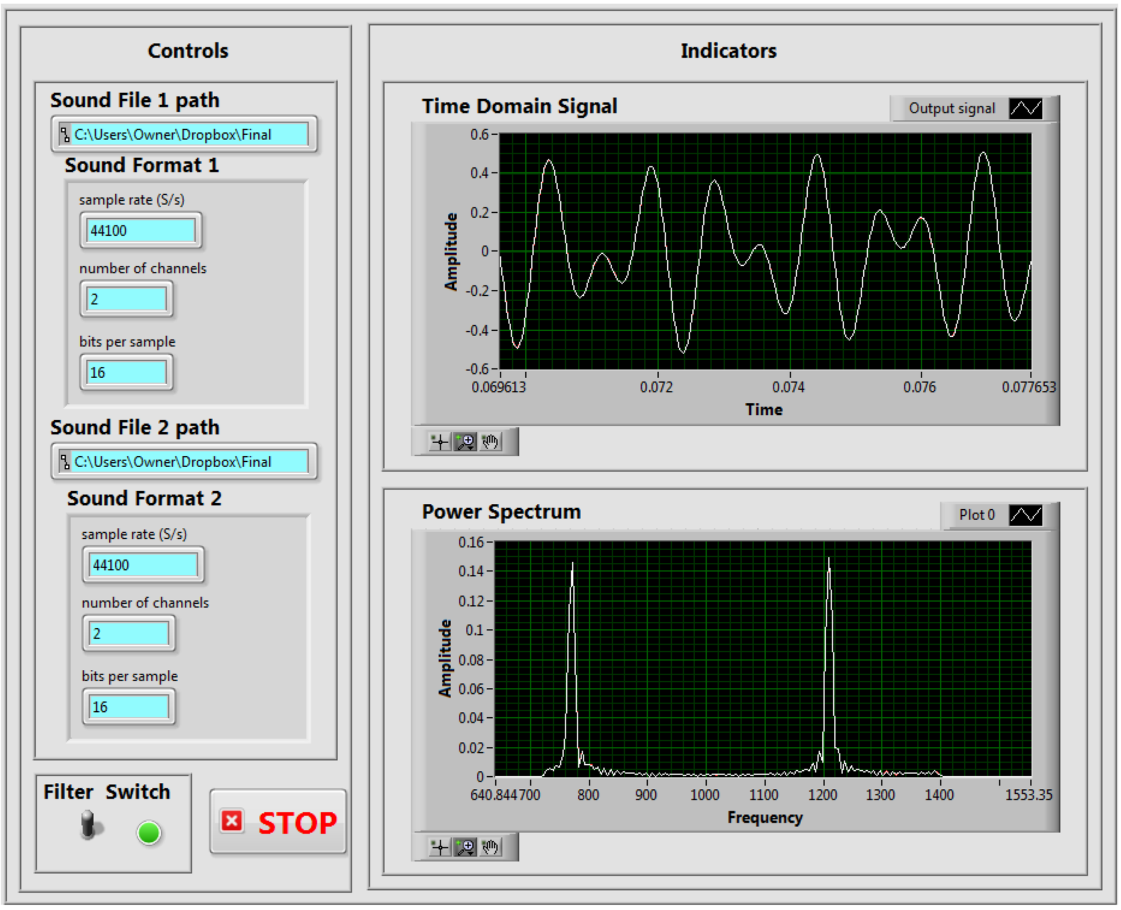


b) Front panel

Fig. 38: DTMF_Demo_BPF_Start VI

2. As before, Navigate to the block diagram for the filter, and remove the blocks from the "disabled" structure as directed.
3. Drag and drop the "Classical Filter" VI from the "disabled" structure into the "while" loop structure.

4. Double click on the “Classical Filter” VI and a pop-up window (Fig. 31) will appear. This pop-up window is your design tool screen where you enter the respective filter design specifications (Table V).
5. Drag and drop the “Filtering” sub-VI (Fig. 32) from the “disabled” structure into the “while” loop.
6. Wire the combined sound file and the designed filter to the filtering VI’s “Signal In” and “Filter” terminals respectively.
7. To analyze the performance of your filter, connect the filtered signal to the “Play Waveform Data” VI.
8. Save the modified VI as “DTMF_Demo_BPF_*YOUR_INITIALS*.vi”.
9. Next, run your VI. You must see that your input signal has been filtered: the outer frequencies are no longer present, and you have just the inner frequencies in the power spectrum (Fig. 39).


Controls

Sound File 1 path: C:\Users\Owner\Dropbox\Final

Sound Format 1
- sample rate (S/s): 44100
- number of channels: 2
- bits per sample: 16

Sound File 2 path: C:\Users\Owner\Dropbox\Final

Sound Format 2
- sample rate (S/s): 44100
- number of channels: 2
- bits per sample: 16

Indicators

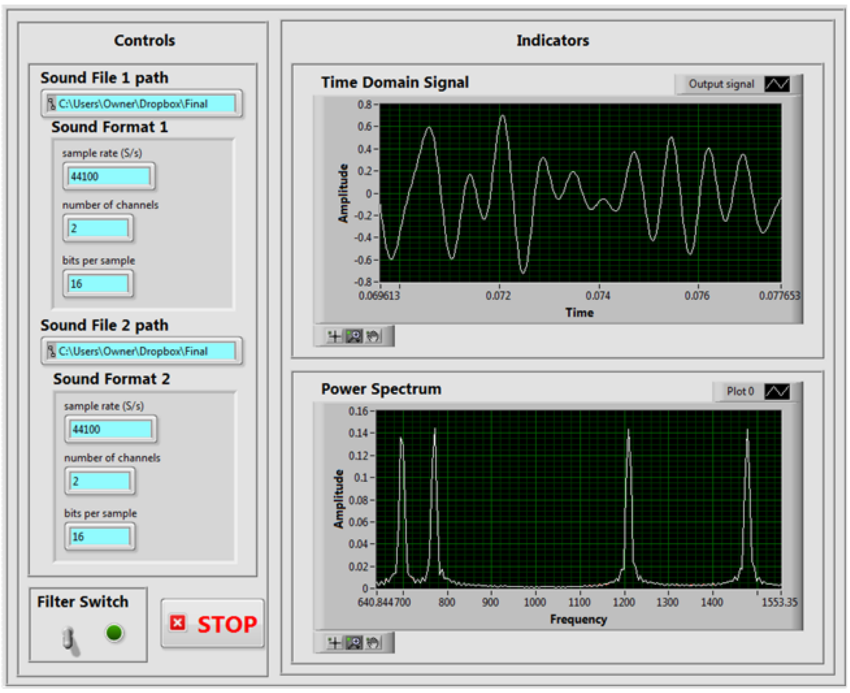
Fig. 39: Desired Band-pass Filter Result.

10. If you don’t get the right result, right click on the “Classical Filter” VI, select “Properties”, and change the filter specifications. Keep testing until you get it right!

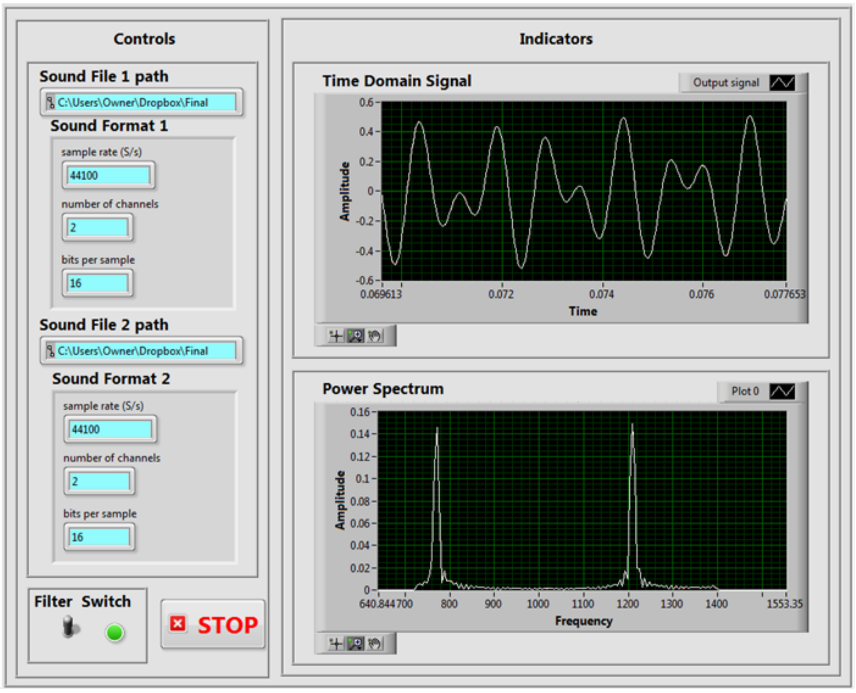
11. As before, drag and drop “Select” comparison VI, “Vertical Toggle” (switch) and a “Round LED” (LED indicator) (Fig. 34, Fig. 35) from the “disabled” structure into the “while” loop so that filtering is done only when the filter is turned on from the front panel. Verify your results for both cases (Fig. 40).

4.4.1.4 Band-stop filter design

1. Copy and paste the band-pass filter design block diagram to a new VI.
2. Save the VI as "DTMF_Demo_BSF_*YOUR_INITIALS*.vi".
3. Double click on the filter you designed to open the properties window (filter design tool screen). Change the filter type to band-stop and modify the other specifications to the ones you designed in the pre-lab session (Table V).
4. Run the VI and verify your results for both cases (Fig. 41).

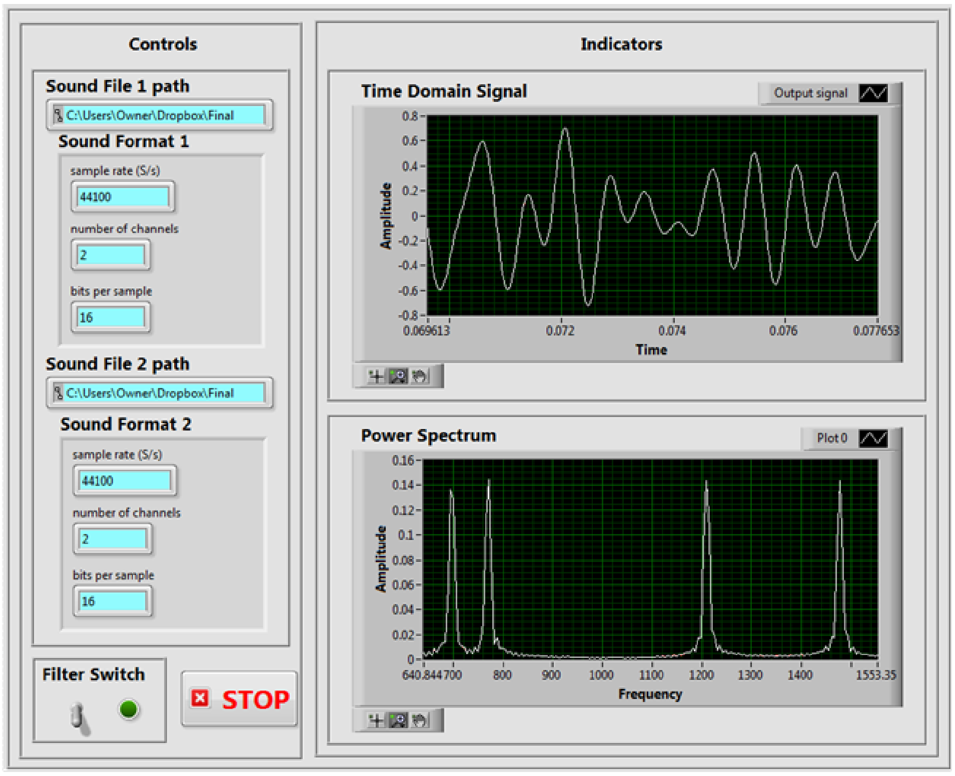

Controls
Sound File 1 path: C:\Users\Owner\Dropbox\Final
Sound Format 1: sample rate (S/s) 44100; number of channels 2; bits per sample 16
Sound File 2 path: C:\Users\Owner\Dropbox\Final
Sound Format 2: sample rate (S/s) 44100; number of channels 2; bits per sample 16

a) Filter off


Controls
Sound File 1 path: C:\Users\Owner\Dropbox\Final
Sound Format 1: sample rate (S/s) 44100; number of channels 2; bits per sample 16
Sound File 2 path: C:\Users\Owner\Dropbox\Final
Sound Format 2: sample rate (S/s) 44100; number of channels 2; bits per sample 16

b) Filter on

Fig. 40: Bandpass Filter


Controls

Sound File 1 path: C:\Users\Owner\Dropbox\Final

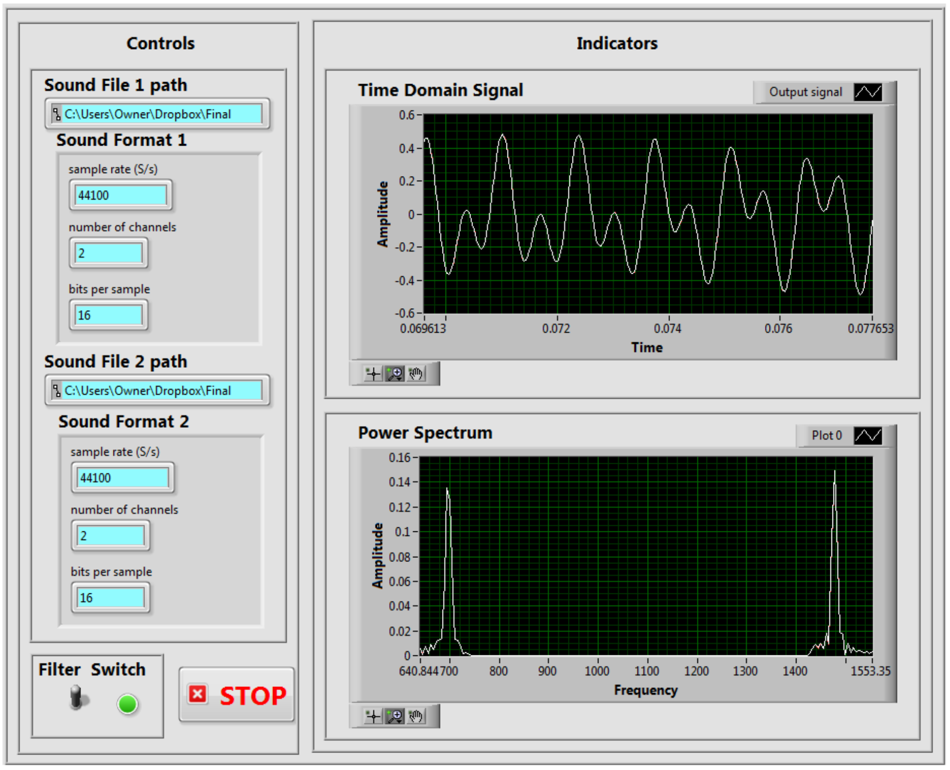
Sound Format 1
sample rate (S/s): 44100
number of channels: 2
bits per sample: 16

Sound File 2 path: C:\Users\Owner\Dropbox\Final

Sound Format 2
sample rate (S/s): 44100
number of channels: 2
bits per sample: 16

Filter Switch

a) Filter off


Controls

Sound File 1 path: C:\Users\Owner\Dropbox\Final

Sound Format 1
sample rate (S/s): 44100
number of channels: 2
bits per sample: 16

Sound File 2 path: C:\Users\Owner\Dropbox\Final

Sound Format 2
sample rate (S/s): 44100
number of channels: 2
bits per sample: 16

Filter Switch

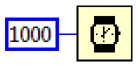
b) Filter on

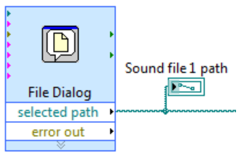
Fig. 41: Band-stop Filter

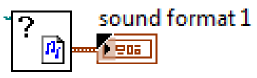
4.5 Lab Write-up

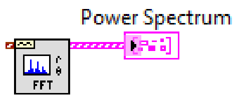
Performance Checklist
Digital Filter Design

Question: From any of the given sample VIs give the function of the following blocks:

a) 

b) 

c) 

d) 

Performance Measures

Task	Standards	Satisfactory/ Unsatisfactory
Description of above sample VIs	Quality of description: Scale (1-4)	
Filter Design	Functionality achieved: 1. Low-pass Filter (Yes/No) 2. High-pass Filter (Yes/No) 3. Band-pass Filter (Yes/No) 4. Band-stop Filter (Yes/No)	

Discussion
Have you started enjoying programming in LabVIEW?
Did all VIs perform as expected?
Did you have any difficulties completing the lab?
Did your TA provide enough guidance?
Do you have any recommendations to improve the lab?

4.6 References

[1] National Instruments, NI Manuals: Digital Filter Design Basics, retrieved August 23, 2014. http://zone.ni.com/reference/en-XX/help/371988F-01/TOC2.htm?nisrc=LV-2013.

5 Amplitude Modulation

5.1 Summary

This laboratory exercise has two objectives. The first is to gain experience in actually programming the USRP to act as a transmitter or a receiver. The second is to investigate classical analog amplitude modulation and the envelope detector.

5.2 Background

5.2.1 Amplitude Modulation

Amplitude modulation (AM) is one of the oldest of the modulation methods. It is still in use today in a variety of systems, including, of course, AM broadcast radio. In digital form it is the most common method for transmitting data over optical fiber [1].

If $m(t)$ is a baseband "message" signal with a peak value m_p, and $A_c \cos(2\pi f_c t)$ is a "carrier" signal at carrier frequency, f_c, then we can write the AM signal $g(t)$ as

$$g(t) = A_c\left[1 + \mu\frac{m(t)}{m_p}\right]\cos(2\pi f_c t) \tag{18}$$

where the parameter μ is called the "modulation index" and takes values in the range $0 < \mu \leq 1$ (0 to 100%) in normal operation. For the special case in which $m(t) = m_p \cos(2\pi f_c t)$ where f_m is the frequency of the message, we can write equation (1) as

$$\begin{aligned} g(t) &= A_c[1 + \mu\cos(2\pi f_m t)]\cos(2\pi f_c t) \\ &= A_c\left[\cos(2\pi f_c t) + \frac{\mu}{2}[\cos(2\pi[f_c - f_m]t) + \cos(2\pi[f_c + f_m]t)]\right] \end{aligned} \tag{19}$$

In the above expression the first term is the carrier, and the second and third terms are the lower and upper sidebands, respectively. Fig. 42 and Fig. 43 is a plot of a 20 kHz carrier modulated by a 1 kHz sinusoid at 100% and 50% modulation.

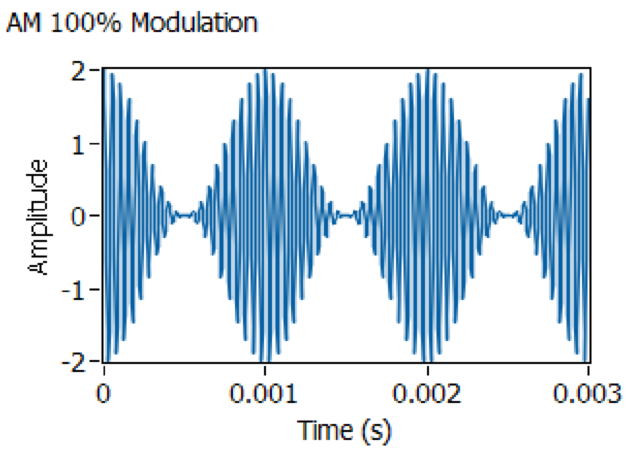


Fig. 42: AM Signal: Modulation Index = 1

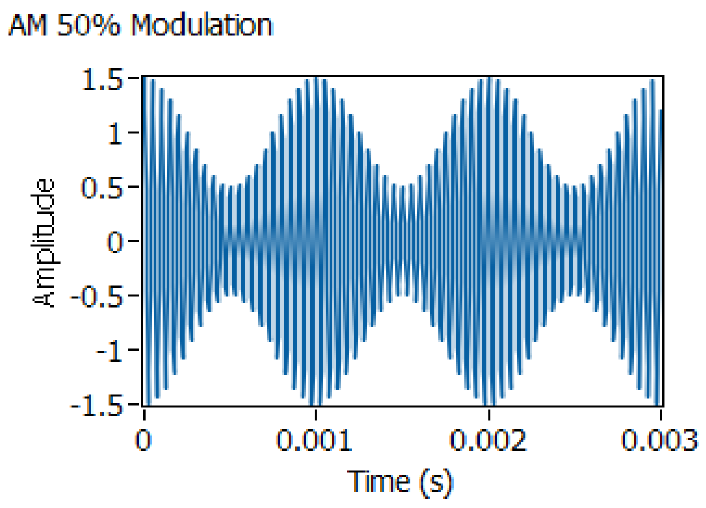


Fig. 43. AM Signal: Modulation Index = 0.5

When the AM signal arrives at the receiver, it has the form

$$r(t) = A_r \left[1 + \mu \frac{m(t)}{m_p}\right] \cos(2\pi f_c t + \theta) \tag{20}$$

where the angle θ represents the difference in phase between the transmitter and receiver carrier oscillators. We will follow a common practice and offset the receiver's oscillator frequency f_O from the transmitter's carrier frequency, f_c. This provides the signal

$$r_1(t) = A_r \left[1 + \mu \frac{m(t)}{m_p}\right] \cos(2\pi f_{IF} t + \theta) \tag{21}$$

where the so-called "intermediate" frequency (IF) is given by $f_{IF} = f_c - f_O$. The signal $r_1(t)$ can be passed through a bandpass filter to remove interference from unwanted signals on frequencies near f_c. Usually the signal $r_1(t)$ is amplified since $A_r < A_c$ due to signal attenuation as it moves through the transmission medium.

Demodulation of the signal $r_1(t)$ is most effectively carried out by an envelope detector. An envelope detector can be implemented as a rectifier followed by a lowpass filter. The envelope $A(t)$ of $r_1(t)$ is given by

$$A(t) = A_r \left[1 + \mu \frac{m(t)}{m_p}\right] = A_r + \frac{\mu A_r}{m_p} m(t) \tag{22}$$

5.3 Pre-Lab

5.3.1 Transmitter

The task is to add blocks as needed to produce an AM signal, and then to pass the AM signal into the *while* loop to the Write Tx Data block. A template for the transmitter has been provided in the file AM_Tx_Template.vi (Fig. 44). This template contains six interface controls, two waveform graphs to display your message signal and scaled amplitude modulated signal, and "message generator" controls set to produce a message signal consisting of three tones. The three tones are initially set to 1, 2, and 3 kHz, but these frequencies can be changed using the message generator front-panel controls.

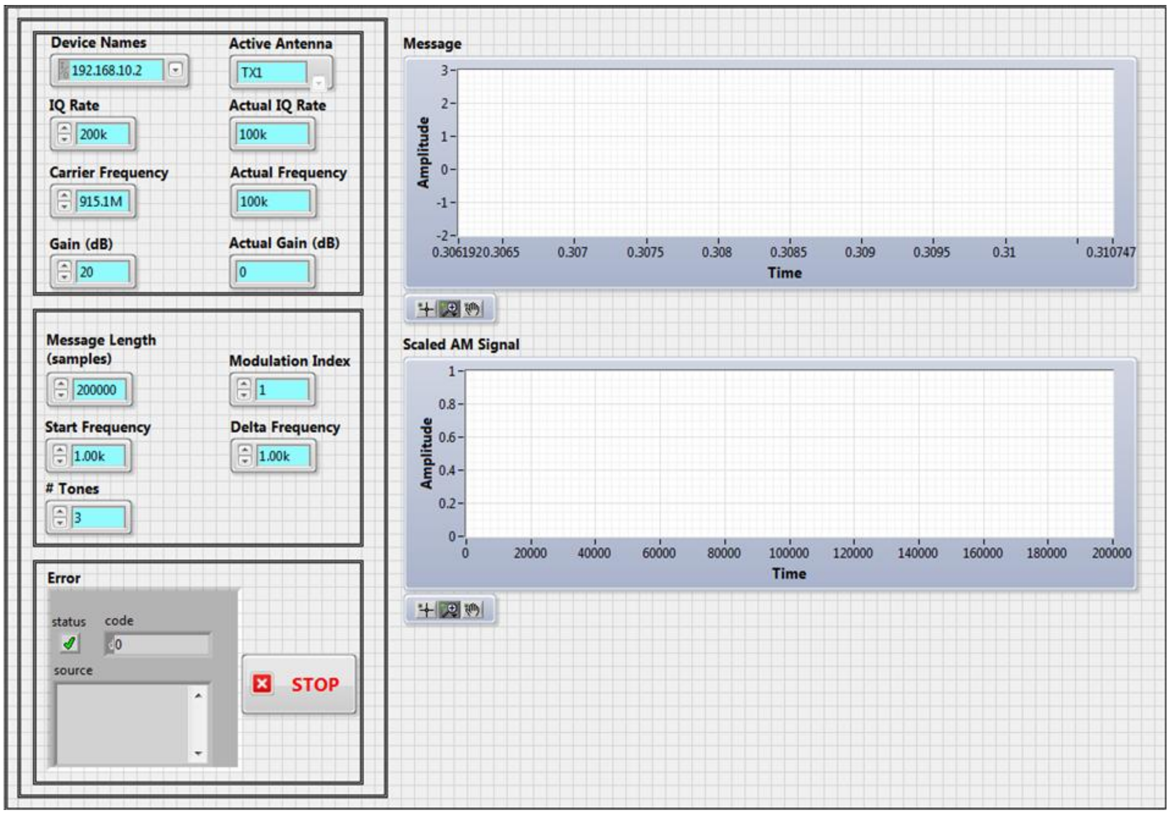


Fig. 44: AM_Tx_Template Front Panel

Tx Programming Notes:

a) Observe that the baseband signal $\tilde{g}(nT)$ is actually two baseband signals. By long-standing tradition, the real part $g_I(nT)$ is called the "in-phase" component of the baseband signal, and the imaginary part $g_Q(nT)$ is called the "quadrature" component of the baseband signal. The AM signal that you will generate in this lab project uses only the in-phase component, with

$$g_I(nT) = A_c\left[1 + \mu \frac{m(t)}{m_p}\right] \tag{23}$$

And

$$g_Q(nT) = 0 \tag{24}$$

You will explore other modulation methods in subsequent lab projects that use both components.

The baseband signal is expressed as

$$\tilde{g}(nT) = g_I(nT) + j\, g_Q(nT) \tag{25}$$

The signal transmitted by the USRP is

$$g(nT) = A_c\, g_I(nT) \cos(2\pi f_c t) + A_c\, g_Q(nT)\ \cos(2\pi f_c t) \tag{26}$$

These values are entered in the Tx Front Panel (Fig. 44) in the following fields

- f_c is the carrier frequency.
- Sampling interval T is the reciprocal of the "IQ rate."

Note that the signal $g(t)$ produced by the USRP is a continuous-time signal; the discrete-to-continuous conversion is done inside the USRP.

b) The message generator creates a signal that is the sum of a set of sinusoids of equal amplitude. You can choose the number of sinusoids to include in the set, you can choose their frequencies, and you can choose their common amplitude. The initial phase angles of the sinusoids are chosen at random, however, and will be different every time the VI runs. Get the data values of the generated signal by using the "Get Waveform Components" VI (Fig. 45) for amplitude modulation operations.

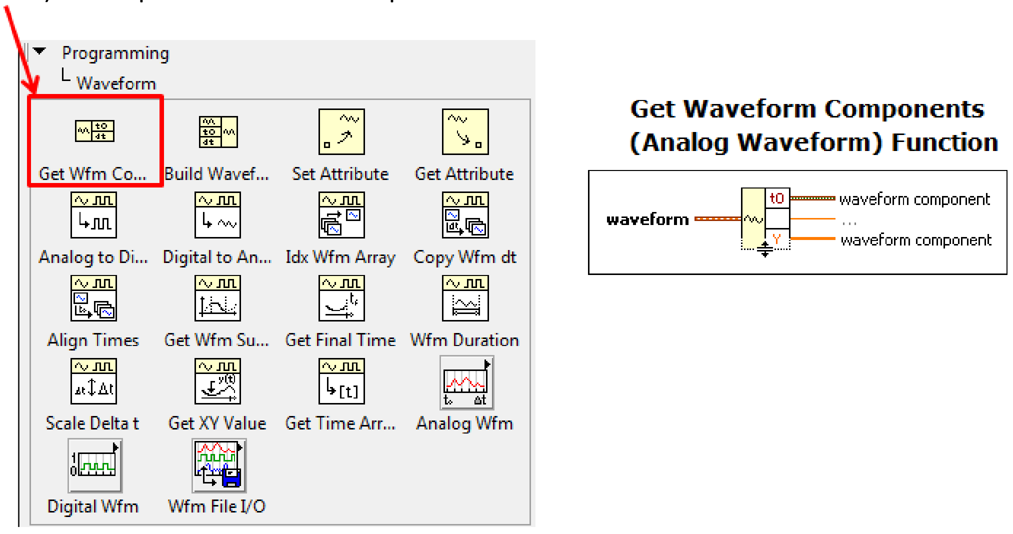
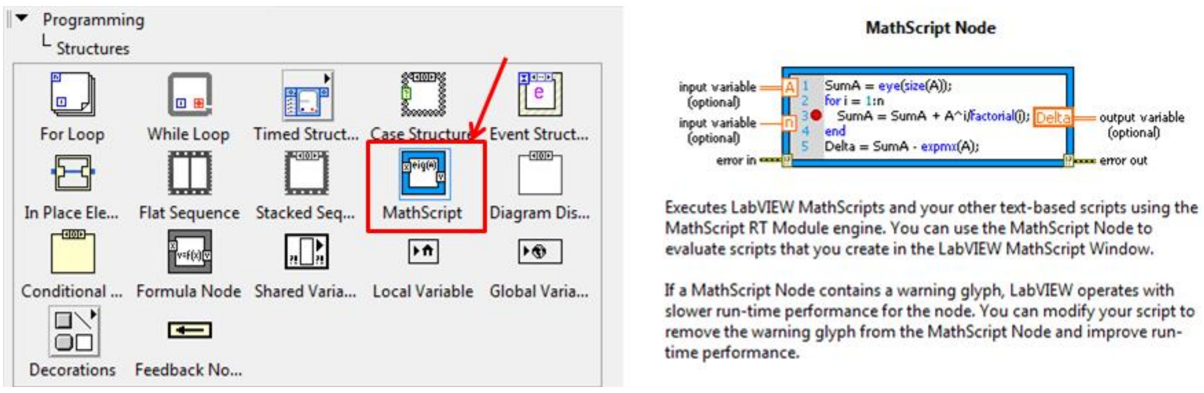


Fig. 45: Get Waveform Components VI

c) Set up a "MathScript Node" (Fig. 46) with data values of the generated signal {*m*}, maximum value of the generated signal {*mp*}, and modulation index {*mu*} as inputs. Use "Array Max and Min" VI (Fig. 47) to get the maximum value of the generated signal, and the "Modulation Index" control provided to set the modulation index {*mu*}. Use equations (23), (24), and (25) to set up the text-based script to get the baseband signal {*b*].

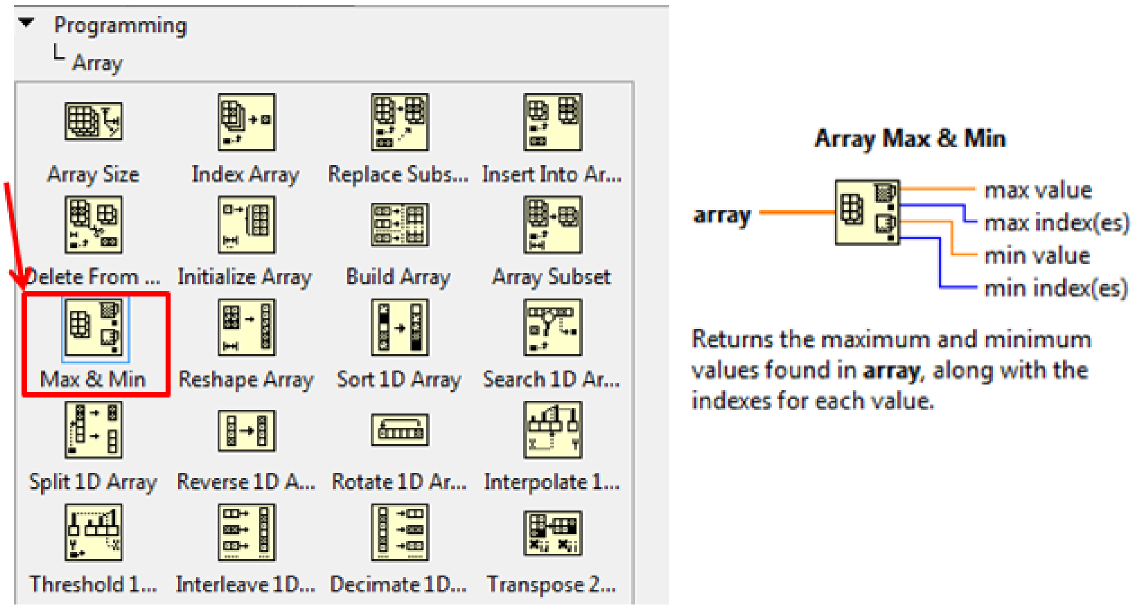


```
SumA = eye(size(A));
for i = 1:n
  SumA = SumA + A^i/factorial(i);
end
Delta = SumA - expm(A);
```

Executes LabVIEW MathScripts and your other text-based scripts using the MathScript RT Module engine. You can use the MathScript Node to evaluate scripts that you create in the LabVIEW MathScript Window.

If a MathScript Node contains a warning glyph, LabVIEW operates with slower run-time performance for the node. You can modify your script to remove the warning glyph from the MathScript Node and improve run-time performance.

Fig. 46: MathScript Node



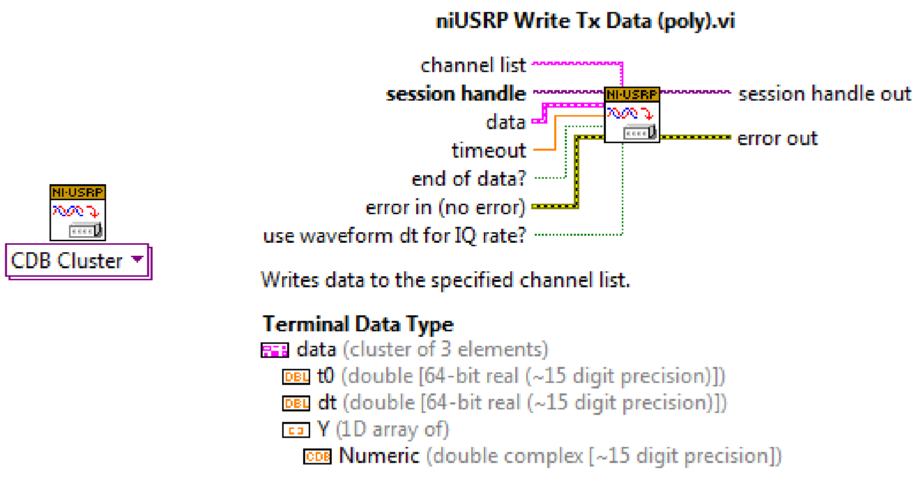
Returns the maximum and minimum values found in **array**, along with the indexes for each value.

Fig. 47: Array Max and Min VI

d) There is one practical constraint imposed by the D/A converters in the USRP: The maximum magnitude of the transmitted signal $|\tilde{g}(nT)|$ needs to have a maximum scaled value of 1. Set up a text-based script by dividing the baseband signal {*b*} by the maximum of its absolute value {*max(abs(b))*} to get the scaled baseband signal {*A*}.

e) The USRP is designed to transmit using a quadrature modulation approach. So in order to use the radio to transmit an AM signal, it is necessary to represent the signal as a complex sequence. The quadrature modulation then transmits the real and complex sequences using two orthogonal waveforms. The real part is sent using a cosine carrier and the complex part using a sine function as the carrier. Set up a text-based script to convert the scaled amplitude modulated signal from 1D double {*A*} to 1D complex double form {*G*}. The 1D complex double form is attained by multiplying the 1D double form by { $e^{(j*0)}$}.

f) Set up both the forms of the scaled baseband signal as outputs of the MathScript Node. Plot the scaled baseband signal {*A*} by using the "Baseband Signal" waveform graph provided, and input the complex form {*G*} to the "niUSRP Write Tx Data" VI (Fig. 48) to be transmitted.



niUSRP Write Tx Data (poly).vi

channel list
session handle
data
timeout
end of data?
error in (no error)
use waveform dt for IQ rate?
session handle out
error out

Writes data to the specified channel list.

Terminal Data Type
data (cluster of 3 elements)
t0 (double [64-bit real (~15 digit precision)])
dt (double [64-bit real (~15 digit precision)])
Y (1D array of)
Numeric (double complex [~15 digit precision])

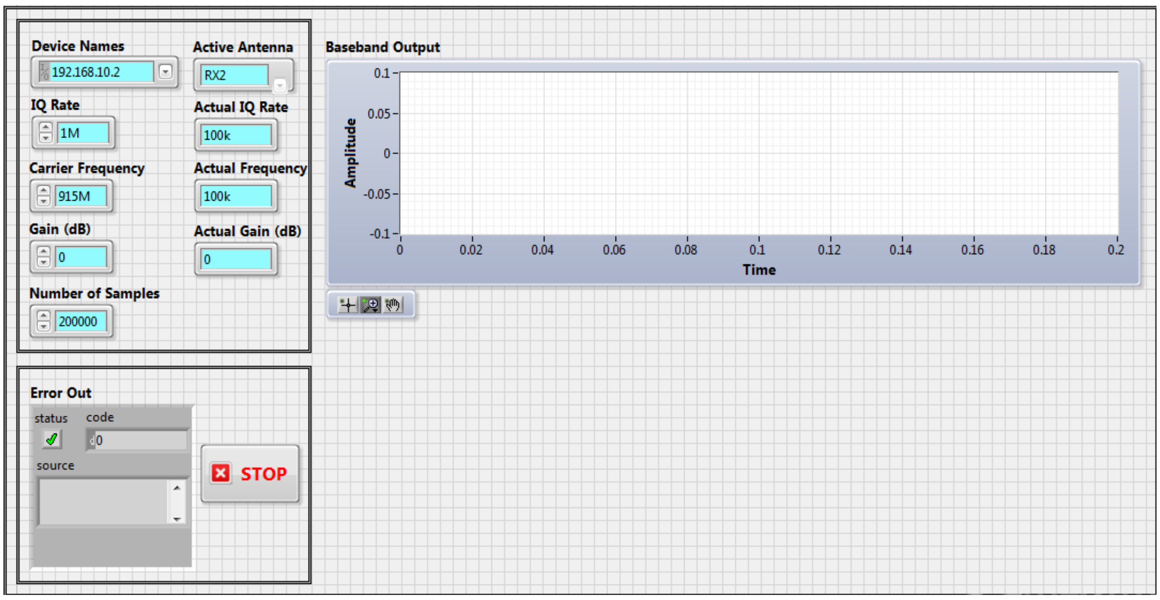
Fig. 48: niUSRP Write Tx Data VI

g) Save your transmitter in a file whose name includes the letters “AM_Tx” and your initials.

   Note: Modulation with the carrier occurs after the baseband signal is sent to the buffer for transmission. To visualize the amplitude modulated signal, you may plot the waveform received at the receiver end.

5.3.2 Receiver

A template for the receiver has been provided in the file AM_Rx_Template.vi (Fig. 49). This template contains the six interface controls and two waveform graphs to display the received amplitude modulated signal and the demodulated baseband output.



Device Names: 192.168.10.2
Active Antenna: RX2
IQ Rate: 1M
Actual IQ Rate: 100k
Carrier Frequency: 915M
Actual Frequency: 100k
Gain (dB): 0
Actual Gain (dB): 0
Number of Samples: 200000

Error Out
status: ✓
code: 0
source:

Fig. 49: Reciever VI Front Panel

Rx Programming Notes:

a) Plot the received amplitude modulated signal from the “niUSRP Fetch Rx Data” VI (Fig. 50) using the “Rx AM Signal” waveform graph provided.

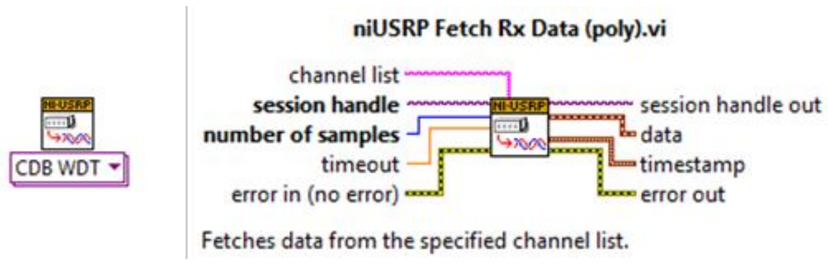


Fig. 50: niUSRP Fetch Rx Data VI

b) Get the data values of the signal received from the "niUSRP Fetch Rx Data" VI (Fig. 50)by using a "Get waveform components" VI (Fig. 45) so as to perform filtering operations.

c) To remove unwanted interferences around carrier frequency, design a fifth order "Chebyshev" band-pass filter (**Fig. 51**) with a high cutoff frequency of 105 kHz, a low cutoff frequency of 95 kHz, pass-band ripple of 0.1 dB, and a sampling frequency equal to the "actual IQ rate" obtained from the niUSRP Configure Signal VI.

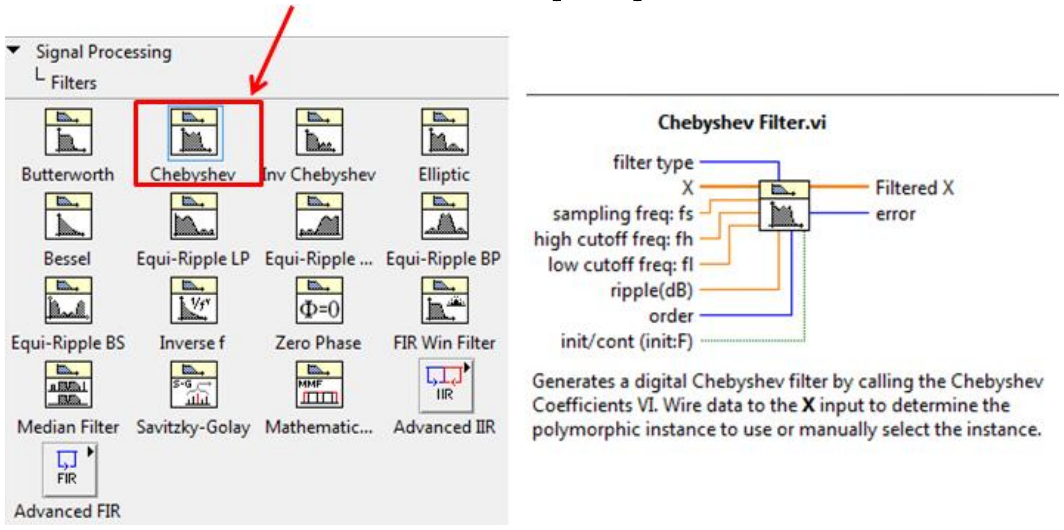


Fig. 51: Chebyshev Filter VI

d) Extract the real part of the complex filtered signal from the output of the Chebyshev band-pass filter using the "Complex to Real/Imaginary" VI (Fig. 52). The real part is expressed as shown in equation (21).

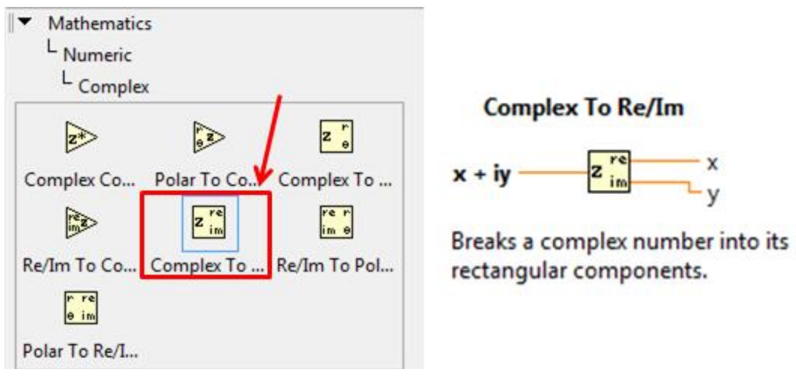


Fig. 52: Complex to Real/Imaginary VI

e) Use "Absolute Value" VI to take the absolute value of the real part of the filtered signal for full-wave rectification.

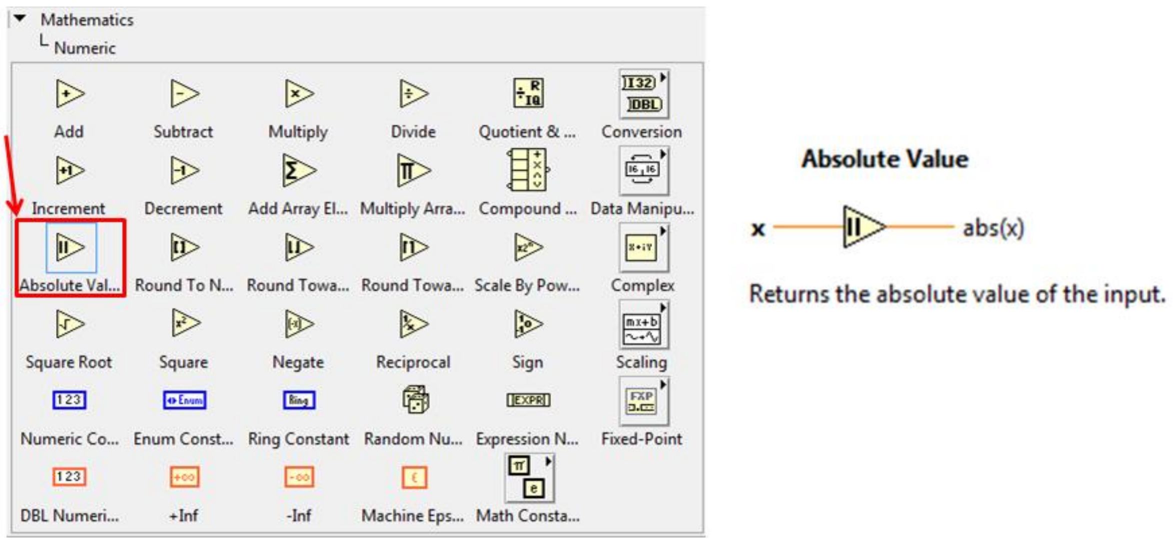


Fig. 53: Absolute Value VI

f) To filter out high frequencies to complete envelope detection, design a second order "Butterworth" low-pass filter (Fig. 54) with a low cutoff frequency of 5 kHz, and a sampling frequency the same as the "actual IQ rate" obtained from the niUSRP Configure Signal VI.

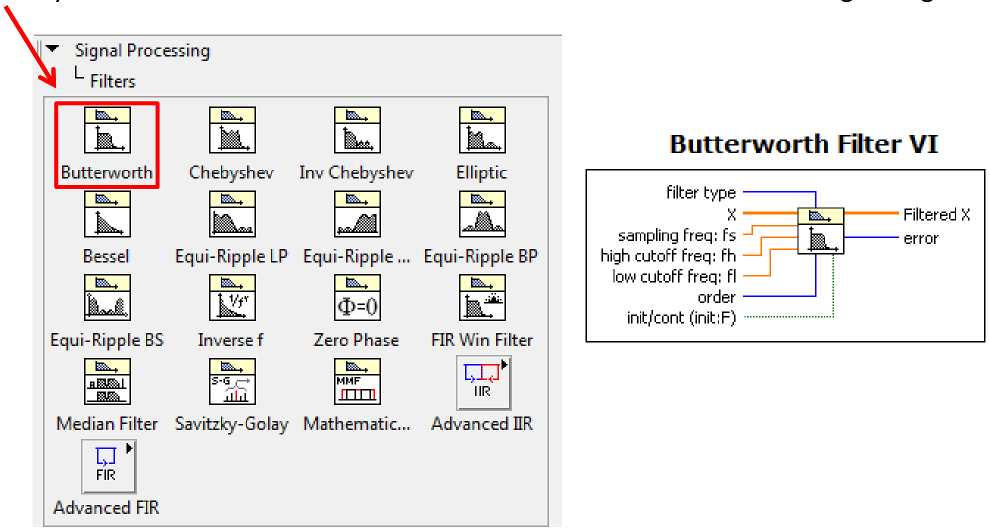


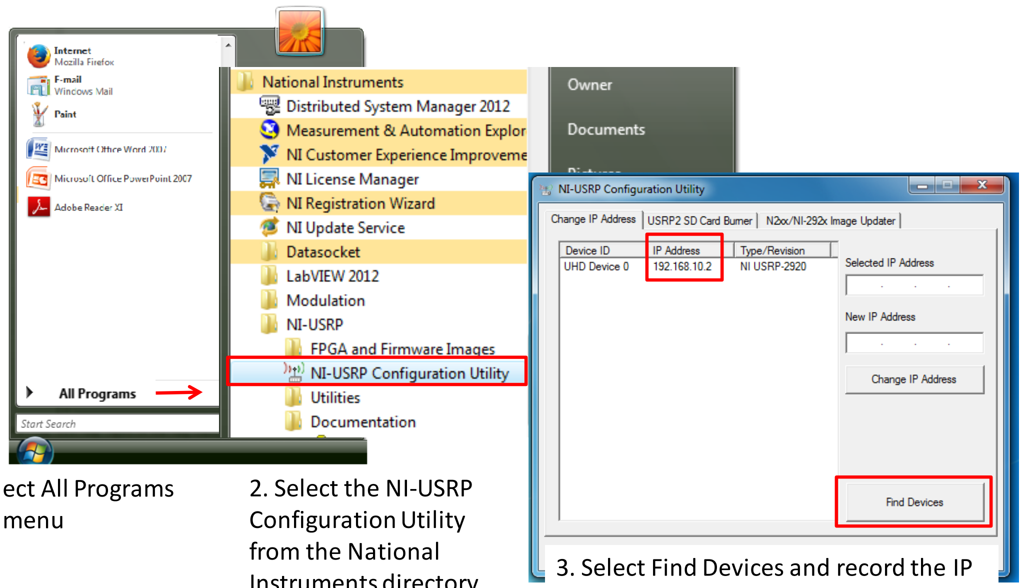
Fig. 54: Butterworth Filter VI

g) Build a waveform from the data values of the output of the low-pass filter designed above by using a "Build Waveform" VI, setting the sampling time interval same as that of the received waveform. Plot the waveform obtained with the "Baseband Output" waveform graph provided.

h) Save your receiver in a file whose name includes the letters "AM_Rx" and your initials.

5.4 Lab Procedure

1. Run LabVIEW and open the transmitter and receiver VIs that you created in the pre-lab.
2. Connect the computer to the USRP using an Ethernet cable.
3. Open the NI-USRP Configuration Utility found in the National Instruments directory under programs files as shown in Fig. 2. Be sure to record the IP addresses since you will need them to configure your software.



Device ID	IP Address	Type/Revision
UHD Device 0	192.168.10.2	NI USRP-2920

1. Select All Programs from menu

2. Select the NI-USRP Configuration Utility from the National Instruments directory

3. Select Find Devices and record the IP address of the radio or radios since you will need them to configure the software in the lab.

Fig. 55: Finding the IP Address: Radio Connectivity Test

If the IP address does not appear in the window then check your connections and ask the Teaching Assistant (TA) to verify that the LAN card has been configured correctly.

4. Connect a loopback cable between the TX 1 and RX 2 antenna connectors. Remember to connect the attenuator to the receiver end.

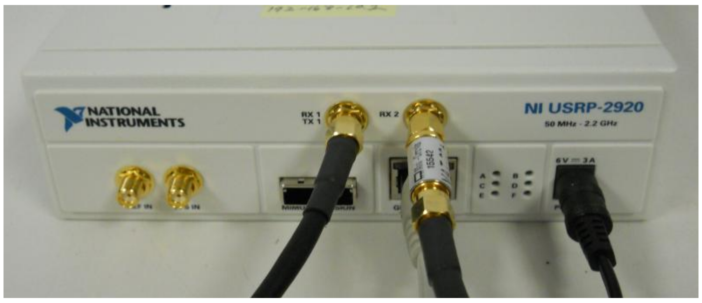


Fig. 56: Broadcast Setup

5. Ensure that the transmitter VI is set up according to Table VI.

Table VI – Transmitter Settings

Field	Setting	Field	Setting
Device Name	192.168.10.x	**Message Length**	200,000 samples
Carrier Frequency	915.1 MHz	**Modulation Index**	1.0
IQ Rate	200 kHz	**Start Frequency**	1 kHz
Gain	20 dB	**Delta Frequency**	1 kHz
Active Antenna	TX1	**Number of Tones**	3

Important set-up notes:

- ✓ Make sure the global set-up configuration has been performed before interfacing with the USRPs.
- ✓ Make sure the Tx and Rx VIs are always set to the same carrier frequency whenever you pair them up to communicate.
- ✓ Transmission should start only after receiving workstations are ready to receive.
- ✓ Verify that device name fields in both Tx and Rx VIs are set to the IP address of the URSP in use
- ✓ Make sure to connect the provided attenuator between the receiver USRP's Rx input and the antenna/loopback-cable. The attenuator is used to decrease the power level of the transmitted signal in order to avoid a high power signal at the receiver's end, due to Rx and Tx inputs' proximity to each other.

6. Run the transmitter VI. LED "A" will illuminate on the USRP if the radio is transmitting. Use zoom operations to check the message and scaled baseband waveforms on the transmitter VI front panel.
7. Stop the transmission by using the large **"STOP"** button on the front panel.
   Note: Using the "STOP" button on front panel rather than stopping from the "Abort Execution" button on the menu bar ensures that the USRP is stopped cleanly.
8. Ensure that the receiver VI is set up according to Table VII.

Table VII – Receiver Settings

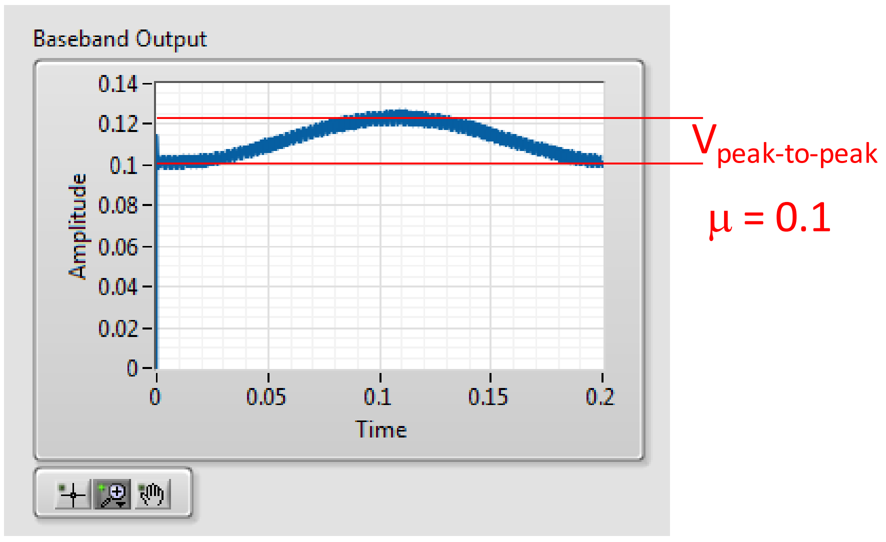
Field	Setting
Device Name:	192.168.10.x
Carrier Frequency:	915 MHz
IQ Rate:	1 MHz
Gain:	0 dB
Active Antenna:	RX2
Number of Samples:	200,000 samples

9. Run the receiver VI. LED "C" will illuminate on the USRP if the radio is receiving data.
10. Next, run the transmitter.
11. Use zooming operations from the graph palette to zoom into the "Rx AM Signal" and "Baseband Output" waveforms on the receiver front panel. The demodulated AM waveform "Baseband Output" should be identical to the "Baseband Signal" waveform, except for scaling (receiver output has a DC offset) and marginal noise.

5.4.1 Worksheet: The Effect of Varying the Modulation Index

1. Set the transmitter to use one of the three tones. Please note that using more than one tone will make it very hard to make the observations.
2. Set the Start Frequency to 1 kHz.
3. Set the transmitter VI modulation index to the first value in Table VIII.
4. Start the transmitter VI.
5. Observe the demodulated signal i.e. "Baseband Output" waveform on the receiver VI. Note the peak to peak voltage in Table VIIITable IX.
6. Stop the receiver VI. Update the modulation index to the next value in Table VIII and repeat steps 4 through 6 until the table is complete.

Table VIII – Modulation Index Observations

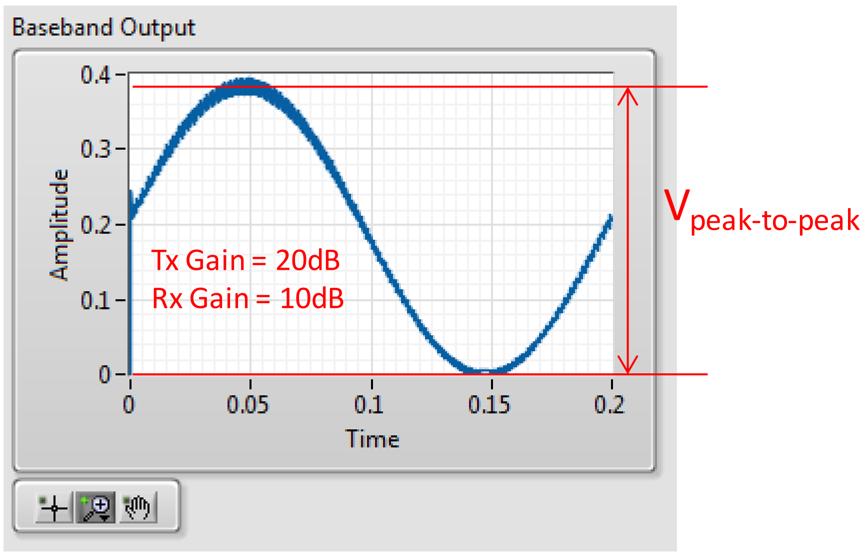
Modulation Index	***Amplitude (Peak to Peak)***	
0.1		 Baseband Output Amplitude 0.14 0.12 0.1 0.08 0.06 0.04 0.02 0 0.05 0.15 0.2 Time $V_{peak-to-peak}$ $\mu = 0.1$
0.2		
0.3		
0.4		
0.5		
0.6		
0.7		
0.8		
0.9		
1.0		

5.4.2 Worksheet: The Effect of Varying the Receiver Gain.

Warning: Too much receiver gain will overload the receiver A/D converters.

1. Set the transmitter to use one of the 3 tones. Please note that using more than one tone will make it very hard to make the observations.
2. Set the transmitter VI gain to 20 dB.
3. Set the receiver VI gain to the first value in Table IX.
4. Run the receiver VI, and then the transmitter VI.
5. Observe the demodulated signal i.e. "Baseband Output" waveform. Note the peak to peak voltage in Table IX.
6. Stop the receiver VI. Update the receiver gain to the next value in Table IX and Repeat steps 4 through 6 until the table is complete.

Table IX – Receiver Gain Observations

Receiver Gain (dB)	***Voltage (Peak-to-Peak)***	
0		 Baseband Output Amplitude 0.4 0.3 0.2 0.1 0 0 0.05 0.1 0.15 0.2 Time Tx Gain = 20dB Rx Gain = 10dB $V_{peak-to-peak}$
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		

5.5 Lab Write-up

Performance Checklist
Amplitude Modulation

Short Answer Questions

1. What is the relation between the message bandwidth and the IF and baseband filter bandwidths?

2. What is the effect of varying the modulation index?

3. What is the effect of varying the transmitter and receiver gain?

Performance Measures

Task	Standards	Sat/Unsat
Hardware Setup	Working setup for all with Loopback-cable.	
Running VIs	Successful transmission and reception of tones.	
Data Collection	Collect data to answer Short Answer Questions.	

Discussion
Did all configurations perform as expected?
Did you have any difficulties completing the lab?
Did your TA provide enough guidance?
Do you have any recommendations to improve the lab?

5.6 References

[1] Lab 2: Amplitude Modulation, Bruce A. Black, Rose-Hulman Institute of Technology, July 2013.

6 Frequency Modulation

6.1 Summary

This laboratory exercise introduces frequency modulation (FM) with two objectives. The first is to provide you an experience in programming the USRP to act as an FM transmitter or a receiver. The second is to investigate demodulating FM in software which is much simpler than demodulation procedures in the traditional hardware approach.

6.2 Background

6.2.1 Frequency Modulation

Frequency modulation (FM) was introduced by E.A. Armstrong in the 1930's as an alternative to the AM commonly in use at the time for broadcasting. The advantage to frequency modulation is that, for a given transmitted power, the signal-to-noise ratio is much higher at the receiver output than it is for AM. The digital version of FM, frequency-shift keying, has been in use since an even earlier date [2].

Of the two forms of angle modulation- frequency modulation (FM) and phase modulation (PM), this lab would focus on FM modulation, and provide you with a nice illustration of the utility of the easier to implement software-defined radio approach. The general angle of the modulated signal is the output ($x_c(t)$) in Fig. 57 and Fig. 58.

$$x_c(t) = A_c \cos\left(2\pi f_c t + \theta(t)\right) \tag{27}$$

where A_c is the amplitude of the carrier signal and f_c is the carrier frequency.

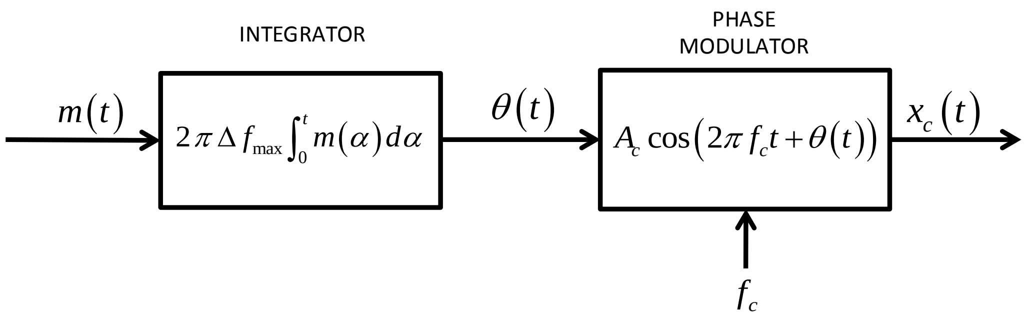

INTEGRATOR
PHASE MODULATOR
$m(t)$
$2\pi \Delta f_{\max} \int_0^t m(\alpha) d\alpha$
$\theta(t)$
$A_c \cos(2\pi f_c t + \theta(t))$
$x_c(t)$
f_c

Fig. 57: Generating a FM Signal Using a Phase Modulator

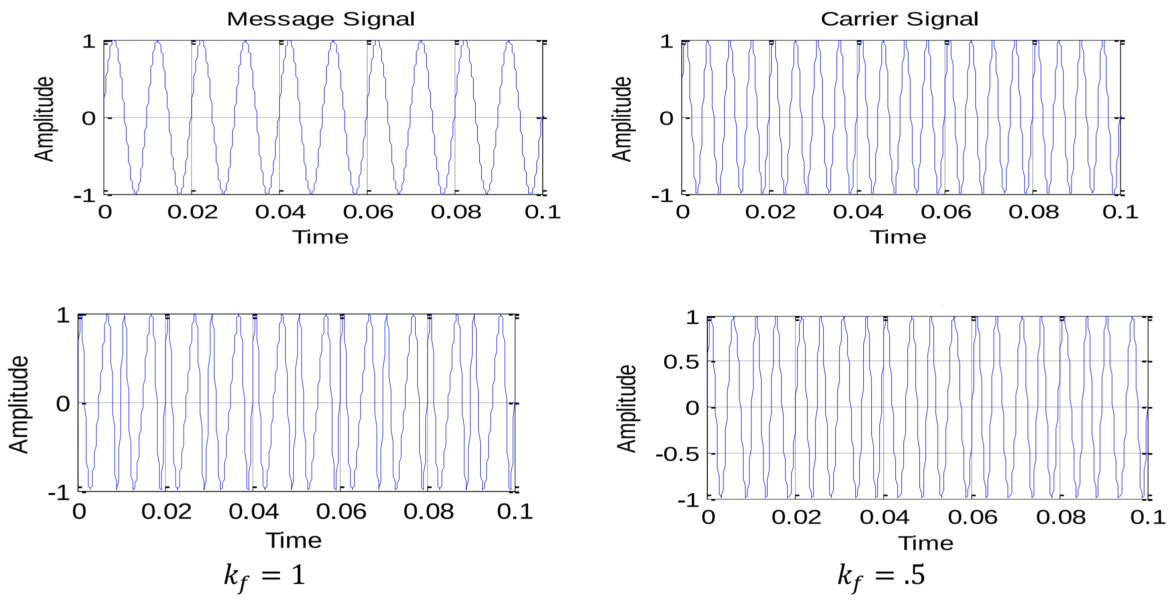


Fig. 58: FM Waveforms

In FM, the instantaneous phase $\theta(t)$ of the carrier is varied linearly with the message signal $m(t)$. Thus, the instantaneous frequency, $f_i(t)$, of the carrier signal varies linearly with the message signal $m(t)$ and can be written

$$f_i(t) = f_c + k_f m(t) \tag{28}$$

where k_f is the "frequency sensitivity" of the FM modulator. The maximum instantaneous frequency change from the carrier frequency produced by the modulating signal is given by

$$\Delta f_{max} \triangleq \max\ |f_i(t) - f_c| = \ k_f \max\ |m(t)| \tag{29}$$

Given the instantaneous frequency, we can find the total instantaneous angle, $\phi(t)$, of the carrier by integrating the instantaneous frequency as

$$\phi(t) = 2\pi \int_0^t f_i(\alpha)d\alpha = 2\pi f_c(t) + 2\pi\Delta f_{max} \int_0^t m_n(\alpha)d\alpha \tag{30}$$

where $m_n(t)$ is the normalized message signal, defined as

$$m_n(t) = \frac{m(t)}{\max\ |m(t)|} \tag{31}$$

Using Equation (30), the FM signal can be expressed in terms of its instantaneous frequency and we can re-write Equation (27) for the FM signal $x_c(t)$, in terms of its instantaneous frequency.

$$x_c(t) = A_c \cos\left(2\pi f_c t + 2\pi\Delta f_{max} \int_0^t m_n(\alpha)d\alpha\right) \tag{32}$$

Note that the second term in (32) represents $\theta(t)$ of the FM signal. Thus, the instantaneous phase is the integral of a normalized message multiplied by a peak frequency deviation (Δf_{max}).

To create an FM signal using the USRP, the complex-valued signal $\tilde{g}(t)$ is formed, where

$$\tilde{g}(t) = A_c \ exp\left(2\pi f_c t + j \left[2\pi \, \Delta f_{max} \int_0^t m_n(\alpha) d\alpha\right]\right) \tag{33}$$

When generating a PM signal, the process is similar to that illustrated in Fig. 57, but it requires a differentiator and a frequency modulator, rather than an integrator and a phase modulator.

6.2.2 Frequency Demodulation

An FM demodulator recovers the message signal from the received FM waveform. This requires a circuit that produces an output that is linearly proportional to the instantaneous frequency of the input FM signal. FM demodulation is much easier to carry out using the USRP rather than conventional hardware. FM demodulation can be divided into three broad categories: Frequency discrimination, Phase-shift discrimination, and Phase-locked loop (PLL). This lab focuses solely on frequency discrimination as illustrated in Fig. 59.

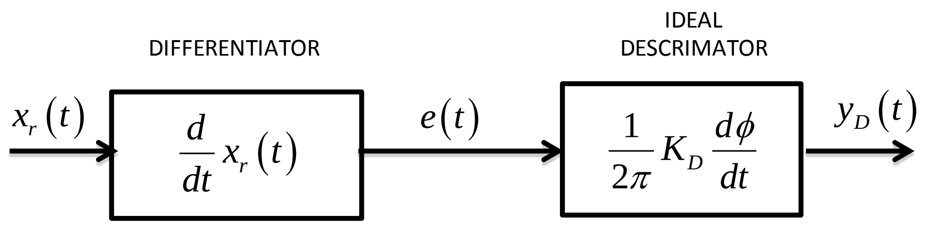


Fig. 59: FM Discriminator

The signal $x_r(t)$ received ideally is the same as the transmitted signal is the same as $\tilde{g}(t)$,

$$x_r(t) = \ A_c \exp(j \ [2\pi f_c t + \theta(t)]) \tag{34}$$

The real component of differentiator's output (Fig. 59) is given in (35),

$$e(t) = -K_D \left[2\pi f_c + \frac{d\theta}{dt}\right] \sin\left(2\pi f_c t + \theta(t)\right) \tag{35}$$

where $K_D = 2\pi A_c$ is the frequency discriminator gain constant, expressed in volts/Hz.
And the output of the ideal discriminator is

$$y_D(t) = \frac{1}{2\pi} K_D \frac{d\phi}{dt} \tag{36}$$

where

$$\phi(t) = 2\pi f_c t + 2\pi \Delta f_{max} \int_0^t m_n(\alpha) d\alpha \tag{37}$$

This means that we can write Equation (36) as

$$y_D(t) = 2\pi f_c + K_D \Delta f_{max} m_n(t) \tag{38}$$

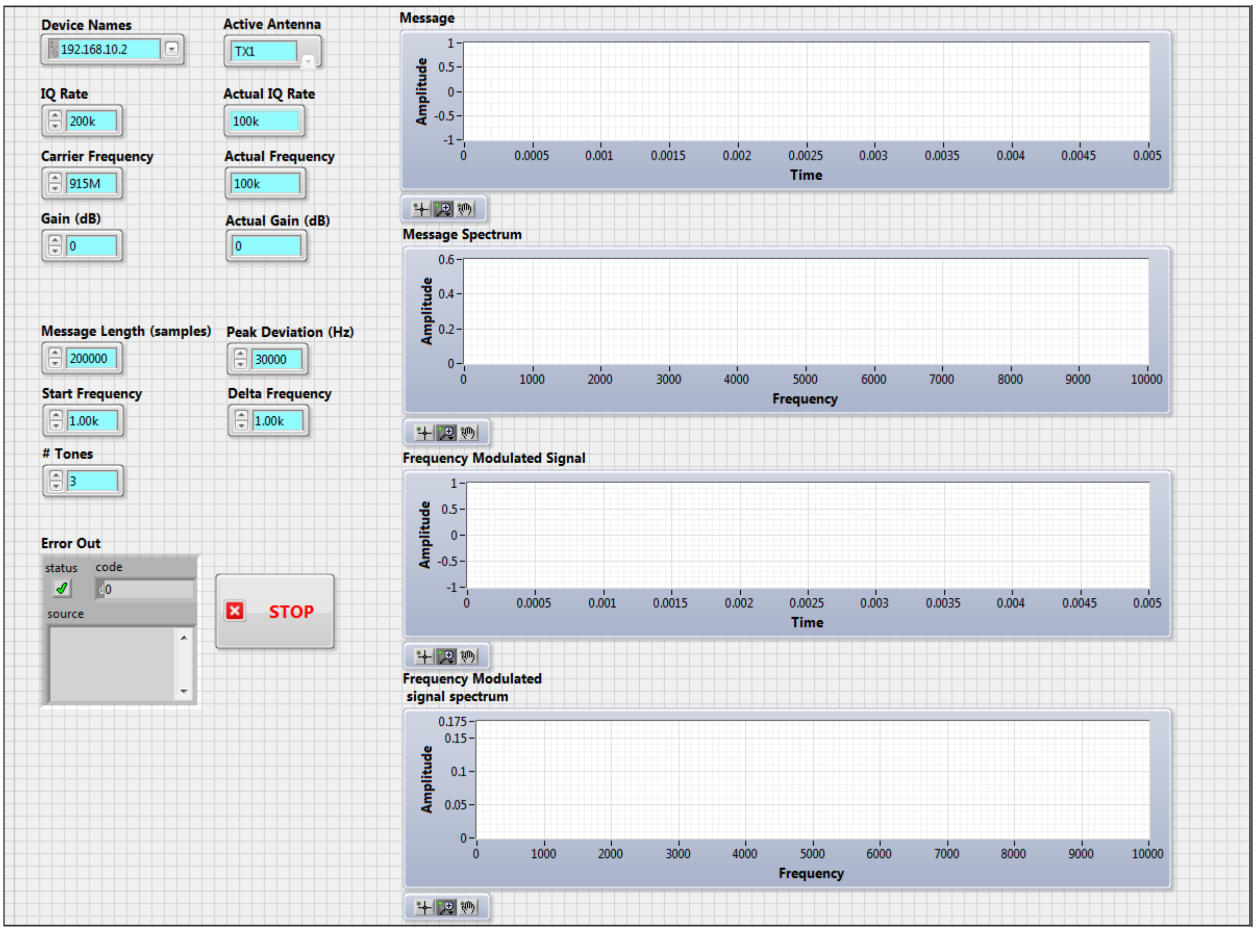
In practice, $f_c \gg d\theta/dt$, so $f_c + d\theta/dt$ is always positive.

6.3 Pre-lab

Your pre-lab task is to design the transmitter VI for frequency modulating a given message signal.

6.3.1 Transmitter

A template for the transmitter has been provided in **FM_Tx_Template.vi**. The Basic Multi-tone VI is used as the "message generator" to produce a message signal $m(t)$ consisting of a chosen number of tones, start frequency (frequency of first tone) and delta frequency (increment in frequency for next tones w.r.t. frequency of first tone). These values can be selected using front-panel controls. Fig. 60 shows the front panel of FM_Tx_Template.vi.



Device Names	192.168.10.2
Active Antenna	TX1
IQ Rate	200k
Actual IQ Rate	100k
Carrier Frequency	915M
Actual Frequency	100k
Gain (dB)	0
Actual Gain (dB)	0
Message Length (samples)	200000
Peak Deviation (Hz)	30000
Start Frequency	1.00k
Delta Frequency	1.00k
# Tones	3

Error Out: status ✓, code 0, source

Fig. 60: Front Panel of FM_Tx_Template.vi

Your task is to add blocks as needed to produce the frequency modulated signal ($\tilde{g}(t)$ of Equation (33)), and then to pass this signal to the "Write Tx Data block".

Step-by-step LabVIEW setup to create a FM modulated signal using USRP:

a) Obtain the data values and sampling time interval of the analog message signal $m(t)$ using "Get Waveform Components" VI (Fig. 61), and get the normalized message signal $m_n(\alpha)$ by using the "Quick Scale" VI (Fig. 62).

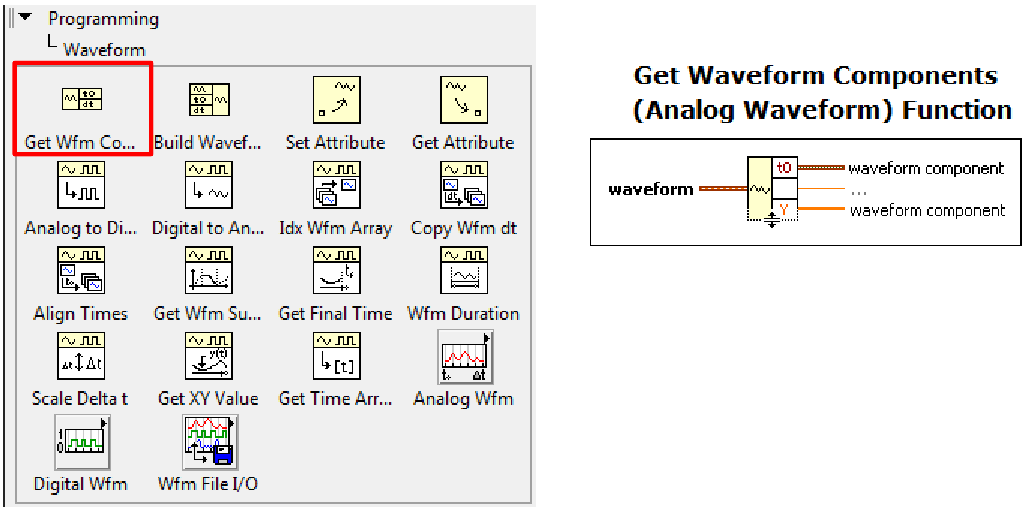


Fig. 61: Get Waveform Components VI

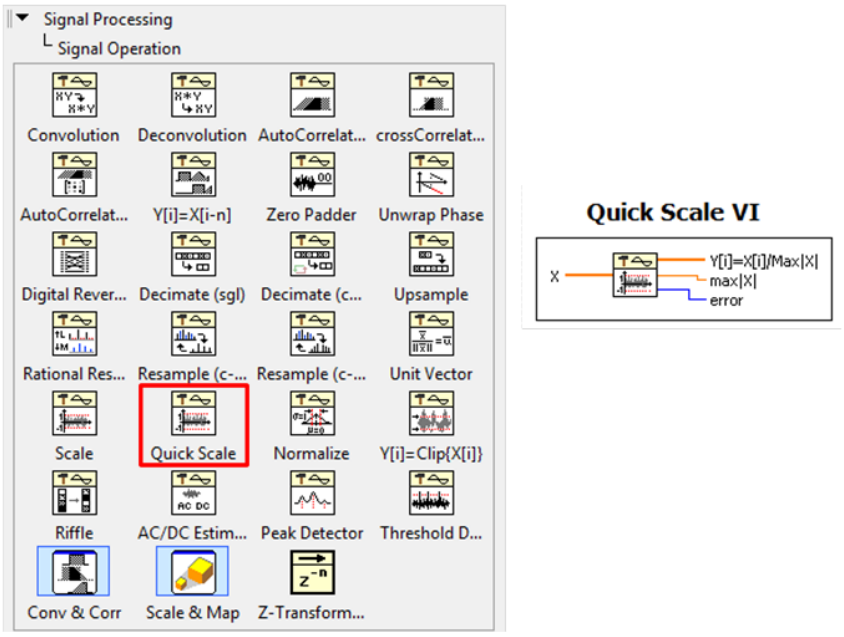


Fig. 62: Quick Scale VI

b) Convert the data values of the scaled message signal into a waveform using "Build waveform" VI (Fig. 63) by setting the sampling time same as that of the original message signal. Plot the scaled message waveform using the waveform chart provided. This would allow you to do a comparison with the demodulated signal waveform charts in the receiver. You may do a similar comparison for the spectrum of the message waveform and its demodulated version. To get the message spectrum, connect the message waveform (original or scaled) to the FFT Power Spectrum and PSD VI (Fig. 8) found in Signal Processing palette, and then connect the output to a waveform graph.

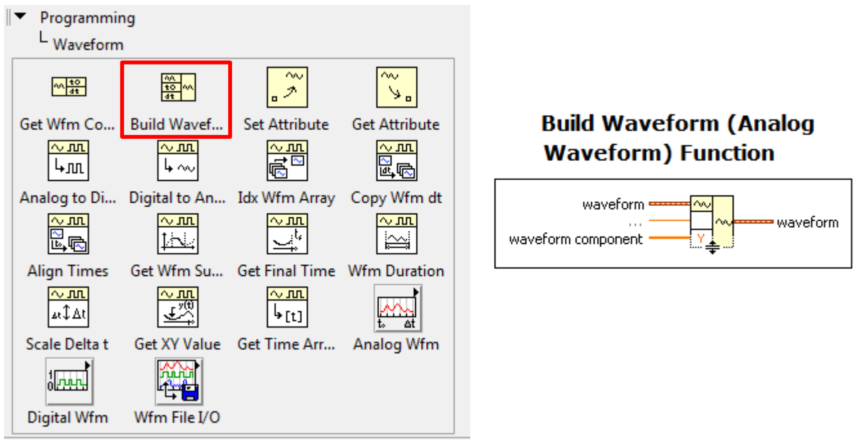


Fig. 63: Build Waveform VI

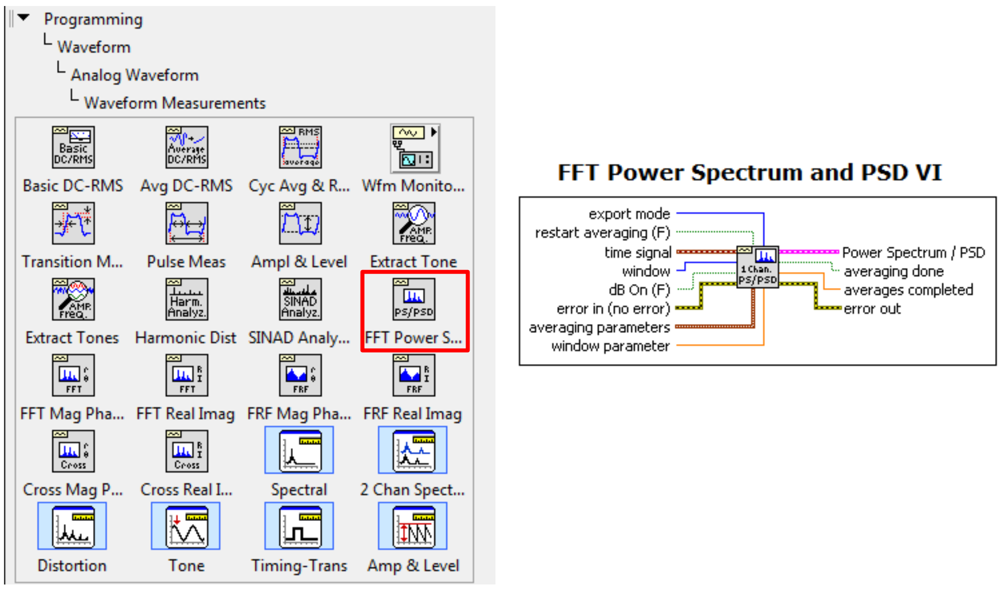


Fig. 64: FFT Power Spectrum and PSD VI

c) Next, you have to convert the normalized analog signal $m_n(\alpha)$ to discrete time before multiplying by $2\pi\Delta f_{max}$T in equation (33). We have,

$$\int_0^t m_n(\alpha)d\alpha = \sum_{k=0}^{n} m_n(nT)T, \tag{39}$$

where T is the reciprocal of the IQ sample rate. If we consider the equation

$$y[n] = \sum\nolimits_{k=0}^{n} m_n(nT)T, \tag{40}$$

then

$$y[n] - y[n-1] = \sum\nolimits_{k=0}^{n} m_n(nT)T - \sum\nolimits_{k=0}^{n-1} m_n(nT)T = x(nT)T \tag{41}$$

Equation (41) is a recurrence relation, or a difference equation of an IIR filter with "forward coefficients" array of $[1]$ and a "reverse coefficients" array of $[1\ -1]$, and can be implemented using "IIR filter with initial conditions" VI (Fig. 65) inside the while loop.

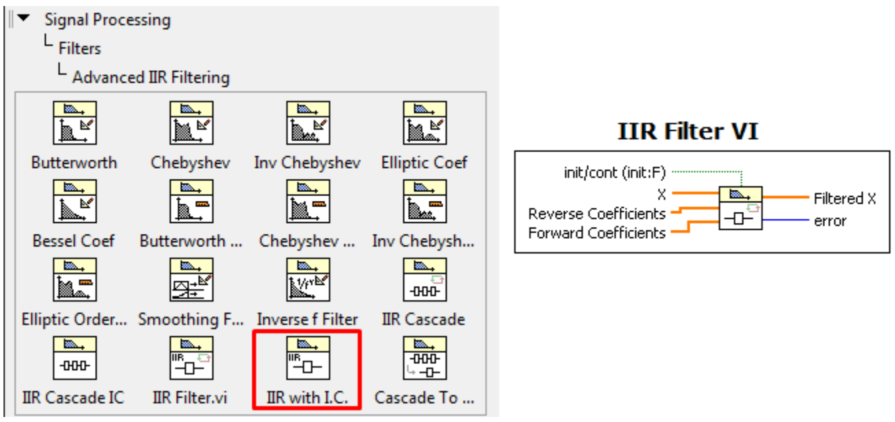


Fig. 65: IIR Filter with Initial Conditions VI

Use shift registers to feedback the final conditions of current loop iteration to be the initial conditions of the next loop iteration for the IIR filter. Initialize shift registers to a zero constant.

d) Multiply the discrete filtered $m_n(\alpha)$ by $2\pi\Delta f_{max}$T, which produces

$$\theta(t) = 2\pi\Delta f_{max} \sum_{k=0}^{n} m_n(nT)T \tag{42}$$

e) Form the complex value frequency modulated signal $\tilde{g}(t)$ (33) by utilizing the "Polar to Complex" VI (Fig. 66). In this case, inputs to the polar to complex VI "**r**" and **theta** are A_c and $\theta(t)$ respectively.

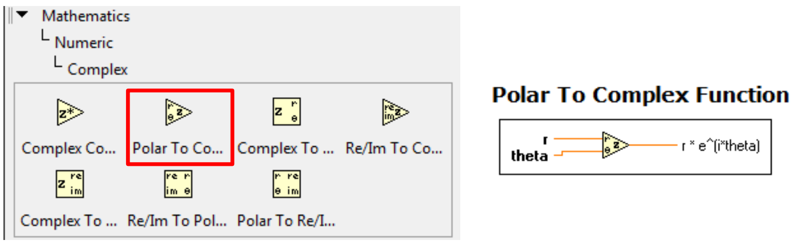


Fig. 66: Polar To Complex VI

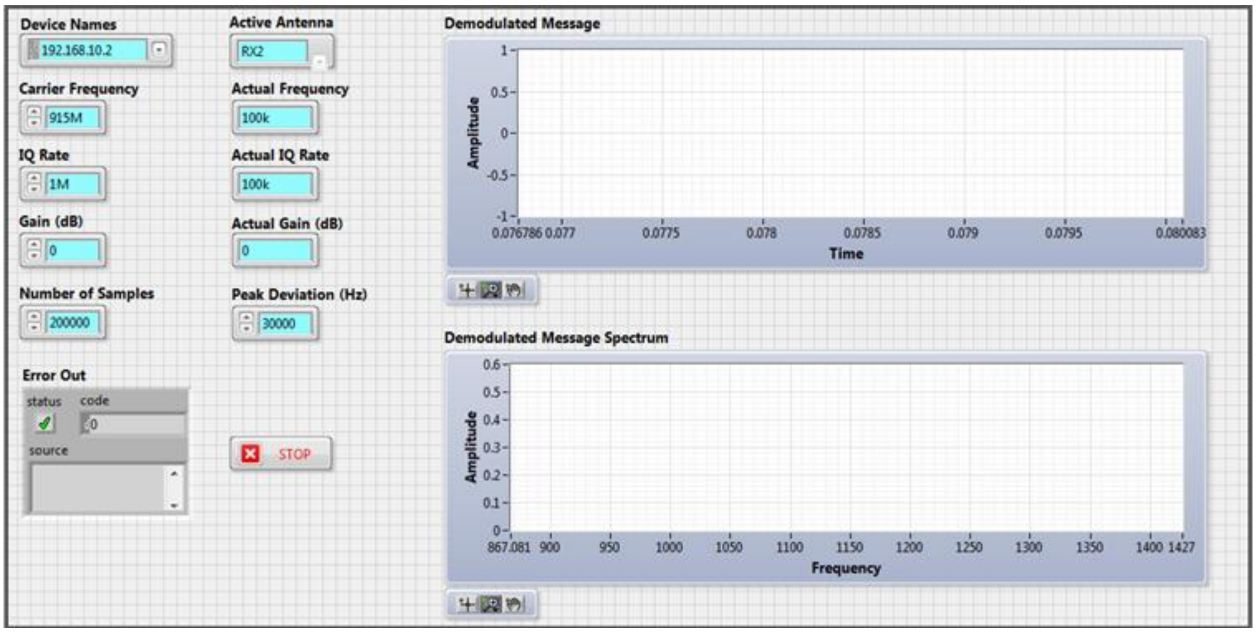
f) Remember that the output signal generated by the IIR filter is a sequence. You need to convert it back into an analog waveform before sending it to the buffer, "Write Tx Data" VI. Use "Build Waveform" VI (See Fig. 63) to convert the data values into a waveform by setting in the sampling interval of the frequency modulated signal as the product of the sampling interval of the original message signal and the term $2\pi\Delta f_{max}$ (f_{max}: peak frequency deviation).
g) Write the frequency modulated waveform onto the buffer, "niUSRP Write Tx Data" VI.
h) Plot the frequency modulated waveform using the waveform chart provided. Get the frequency modulated signal spectrum by connecting the waveform (original or scaled) to the FFT Power Spectrum and PSD VI (Fig. 8), and then connect the output to the waveform graph provided.

i) Save your transmitter as a file whose name includes the letters "FM_Tx" and your initials.

6.4 Lab Procedure

6.4.1 Receiver

Your task is to add blocks as needed to the given receiver VI template "FM_Rx_Template.vi" to demodulate the complex array $\tilde{g}(t)$ retrieved by the "Fetch Rx Data" VI and reproduce $K_D \Delta f_{max} m_n(t)$ to get the quick scaled message signal $m_n(t)$. This template contains the six interface VIs along with a waveform graph on which to display your demodulated output signal. Fig. 67 shows the front panel of FM_Rx_Template.vi.



Device Names: 192.168.10.2
Carrier Frequency: 915M
IQ Rate: 1M
Gain (dB): 0
Number of Samples: 200000
Active Antenna: RX2
Actual Frequency: 100k
Actual IQ Rate: 100k
Actual Gain (dB): 0
Peak Deviation (Hz): 30000
Error Out: status ✓ code 0 source

Fig. 67: Front panel of FM_Rx_Template.vi

Step-by-step LabVIEW setup to demodulate the FM signal using USRP:

1. After fetching $x_r(t)$ from the "niUSRP Fetch Rx Data VI", get data values and sampling time interval of this received waveform using the "Get Waveform Components" VI (Fig. 61).
2. Extract the angle of $x_r(t)$ using the "Complex to Polar" VI (Fig. 68).

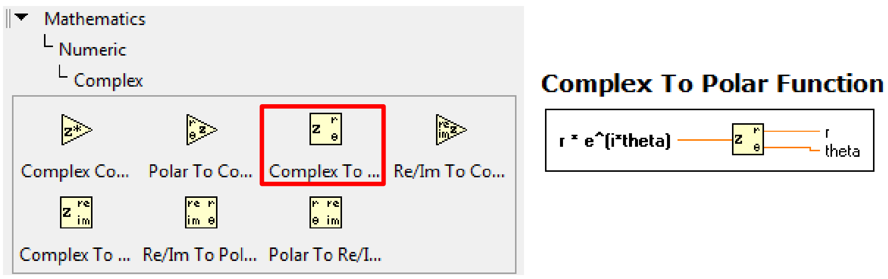


Fig. 68: Complex to Polar VI

3. Unwrap the angle with the "Unwrap Phase" VI (Fig. 69).

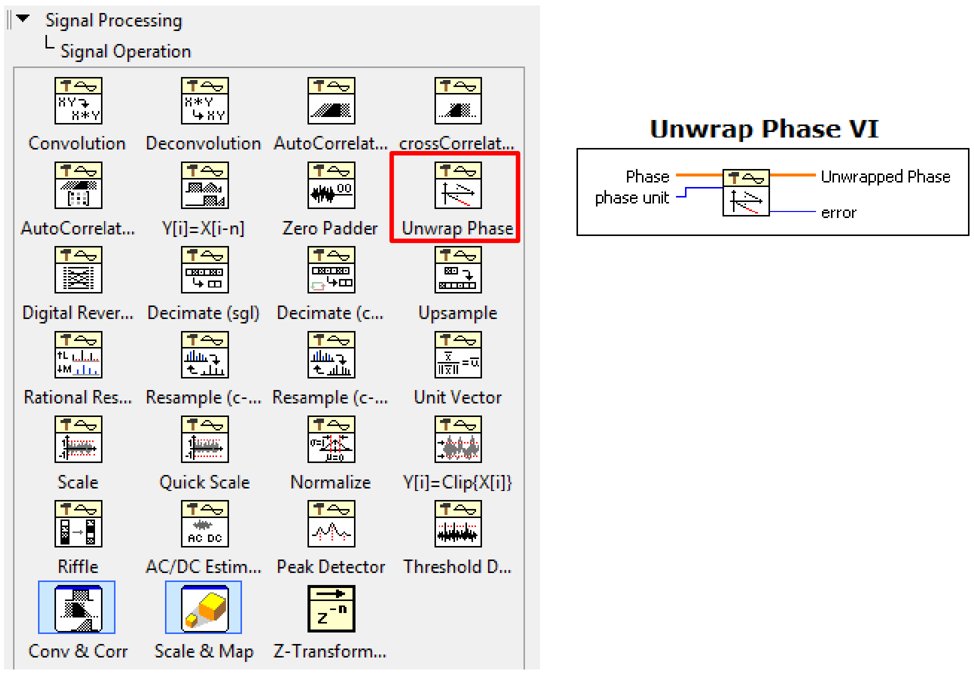


Fig. 69: Unwrap Phase VI

4. The unwrapped sequence of angles is then passed into a differentiator. We recognize that in discrete time

$$\frac{d\theta}{dt} = \frac{\theta[n] - \theta[n-1]}{T} \tag{43}$$

where T is the sampling time interval. The differentiator can be implemented using one of the "FIR Filter" VIs (Fig. 70). For the "FIR coefficients" array use $\left(\frac{1}{T}, -\frac{1}{T}\right)$. Generate this array by using the "Build Array" VI, reciprocal block and negate block.

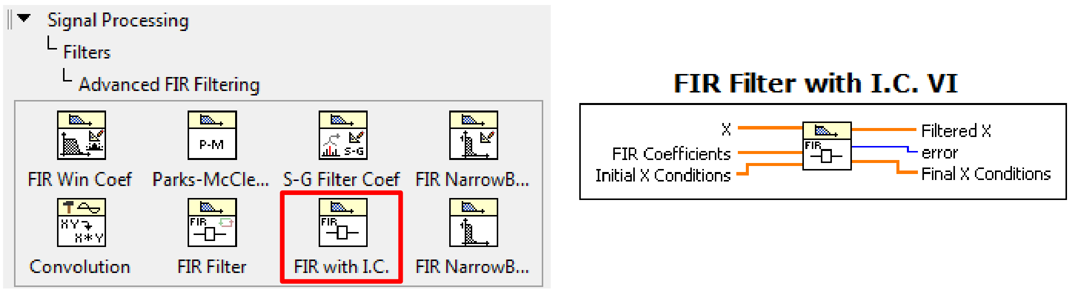


Fig. 70: FIR Filter with Initial Conditions VI

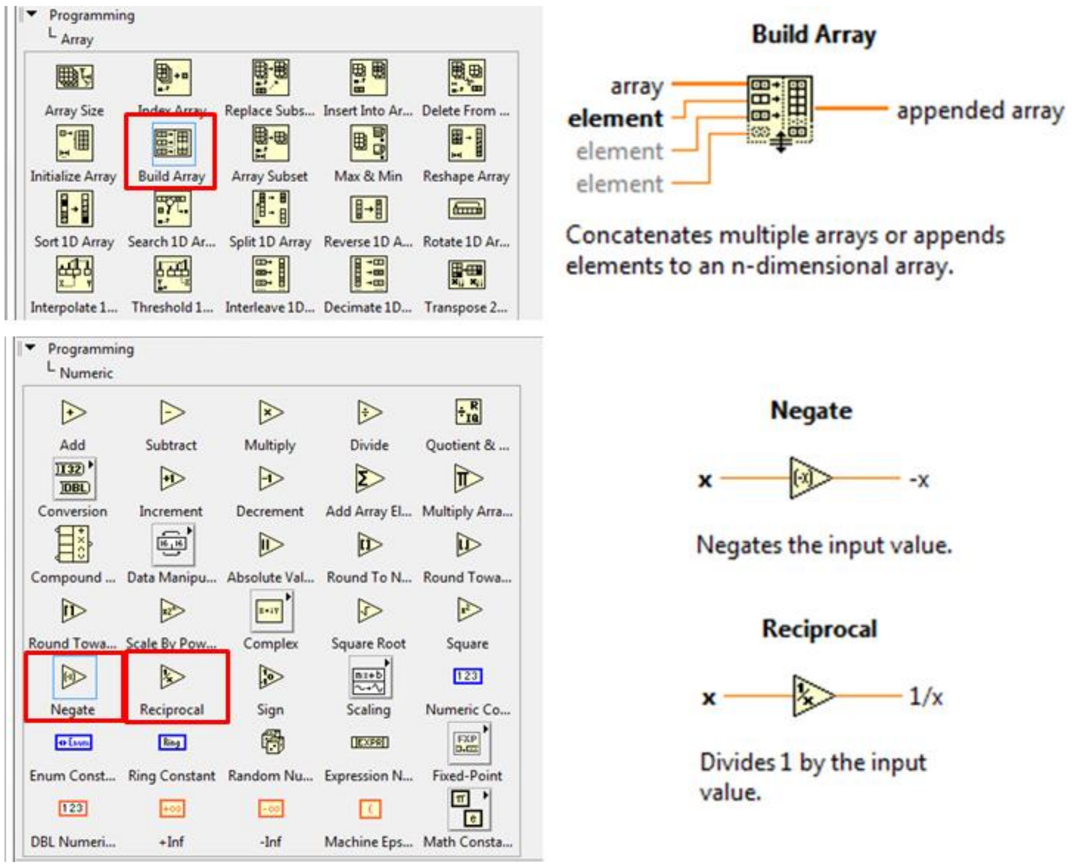


Fig. 71: Top: Build Array VI; Bottom: Negate and Reciprocal functions

Use shift registers to feedback the final conditions of current loop iteration to be the initial conditions of the next loop iteration for the FIR filter. Initialize shift registers to a zero constant. The output from the differentiator is given as $2\pi\Delta f_{max} m_n(\alpha)$.

5. The output from the differentiator is then passed into an envelope detector; a low-pass filter would be best since differentiation tends to enhance high-frequency noise. In this example, we use the Butterworth Filter VI (Fig. 13), but any low-pass filter will produce usable results. Set the filter type as low- pass, filter order as 5, low-cut off frequency as 5000 Hz, and the sampling frequency as reciprocal of sampling time.

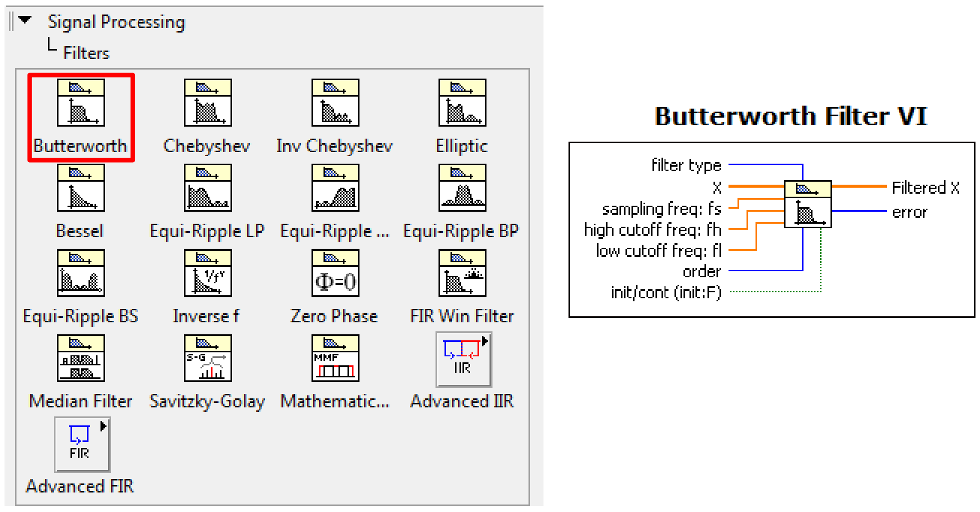


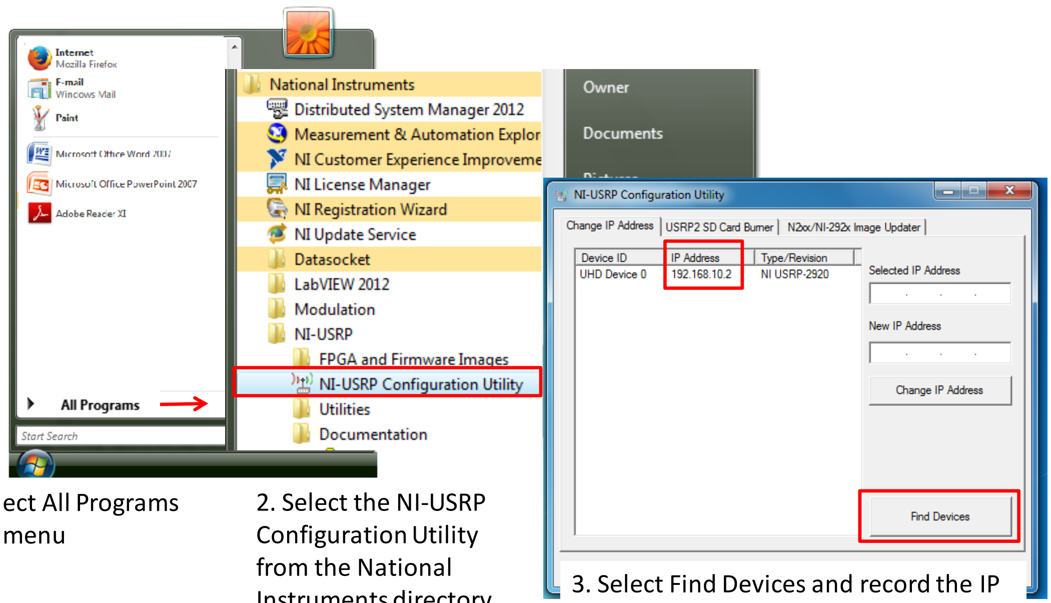
Fig. 72: Butterworth Filter VI

6. Connect the output of the LPF to the "Build Waveform" VI (Fig. 63) to get the analog waveform. Set the sampling time interval to be the same as that of the FM signal fetched by the buffer.
7. Downscale the amplitude of the waveform by a factor of $2\pi\Delta f_{max}$ to get the message signal $m_n(t)$.
8. Plot the downscaled waveform and compare it with the original message signal you modulated and transmitted. Plot the spectrum of the demodulated signal using the "FFT Power Spectrum and PSD" VI (Fig. 64), and then connecting the output thereof to a waveform graph.
9. Save your receiver as a file whose name includes the letters "FM_Rx" and your initials.

6.4.2 Testing the transmitter and receiver

Test your designed transmitter and receiver VIs for this lab.

1. Run LabVIEW and open the transmitter and receiver VIs that you created.
2. Connect the computer to the USRP using an Ethernet cable.
3. Open the NI-USRP Configuration Utility found in the National Instruments directory under programs files as shown in Fig. 2. Be sure to record the IP addresses since you will need them to configure your software.



Device ID	IP Address	Type/Revision
UHD Device 0	192.168.10.2	NI USRP-2920

1. Select All Programs from menu

2. Select the NI-USRP Configuration Utility from the National Instruments directory

3. Select Find Devices and record the IP address of the radio or radios since you will need them to configure the software in the lab.

Fig. 73: Finding the IP Address: Radio Connectivity Test

If the IP address does not appear in the window then check your connections and ask the Teaching Assistant (TA) to verify that the LAN card has been configured correctly.

4. Connect a loopback cable between the TX 1 and RX 2 antenna connectors. Remember to connect the attenuator to the receiver end.

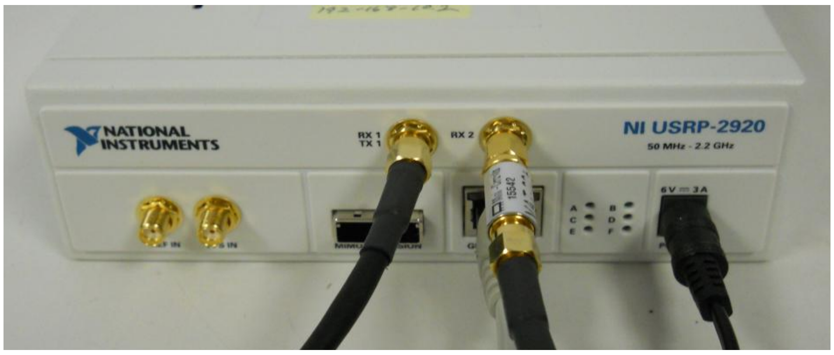


Fig. 74: Broadcast Setup

5. Ensure that the **transmitter VI** is set up according to Table II and Table XI.

Table X – Transmitter Settings

Field	Setting
Device Name:	192.168.10.x
Carrier Frequency:	915 MHz
IQ Rate:	1 MHz
Gain:	Use default
Active Antenna:	TX1
Message Length:	200,000 samples

Table XI – Transmitter Message Settings

Field	Setting
Peak Frequency Deviation	30 kHz
Start Frequency	1 kHz
Delta Frequency	1 kHz
Number of Tones	1

6. Ensure that **receiver VI** is set up according to Table VII.

Table XII – Receiver Settings

Field	Setting
Device Name:	192.168.10.x
Carrier Frequency:	915 MHz
IQ Rate:	1 MHz
Gain:	0 dB
Active Antenna:	RX2
Number of Samples:	200,000 samples
Peak Frequency Dev.	30 kHz

7. Run the receiver VI. LED “C” will illuminate on the USRP if the radio is receiving data.
8. Run the transmitter VI. LED “A” will illuminate on the USRP if the radio is transmitting data.
9. After a few seconds, stop the receiver using the STOP button on the receiver VI, and then stop the transmitter using the STOP button on the transmitter VI. Use the large STOP button on the front panel of the VI to stop transmission; otherwise the USRP may not be stopped cleanly.

10. Use the horizontal zoom feature on the graph palette to expand the “message” waveform in the transmitter VI and the “demodulated output” waveform in the receiver VI. Both waveforms should be identical. If not, review the steps to do in the transmitter and receiver setup. If they seem identical, compare and analyse the plotted spectra.
11. Once you get identical waveforms, change the number of tones, start frequency, delta frequency, and peak frequency deviation to different values and observe changes in the waveforms.

6.5 Lab Write-up

<u>Performance Checklist</u>
Frequency Modulation

<u>Short Answer Questions</u>

1. Let the number of tones in the message signal = 5, start frequency = 1 kHz, delta frequency = 1 kHz, and peak deviation frequency = 30 kHz. You are asked to analyze the bandwidth requirement for the carrier signal frequency modulated by this message.

   a) Figure the bandwidth requirement by checking the spectrum of the FM modulated signal in the transmitter VI by using "FFT Power Spectrum and PSD" VI and a waveform graph.

   b) Use Carson's rule to evaluate the same.

   Comment on the similarities and differences of your findings in both parts.

2. Why is the IIR filter used in the transmitter setup? Can it be used outside the while loop in this VI? Give reasoning.

3. Why is the low-pass filter required, after the signal is differentiated in the receiver setup?

<u>Performance Measures</u>

Task	Standards	Sat/Unsat
Running VIs	Successful modulation and demodulation of signal.	
Questions	Quality of reasoning.	

<u>Discussion</u>
Did all configurations perform as expected?
Did you have any difficulties completing the lab?
Did your TA provide enough guidance?
Do you have any recommendations to improve the lab?

6.6 References

[1] R.E. Ziemer and W.H. Tranter, Chapter 3: Basic Modulation Techniques, In “Principles of Communications: Systems, Modulation and Noise”, Fifth Edition.
[2] Lab 6: Frequency Modulation, Bruce A. Black, Rose-Hulman Institute of Technology, July 2013.
[3] M.F. Mesiya. “Contemporary Communication Systems.” New York: The McGraw-Hill Companies, Inc, 2013

7 Pulse-Position Modulation

7.1 Summary

The objective of this lab is to explore Pulse-Position Modulation (PPM), a form of Pulse Time Modulation (PTM) in which analog sample values determine the position of a narrow pulse relative to the clocking time [1],[2]. At the receiving end, the original waveforms may be reconstituted from the PPM pulse train.

Overall, this lab will answer the following questions:

- How does PPM work?
- How can signals be generated, modulated and demodulated in NI LabVIEW with respect to a particular modulation scheme?
- How is transmission and reception carried on NI USRPs?

7.2 Background

7.2.1 Pulse-Position Modulation

PPM signal is composed of a sequence of pulses displaced from a specific time reference in a manner proportional to the sample values of the signal at that time **(**Fig. 9). The value of the analog message signal at the clock pulses $(y_1, y_2, \cdots)$ is then converted to a time delay with greater time shifts assigned to larger sample values (y_i). One drawback to this approach is that the maximum magnitude of samples must be known. The maximum magnitude is used to scale the sampled values such that

$$\frac{y_i}{|y|_{max}} = \frac{\Delta t_i}{T} \Rightarrow \Delta t_i = \frac{y_i}{|y|_{max}} T \tag{44}$$

where T is the sampling period of the clock signal. This implies that if $|y|_{max} \gg y_i$, then the respective samples are given a value very close to zero. If this happens with a majority of the sampled values, the resulting reconstructed signal will be distorted. As the figure illustrates, the time delays occur after the clock signal at the receiver (lags behind the clock signal) for positive values, and before the clock signal at the receiver (or lead the clock signal) for negative values.

Thus a PPM signal is represented by the expression $x(t)$ as a series of signal pulses:

$$x(t) = \sum_{n=0}^{\infty} g(t)\, rect\big(t - (nT + \Delta t_n)\big) \tag{45}$$

where $g(t)$ represents the shape of the individual pulses, and the occurrence times Δt_n are related to the values of the message signal $m(t)$ at the sampling instants nT. All signal information together with synchronizing pulses is transmitted at the sampling times only. Usually, $g(t)$ is a sinusoid oscillating at the carrier frequency (f_c). The resulting transmitted and received pulses are shown in Fig. 76.

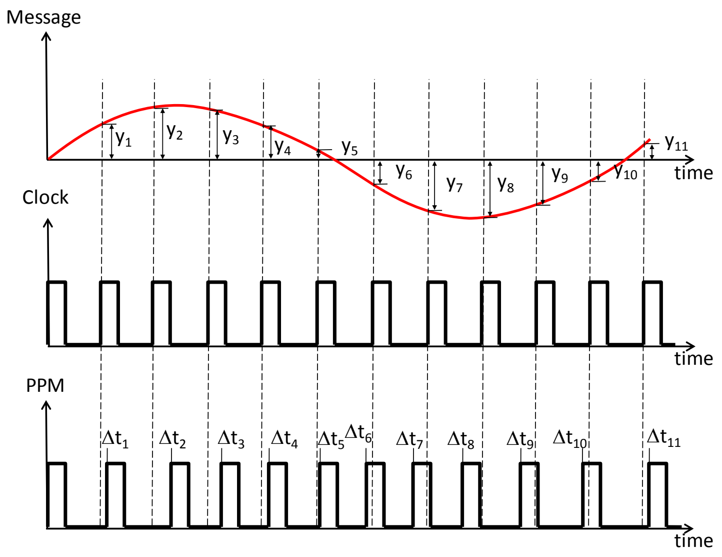


Fig. 75: Waveforms used to Create Transmitter PPM Signal

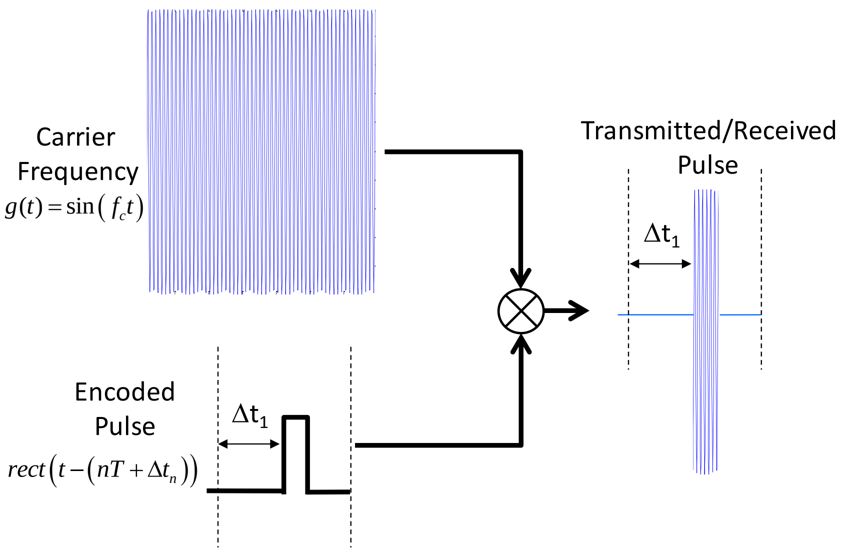


Fig. 76: Transmitted Pulse Signal

At the PPM receiver, the original signal can be reconstituted from information related to the samples. (Note: the signal needs to be sampled at a high enough frequency to avoid aliasing.) The resulting receiver output has negligible distortion, even though information related to the signal is not passed continuously as in AM/FM. Unlike these other methods, the pulse width and amplitude is kept constant in a PPM system, whereas the pulse positions, in relation to the clock pulse position, are varied by the modulating wave's respective instantaneous sampled value. To do this, it is important that the transmitter aligns the receiver clock by sending synchronizing pulses (red arrow in Fig. 77) so as to operate the timing circuits. The Δt_i now have the correct reference to accurately recover the values of y_i as shown in Fig. 77 and

$$\frac{y_i}{|y|_{max}} = \frac{\Delta t_i}{T} \Longrightarrow y_i = \begin{cases} -\frac{\Delta t_i}{T}|y|_{max}, & \Delta t_i \; leads \; RX \; Clock \\ \frac{\Delta t_i}{T}|y|_{max}, & \Delta t_i \; lags \; RX \; Clock \end{cases} \tag{46}$$

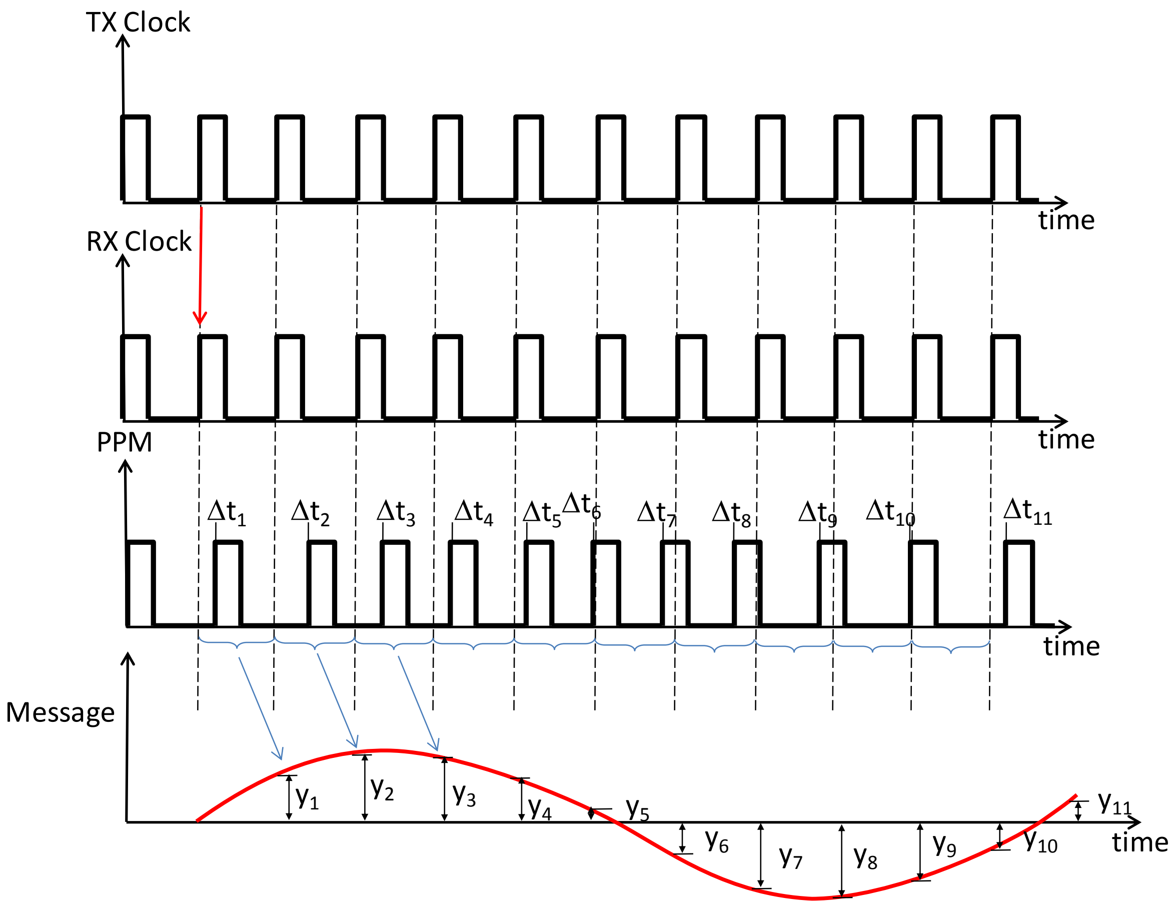

TX Clock
time
RX Clock
time
PPM
Δt1 Δt2 Δt3 Δt4 Δt5 Δt6 Δt7 Δt8 Δt9 Δt10 Δt11
time
Message
y1 y2 y3 y4 y5 y6 y7 y8 y9 y10 y11
time

Fig. 77: Waveforms Used in Reconstruction of Message

Another approach to using PPM is encoding a digital message composed of characters. Each character has M bits, thus there are 2^M possible characters. Each character is assigned a unique time-shift (Δt_n). As an example suppose that we are sending characters using the American Standard Character II (ASCII) codes. This means that there are 2^8 bits (256 characters). Each character in the code receives a value like the character "A" is assigned to 65. So the time delay would be calculated using (5):

$$\frac{65}{256} = \frac{\Delta t}{T} \Longrightarrow \Delta t_n = \frac{65}{256} T \qquad (47)$$

The message is then recovered by translating the character time delays back to characters, as shown in Fig. 78.

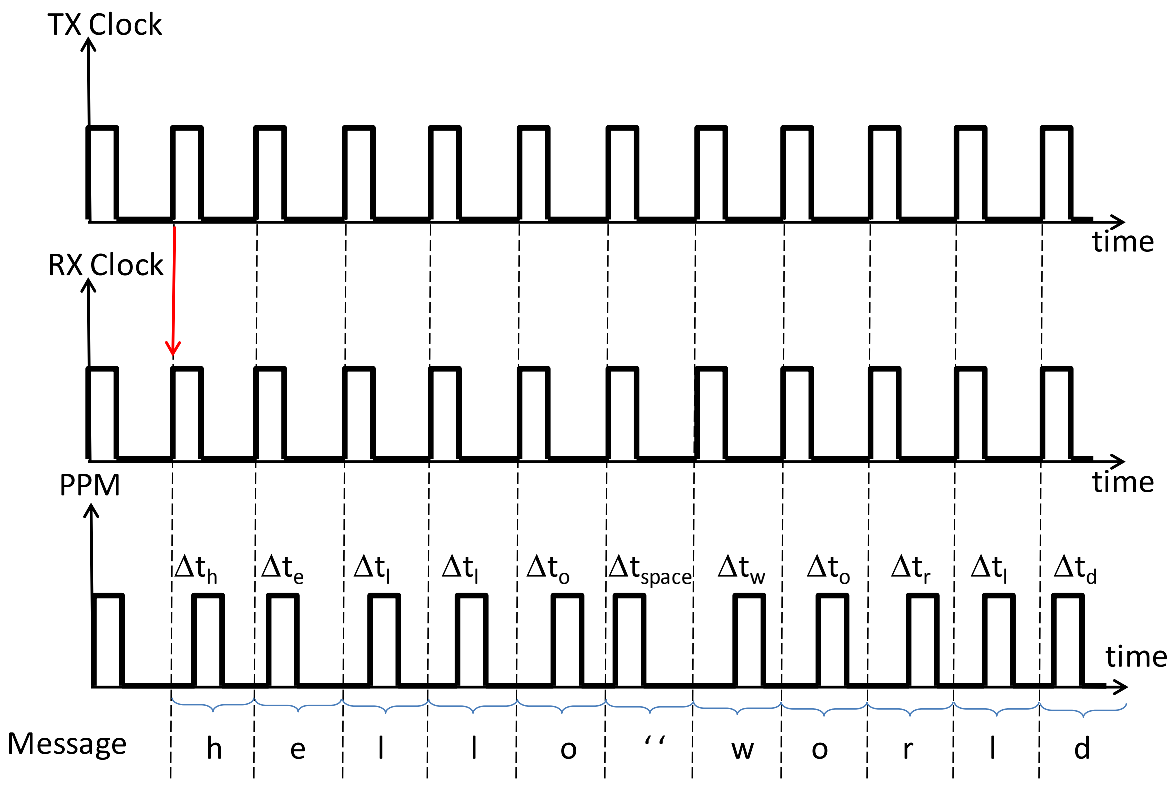


Fig. 78: Waveforms Used in Digital Message Reconstruction

This form of digital encoding is primarily useful for optical communications systems, where there tends to be little or no multipath interference. The digital encoding is also being used in the the Automatic Dependent Surveillance-Broadcast (ADS-B) messages transmitted by aircrafts equipped with ADS-B to augment data used by the air traffic management systems around the world.

Pulse-Position Modulation has the following advantages:

1. High noise immunity as amplitude is constant, which implies lesser noise interference;
2. Easy noise and signal separation;
3. Suitability for applications that require low-noise interference over long distances.
4. Constant transmission power for each pulse, due to constant amplitudes and pulse widths;
5. Simple demodulation that requires least circuitry.
6. And, high suitability for important applications such as ADS-B (Airplane tracking) [3].

7.3 Pre-Lab

Prior to this lab it is expected that students-

1. Understand PPM by reviewing the Summary section.
2. Have basic knowledge of LABVIEW and NI USRPs.

The students have to complete instructed steps on the Transmitter and Receiver VIs before coming to the lab.

7.3.1 VIs to be used in the lab:

Two VI templates have been provided for this lab: a transmitter VI (PPM_Lab_TX.vi) and a receiver VI (PPM_Lab_RX.vi).

7.3.2 Transmitter (PPM_Lab_TX.vi)

This is the transmitter VI that needs to be designed. In the pre-lab, you have to generate the message and the modulated signals to be transmitted. **Note: It is expected that you will save a copy of the VIs you design in the pre-lab before coming to the lab.**

The steps for generating the modulated signal to be transmitted are shown below.
Generate a waveform containing a "Square wave" and another containing a 'Sine wave' with the *'Simulate Signal Express VI'* by selecting the relevant items from the Waveform generation palette (Fig. 79) and configure both waveforms using the specifications in Table XIII.

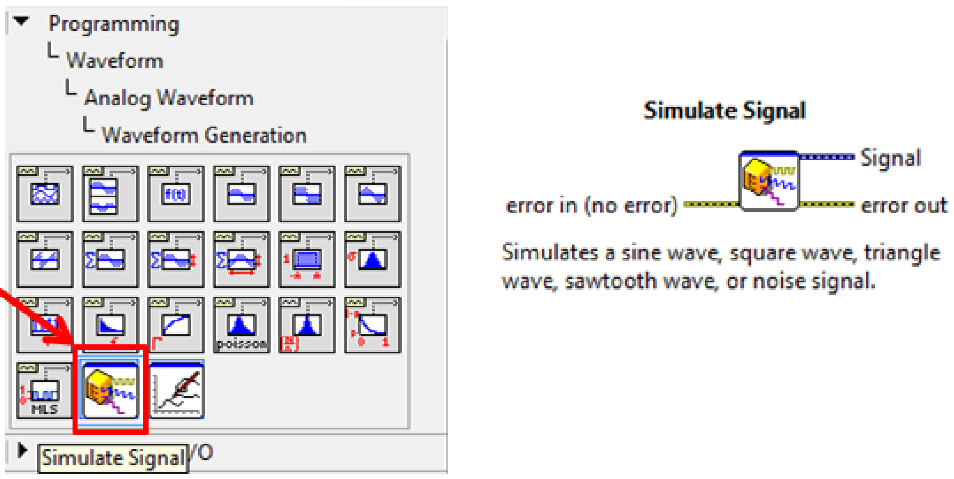
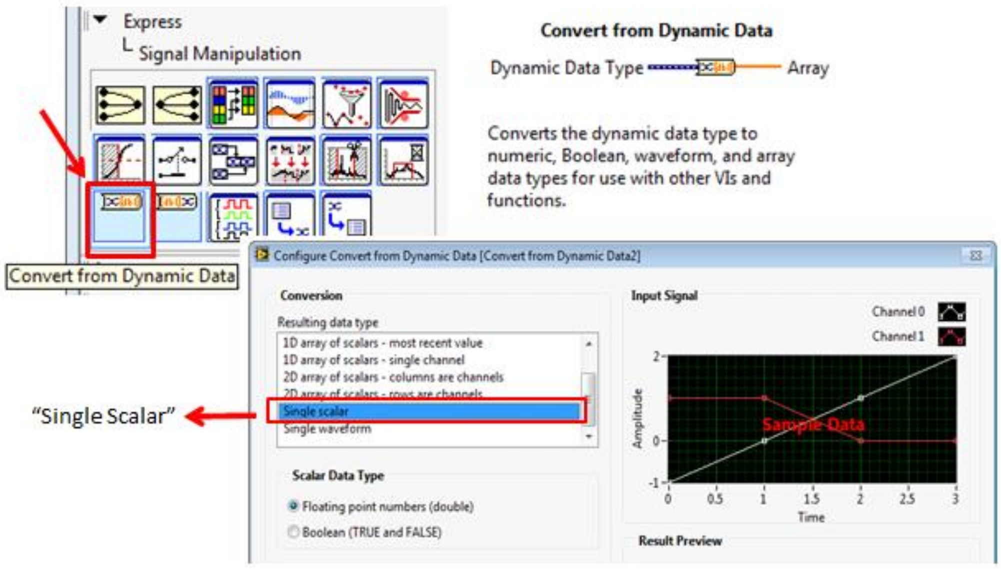


Fig. 79: Simulate Signal Express VI in the Waveform Generation Palette

Table XIII - 'Specifications for 'Square Wave' and 'Sine Wave'

Signal Type	Square	Sine
Frequency	5 Hz	5 Hz
Amplitude	1	20
Phase	0	0
Offset	0	30
Duty Cycle	10	--
Sampling Rate	1000 Hz	1000 Hz
Number of Samples	10	10
Timing	Simulate Acquisition Timing	Simulate Acquisition Timing
Time Stamps	Relative	Relative
Reset Signal	Use Continuous Generation	Use Continuous Generation
Signal Name	Square	Sine

a) Message Signal
Convert the simulated sine wave from dynamic data into a scalar by using the *'Convert from dynamic data'* function from the Signal manipulation pallete (Fig. 80), and configure it using the specifications in Table XIV.



Converts the dynamic data type to numeric, Boolean, waveform, and array data types for use with other VIs and functions.

Conversion
Resulting data type
1D array of scalars - most recent value
1D array of scalars - single channel
2D array of scalars - columns are channels
2D array of scalars - rows are channels
Single scalar
Single waveform

Scalar Data Type
Floating point numbers (double)
Boolean (TRUE and FALSE)

Fig. 80: 'Convert from Dynamic Data' function in the Signal Manipulation Palette

Table XIV - Specifications for 'Convert from Dynamic Data'

Conversion	**Single Scalar**
Data type	Floating point numbers (double)
Channel	0

The output of this function is the original message signal (sinusoidal). Add a waveform chart to the front panel as illustrated below.

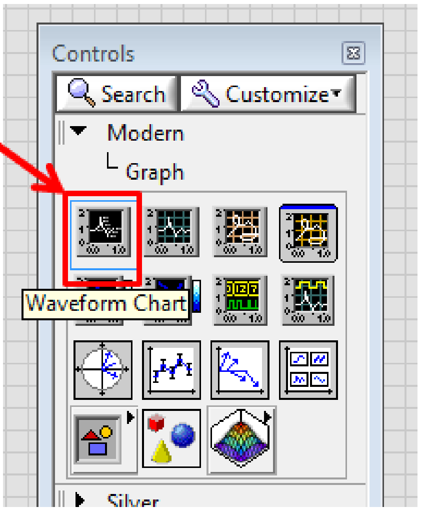


Fig. 81: Adding a 'Waveform Chart' to front panel

After placing the '*Waveform Chart'* function on the front panel, feed the scalar message signal to this waveform chart icon on the block diagram to view your **'Message Signal'** waveform. The message should be a sine wave as illustrated figure below.

Note: Scaling [5] may differ as per your settings. You may scale manually or auto-scale.

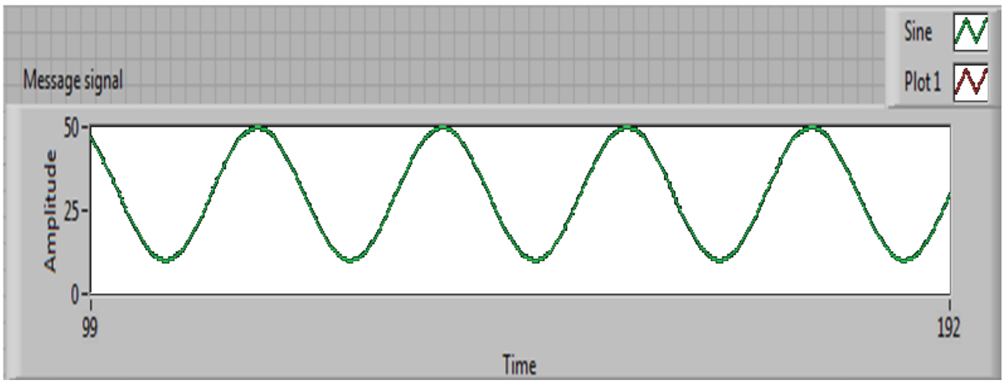


Fig. 82: Message Signal

b) Clock, PPM Waveforms
Generate two waveforms (A- Gives Clock Signal, and B- Gives PPM signal) containing a square wave with the *'Square Waveform VI'* from the Waveform Generation pallete (Fig. 83), and configure the waveforms using the specifications in Table XV.

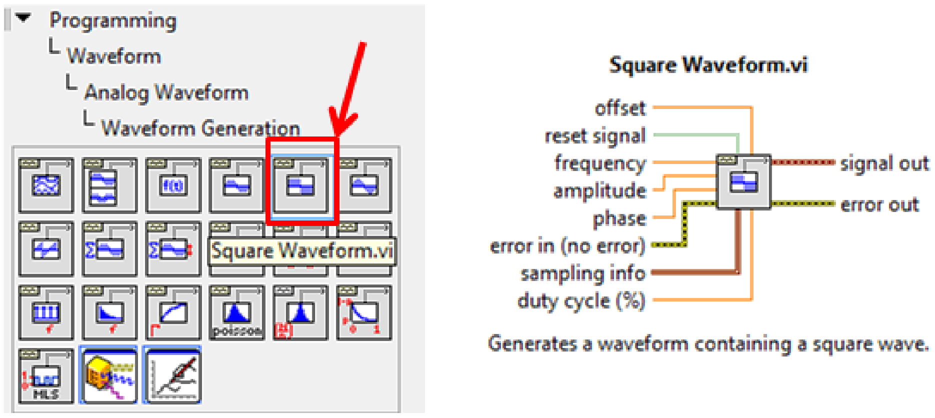


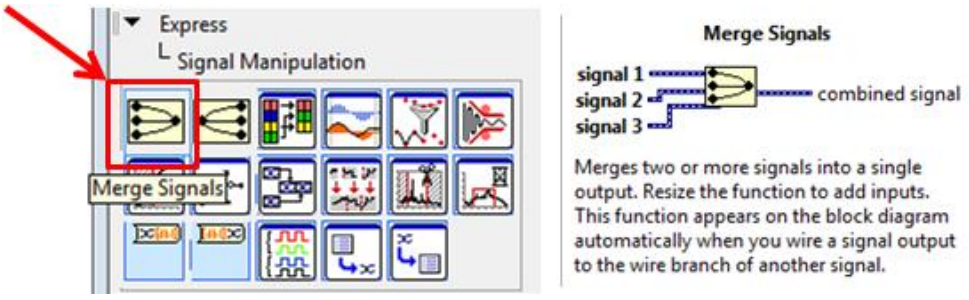
Fig. 83: Adding a 'Square Waveform' generator

Table XV - Specifications for the square waveforms A and B

Square waveform VI number	**A**	**B**
Offset	1	0
Reset	True	True
Frequency	100	100
Amplitude	1	1
Phase	Signal from square wave Simulate Signal Express VI	Scalar Message Signal + 90 degrees (to get PPM waveform)
Sampling rate	100000 Hz	100000 Hz
Number of samples	30000	30000
Duty Cycle	1%	10%

The output of the *'Square Waveform VI A'* is the ***'Clock signal'***, and the output of the *'Square Waveform VI B'* is the ***'PPM signal'.***

c) PPM with Clock
Scale the generated PPM signal to (-1, 1) by dividing it by 2 and then adding 0.5. Merge this scaled PPM signal with the clock signal by using the *'Merge Signals'* function to get the *'PPM with clock'* signal. The *"Merge Signals"* VI can be added to the block diagram as shown in Fig. 84.



Merges two or more signals into a single output. Resize the function to add inputs. This function appears on the block diagram automatically when you wire a signal output to the wire branch of another signal.

Fig. 84: Adding a 'Merge Signals' VI

Now, feed the PPM with clock signal to a waveform chart. The generated waveform should look like:

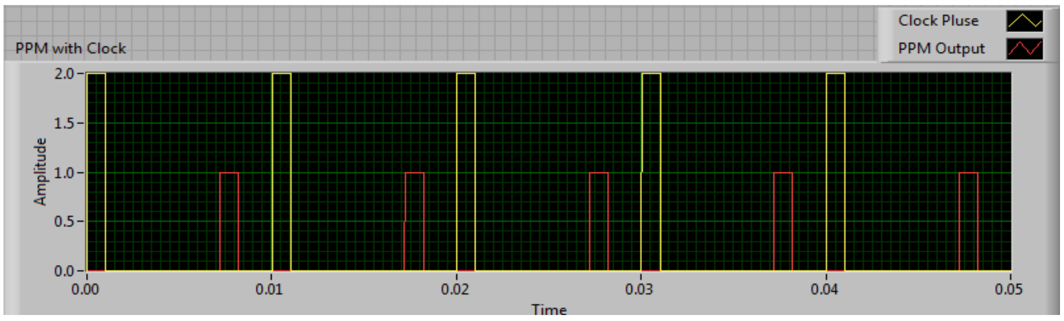


Fig. 85: PPM with Clock waveform

d) Transmitted Waveform
Next, generate a sine waveform with the *'Sine Waveform VI'* with the specifications in Table XVI.

Table XVI - Specifications for the sine waveform C

Sine waveform VI number	**C**
Frequency	5000
Amplitude	1

e) Modulate Signal
Scale (multiply) the scaled PPM signal (not the PPM with Clock signal) with the sine wave generated in the previous step to get the modulated waveform for transmission. The transmitted waveform should look like this:

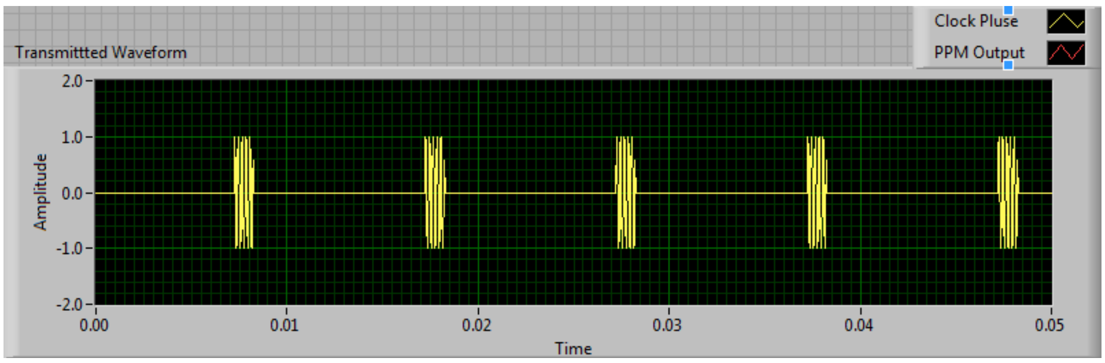


Fig. 86: Transmitted Waveform

f) Writing to the Transmission Buffer
For transmission, the '*transmitted waveform*' (double-precision floating point data) generated in the previous step should be written into a waveform data type by wiring it to the data terminal of the NI USRP Write TX VI (configured in the CDB WDT mode).

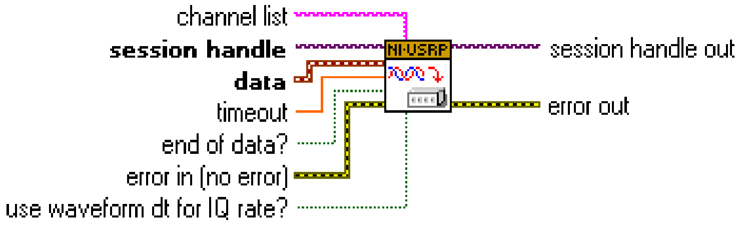


Fig. 87: NI USRP Write Tx Data VI (CDB WDT mode)

The waveform data type is now written into the specified channel. Your pre-lab design for the transmitter is now ready to be put up in the while loop in 'PPM_Lab_TX.vi' in order to repeat the operation until the stop button is enabled.

g) Summary of overall transmission
The 'NI USRP Open Rx Session VI' opens the Rx session to the device and the 'NI USRP Configure Signals VI' configures the properties of the channel. The modulated signal i.e. ***'Transmitted signal'*** you generate is fed into a buffer (NI USRP Write TX VI), the session handle terminal of which is wired to NI USRP Close Session VI to close the session handle to the device.

7.3.3 Receiver (PPM_Lab_RX.vi)

This is the receiver VI that needs to be designed. The 'NI USRP Open Rx Session VI' opens the Rx session to the device and the 'NI USRP Configure Signals VI' configures the properties of the channel to match with that of the transmitted channel. The NI USRP Initiate VI starts the Rx acquisition and the acquired data is wired to the ***'Session Handle'*** input terminal of the NI USRP Fetch Rx Data VI (in CBT WDT mode) which fetches data from the specified channel.

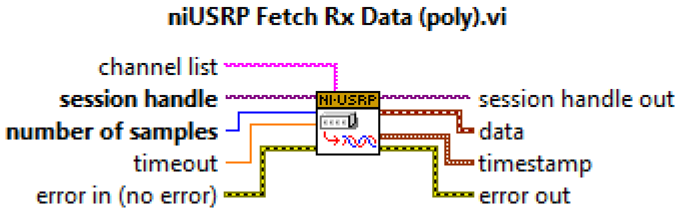
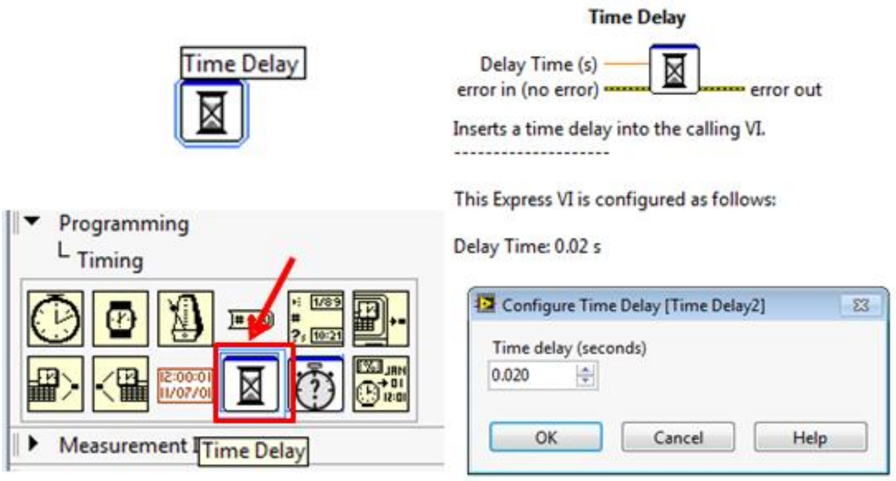


Fig. 88: NI USRP Fetch Rx Data VI

Note: Make sure that the number of samples you enter here is same as that in your transmitted signal.

a) Add a ***"Time Delay"*** VI to perform synchronization to align the local clock with the beginning of each symbol. Configure it to a time delay of 0.02 seconds.



Time Delay

Delay Time (s)
error in (no error)
error out

Inserts a time delay into the calling VI.

This Express VI is configured as follows:

Delay Time: 0.02 s

Configure Time Delay [Time Delay2]

Time delay (seconds)
0.020

OK Cancel Help

Fig. 89: Time delay VI

b) In the pre-lab for receiver design, you just have to input the fetched data to the demodulating function (***'Basic Level Trigger Detection VI')***, configure this function, and feed the output of this function to a waveform chart for comparison with the message signal.

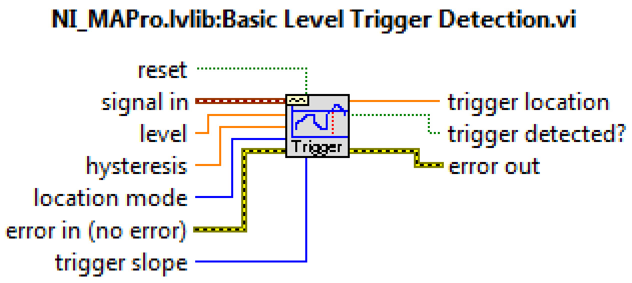


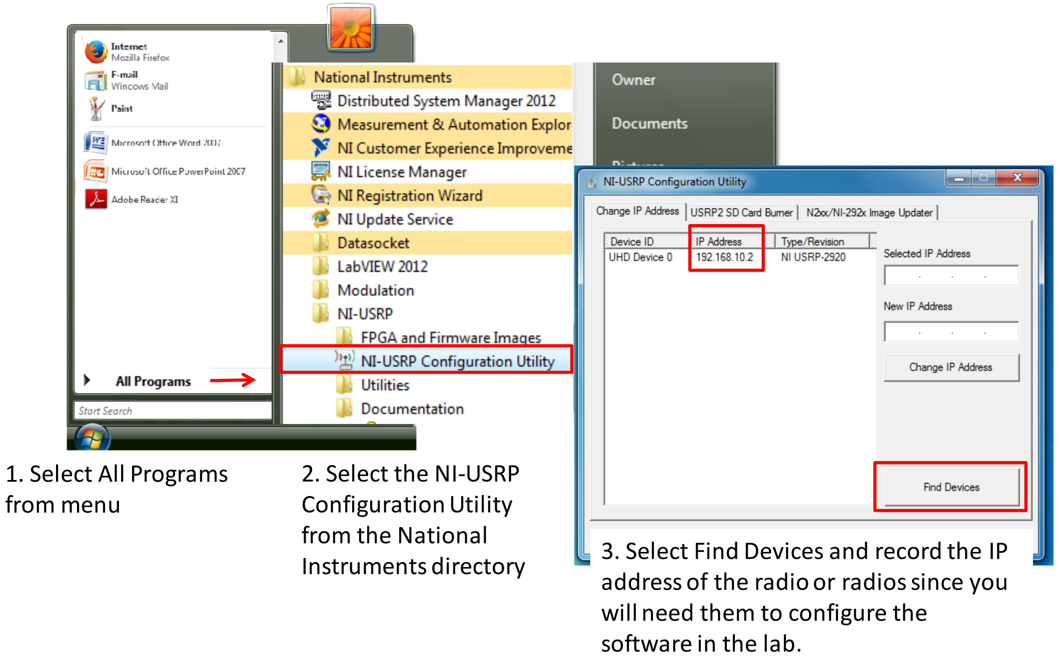
Fig. 90: Basic Level Trigger Detection VI

c) The ***'Basic Level Trigger Detection VI'*** finds the first level-crossing location in a waveform. For this lab you have to set up the trigger location as ***'Index'*** and trigger condition as ***'Rising Edge'***. The trigger location terminal gives the demodulated signal which should be wired to a waveform chart for a comparison with the original ***'Message Signal'***.

From the theory, we know that the decoder/receiver units for PPM demodulation are required to be extremely simple. The simplicity of the receiver design for this lab is established by the fact that you have to implement only the "Basic Level Trigger Detection" and "Time Delay" functions to demodulate the received signal.

7.4 Lab Procedure

1. Run LabVIEW and open the transmitter and receiver VIs that you created in the pre-lab.
2. Connect the computer to the USRP using an Ethernet cable.
3. Open the NI-USRP Configuration Utility found in the National Instruments directory under programs files as shown in Fig. 2. Be sure to record the IP addresses since you will need them to configure your software.



Device ID	IP Address	Type/Revision
UHD Device 0	192.168.10.2	NI USRP-2920

1. Select All Programs from menu

2. Select the NI-USRP Configuration Utility from the National Instruments directory

3. Select Find Devices and record the IP address of the radio or radios since you will need them to configure the software in the lab.

Fig. 91: Finding the IP Address: Radio Connectivity Test

If the IP address does not appear in the window then check your connections and ask the Teaching Assistant (TA) to verify that the LAN card has been configured correctly.

4. Connect a loopback cable between the TX 1 and RX 2 antenna connectors. Remember to connect the attenuator to the receiver end.

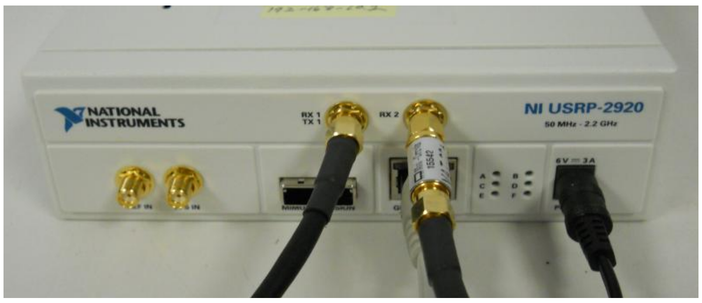


Fig. 92: USRP Setup

5. Configure the transmitter VI controls using the parameters in Fig. 93. In case you don't feed values supported by the device, the USRP would coerce your values to values supported by the device. The coerced values are shown up as in the ***'Actual IQ Rate'***, ***'Actual Carrier Frequency'*** and ***'Actual Gain'***.

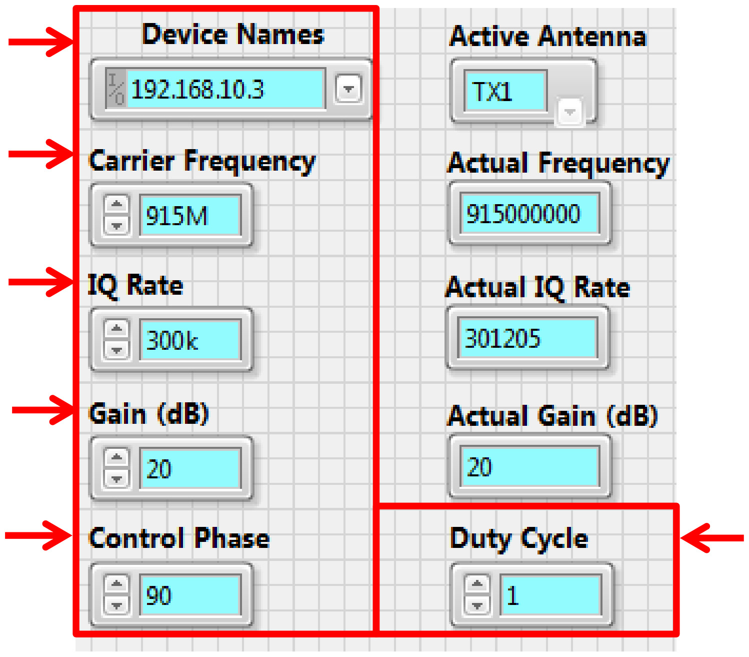

Device Names: 192.168.10.3
Carrier Frequency: 915M
IQ Rate: 300k
Gain (dB): 20
Control Phase: 90
Active Antenna: TX1
Actual Frequency: 915000000
Actual IQ Rate: 301205
Actual Gain (dB): 20
Duty Cycle: 1

Fig. 93: Transmitter VI Controls

6. Ensure that the **receiver VI** is set up to use the correct ***'Device name'***. Configure the receiver controls using parameters in Fig. 94. Note that the ***'Carrier Frequency'*** and ***'IQ Rate'*** values match those used by the transmitter, and the ***'Number of samples'*** corresponds with the value of the transmitted waveform.

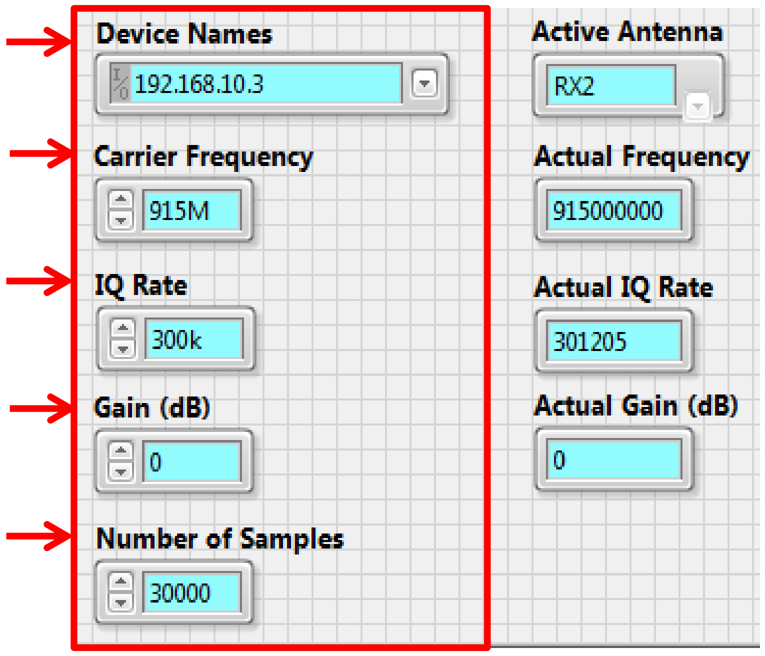

Device Names: 192.168.10.3
Carrier Frequency: 915M
IQ Rate: 300k
Gain (dB): 0
Number of Samples: 30000
Active Antenna: RX2
Actual Frequency: 915000000
Actual IQ Rate: 301205
Actual Gain (dB): 0

Fig. 94: Receiver VI Controls

Important set-up notes:

- ✓ Make sure the global set-up configuration has been performed before interfacing with the USRPs.
- ✓ Make sure the Tx and Rx VIs are always set to the same carrier frequency whenever you pair them up to communicate.
- ✓ Transmission should start after receiving workstations are ready to receive.
- ✓ Verify that device name fields in both Tx and Rx VIs are set to the IP address of the URSP in use.
- ✓ Make sure to connect the provided attenuator between the receiver USRP's Rx input and the antenna/loopback-cable. The attenuator is used to decrease the power level of the transmitted signal in order to avoid a high power signal at the receiver's end, due to the radios proximity.

7.4.1 Worksheet: The Effect of Removing Timer Sub-VI

1. Run the receiver VI. (LED "C" will illuminate on the USRP if the radio is receiving data.)
2. Run the transmitter VI. (LED "A" will illuminate on the USRP if the radio is transmitting data.)
3. After a few seconds, stop the receiver using the STOP button.
4. Stop the transmitter.

Note: Use the large STOP button on the front panel when you want to stop transmission or reception, otherwise the USRP may not be stopped cleanly.

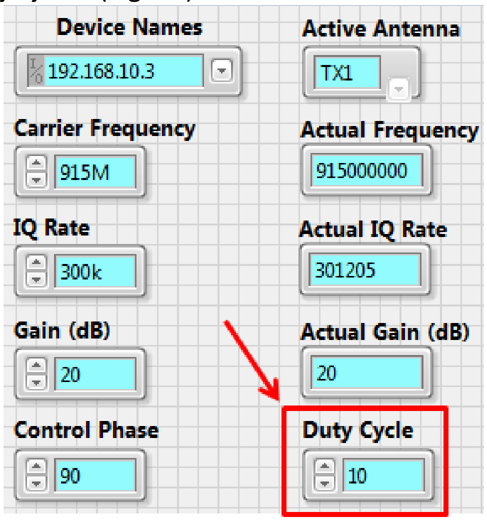
5. Observe waveform on the **receiver VI** and record your observations about the transmitted and the received signals. The received signal should be very similar to the original ***'Message Signal'***, except for the distortion. You may encounter a timing problem since the transmitter receiver synchronization hasn't been performed to keep the circuitry simple. So it is recommended you try again before making any conclusion.
6. If you see the message and demodulated sine waves similar, go to the block diagram for the receiver VI and delete the time delay VI.
7. Run the receiver VI and then the transmitter.
8. After a few seconds, stop the receiver and transmitter using the STOP buttons.
9. Observe waveform on the **receiver VI,** and record your observations about the message and the demodulated signals in the space below. Is the demodulated signal similar to the original ***'Message Signal'***?

10. Go to the block diagram for the receiver VI and restore the timer sub-VI (Fig. 89).

Note: Difference in your results (waveforms) and the provided illustrations could be due to different x/y scaling for a particular waveform charts. If you are unfamiliar with scaling charts and graphs, refer to the NI tutorial on customizing graphs and plots [5].

7.4.2 Worksheet: The Effect of The Clock's Duty Cycle

1. Change the clock's ***"Duty cycle"*** (Fig. 95) to 10%.



Device Names: 192.168.10.3
Active Antenna: TX1
Carrier Frequency: 915M
Actual Frequency: 915000000
IQ Rate: 300k
Actual IQ Rate: 301205
Gain (dB): 20
Actual Gain (dB): 20
Control Phase: 90
Duty Cycle: 10

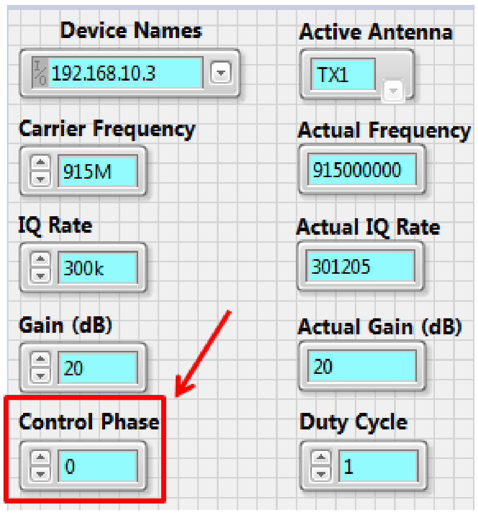
Fig. 95: Transmitter Duty Cycle

2. Run the receiver VI. (LED "C" will illuminate on the USRP if the radio is receiving data.)
3. Run the transmitter VI. (LED "A" will illuminate on the USRP if the radio is transmitting data.)
4. After a few seconds, stop the receiver and transmitter using the STOP buttons.
   Note: Use the large STOP button on the front panel when you want to stop transmission or reception, otherwise the USRP may not be stopped cleanly.
5. Observe waveform on the **receiver VI** and record your observations about the associated waveforms in the space below.

6. When you finish your analysis, change the clock's ***"Duty cycle"*** (Fig. 95) back to 1%.

7.4.3 Worksheet: The Effect of Phase on PPM Transmission

1. Change the *"Control Phase"* to the first entry in the table of observations (Fig. 96).



Device Names	Active Antenna
192.168.10.3	TX1

Carrier Frequency: 915M	Actual Frequency: 915000000
IQ Rate: 300k	Actual IQ Rate: 301205
Gain (dB): 20	Actual Gain (dB): 20
Control Phase: 0	Duty Cycle: 1

Fig. 96: Transmitter Control Phase

2. Run the transmitter VI. (LED "A" will illuminate on the USRP if the radio is transmitting data.)
3. Run the receiver VI. (LED "C" will illuminate on the USRP if the radio is receiving data.)
4. After a few seconds, stop the receiver and transmitter using the STOP buttons.

   Note: Use the large STOP button on the front panel when you want to stop transmission or reception, otherwise the USRP may not be stopped cleanly.

5. Observe waveform on the **receiver VI** and record your observations about the transmitted and received signals in Table XVII.
6. Change the phase to 45 degrees and note the observation.
7. Lastly, change the phase back to 90 degrees.

Table XVII - Phase Observation Table

Phase Value (Degrees)	Observations
0	
45	

7.5 Lab Write-up

Performance Checklist

Pulse-Position Modulation

Questions (Please attach your answers).

1. What happens when you remove the time delay from the receiver?

2. What is the effect of increasing the duty cycle of the clock signal ('Square Waveform A')?

3. Why do we add 90 degrees to the phase of the Sine Wave simulated by '*Simulate Signal Express VI*'? [Hint: Try changing this value to 0, 45, 90 degrees. Analyze the PPM with Clock Signal on the transmitter VI and the demodulated signal on the receiver VI.]

Performance Measures

Task	Standards	Satisfactory/ Unsatisfactory
Hardware Setup	Working setup (transmission/reception).	
Running VIs	Successful modulation and demodulation.	
Data Collection	Quality of argument to Short Answer Questions.	

Discussion

Did all VIs perform as expected?

Did you have any difficulties completing the lab?

Did your TA provide enough guidance?

Do you have any recommendations to improve the lab?

7.6 References

[1] R.E. Ziemer and W.H. Tranter, Chapter 3: Basic Modulation Techniques, In “Principles of Communications: Systems, Modulation and Noise”, Fifth Edition.
[2] Leon. W. Couch II, Chapter 3: Baseband Pulse and Digital Signaling, In “Digital and Analog Communication Systems”, Fourth Edition.
[3] National Instruments, Community: ADS-B Decoder (Airplane Tracker), retrieved August 23, 2014, from https://decibel.ni.com/content/docs/DOC-22955.
[4] National Instruments, Tutorial: RF Simulation Demo: Pulse-Position Modulation, retrieved August 23, 2014, from http://www.ni.com/example/30165/en/.
[5] National Instruments, Tutorial: Customizing Graphs and Charts, retrieved August 23, 2014, from http://zone.ni.com/reference/en-XX/help/371361J-01/lvconcepts/customizing_graphs_and_charts/.

8 Random Process, Cross-correlation and Power Spectral Density

8.1 Summary

The objective of this laboratory would be to utilize LabVIEW and Universal Software Radio Peripheral (USRP) as a Radio Detection And Range (RADAR). This laboratory would illustrate how Cross-Correlation and Power Spectral Density (PSD) can be used to predict the distance of an object (or target) by comparing the transmitted signal and received signal.

8.2 Background

8.2.1 RADAR

The general principle is best understood by looking at the general relationship for a reflected signal.

$$r[n] = \alpha\, p[n-\tau] + w[n] \tag{48}$$

where, r is the reflected pulse; p is the transmitted pulse; α represents an attenuation factor (or signal loss of power); $w[n]$ is an additive white noise (Gaussian distributed with zero mean a variance equal to the power of the noise).

The lab uses a frequency-modulated continuous-wave (FMCW) RADAR pulse. In a FMCW RADAR a linear sweep in frequency is used. The resulting pulse is the signal (49).

$$p(t) = a_1 \cos[\phi(t)] \tag{49}$$

where

$$\phi(t) = 2\pi f_c t + \pi \frac{B}{T_m} t^2 \tag{50}$$

The phase ($\phi(t)$) is a function of the RADAR carrier frequency (f_c), the sweep width B, and the pulse repetition period (T_m). (See Fig. 97).

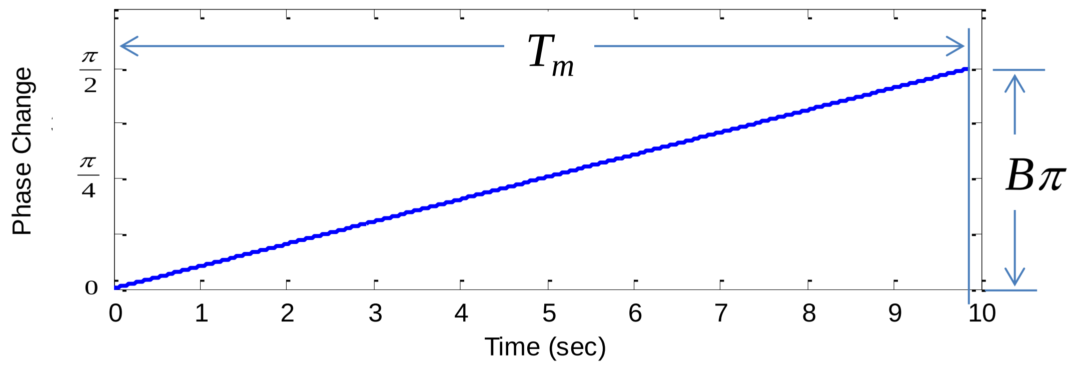


Fig. 97: Phase Component

The frequency modulation's instantaneous frequency ($f_i(t)$) is

$$f_i(t) = \frac{1}{2\pi}\frac{d}{dt}\phi(t) = f_c + \frac{B}{T_m} t \tag{51}$$

During the frequency sweep, the instantaneous frequency varies between the carrier frequency to the carrier frequency plus bandwidth. In FMCW RADAR, the signal generated by the RADAR is used for 2 purposes. The first purpose is determining the distance to the target by comparing the pulse with the echo signal. The other is to radiate out of the antenna. The radiated signal travels to the target bounces back to the antenna. The return is an echo. This echo is a phased shifted version of the transmitted signal (49) as shown in Fig. 98. The distance (r) to the target is given by

$$r(t) = a_2 \cos[\phi(t - \tau)] \tag{52}$$

where a_2 is the attenuated amplitude resulting from losses, reflectivity, and RADAR performance parameters. The propagation delay of the echo (τ) is

$$\tau = 2\,\mathrm{r/c} \tag{53}$$

where, c is the speed of light ($2.98\ x\ 10^8 m/sec$).

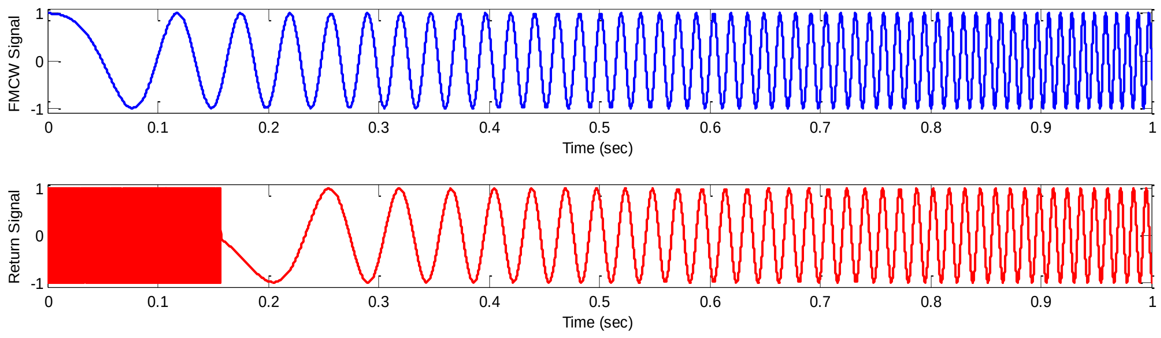


Fig. 98: Simulation of a Transmitted and Received Pulse (Echo) With a 0.1 Second Delay

8.2.2 Cross-Correlation

Cross-correlation [1] is a measure of similarity of two waveforms (pulse and return signal) as a function of time-lags. Given two real-valued sequences $p[n]$ and $r[n]$ of finite energy, the cross-correlation of $p[n]$ and $r[n]$ is a sequence $r_{pr}(l)$ defined as

$$r_{pr}(l) = \sum_{n=-\infty}^{\infty} p^*[n]r[n+l] \tag{54}$$

Notice the change in index variable from n to l in (54). This change indicates a change in focus from the individual observations (n) to the time between an observation and all other observations with a time difference or lag (l). Although the cross-correlation calculation is similar to convolution, they are not the same. The relationship is between the cross-correlation and convolution is given by (55).

$$r_{xy}(l) = x[l] * y[-l] \tag{55}$$

To understand how this applies to a RADAR system, we will examine the pulse and its return in Fig. 98. The cross-correlation of the transmitted and received signals shows they are correlated with a 0.1 second delay (Fig. 99). The resulting cross-correlation with the transmitted signal clearly shows a correlation with a 0.1 second delay (Fig. 100).

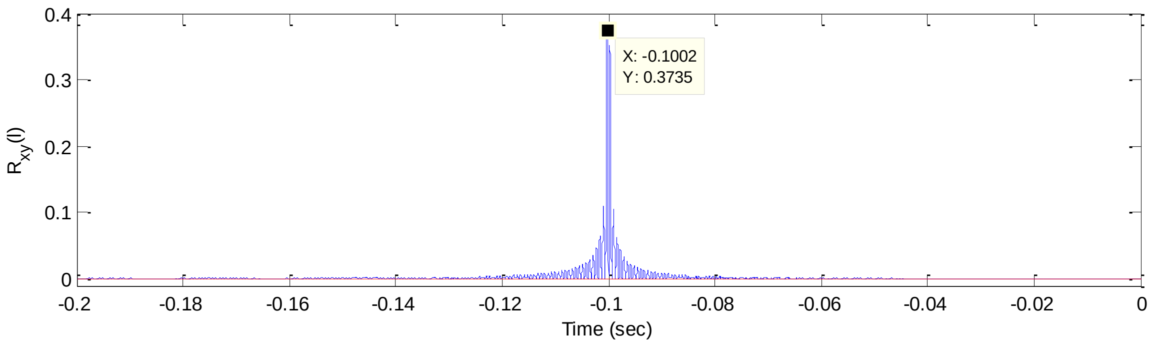


Fig. 99: Cross-correlation between Signals with a 0.1 sec Delay and Noise

Now suppose the target is further away, and the returned signal has an 0.8 sec delay.

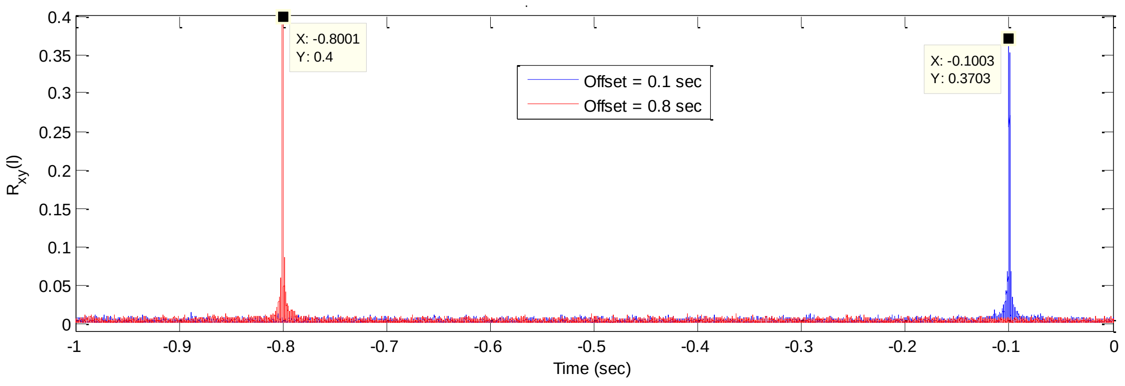


Fig. 100: Cross-Correlation with Noisy Echo with a 0.8 sec Delay with Noise

8.2.3 Power Spectral Density

The Power Spectral Density [3] can also be used to estimate the distance. In this approach the return signal and the pulse signal are multiplied together. The product contains the sum and difference frequencies. The sum of frequencies is approximately $2f_c$. This frequency is beyond the frequencies the electronics can respond to. Only the terms related to the difference frequencies are retained (56).

$$\begin{aligned} m(t) &= a_3 \cos[\phi(t) - \phi(t-\tau)] \\ &= a_3 \cos\left(2\pi f_{beat} t + 2\pi f_c \tau - \frac{\pi B}{T_m}\tau^2\right) \end{aligned} \qquad (56)$$

where the return's beat frequency (57) is a function of the pulse modulation frequency ($f_m = T_m^{-1}$) and the range resolution (dr) (58).

$$f_{beat} = \frac{B}{T_m}\tau = f_m \frac{r}{dr} \qquad (57)$$

$$dr = c/(2B) \qquad (58)$$

The f_{beat} is the largest peak in the spectrum of the product of the pulse and return signals (Fig. 101).

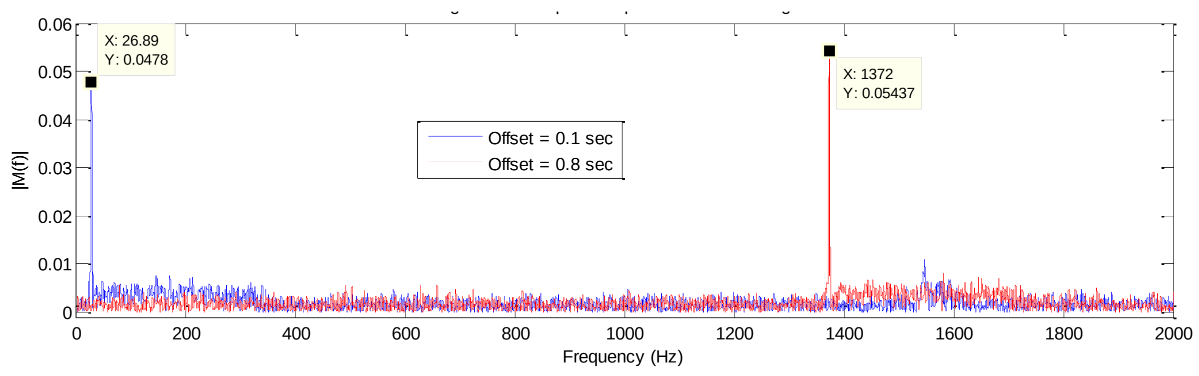


Fig. 101: Power Spectral Density (M(f)) Showing Beat Frequencies

8.3 Pre-Lab

The USRP RADAR that we will be implementing in this lab is designed to use the USRP's transceiver (TX1/RX1) as the emitter and the second receiver (RX2)will act as the receiver of the reflections. The two functions are part of the same VI. This combination will allow for the use of the transmitted signal as a reference signal as well as the pulse generator. The focus of the lab is on using LabVIEW's signal processing tools for cross-correlation and Fast Fourier Transforms (FFT) analysis.

Fig. 102 shows the general data flow for the data. The RADAR VI sends the pulse signal to the transmitter radio through the data switch. The radio emits the signal through the wave guide antenna (see Appendix C). The signal will bounce off a target such as the wall and picked up by the receiver wave guide (see Appendix C). The receiving radio will then collect the return signal and send it to the VI through the data switch.

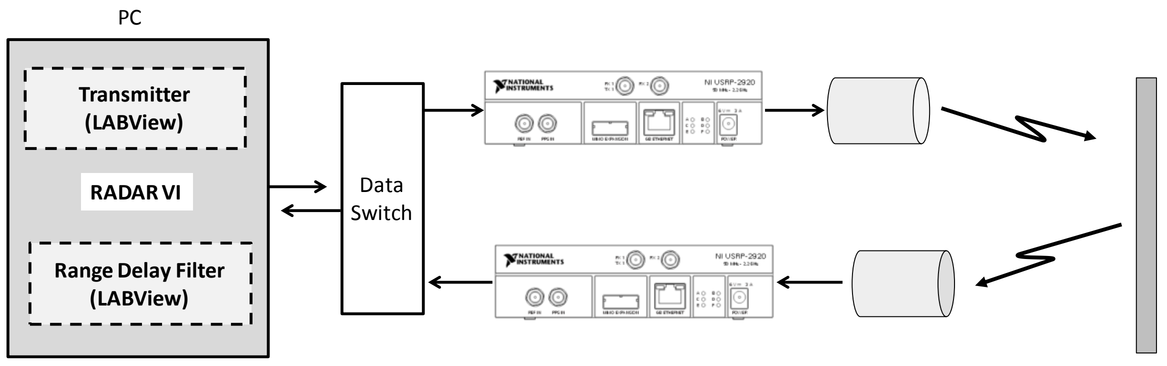


Fig. 102: Top Level Block Diagram of RADAR System for Lab

8.3.1 RADAR Transmitter

The RADAR Transmitter portion is divided into two parts. The first part deals with the initialization of the transmitter. The second part involves the generation and modulation of the RADAR pulse. The initialization of the transmitter uses the Open Tx Session VI and Configure Signal VI as usual. The niUSRP Set Time VI (Fig. 103, 1a) resets the timer on the transmitting radio to 0 (Fig. 103, 1b). The niUSRP Configure Trigger V (Fig. 103, 2a) configures the trigger generated by the onboard device timer. The trigger specifies the time (Fig. 103, 2b) to acquire or generate the first sample. These additional blocks are necessary to ensure that the timing elements of the signals are consistent.

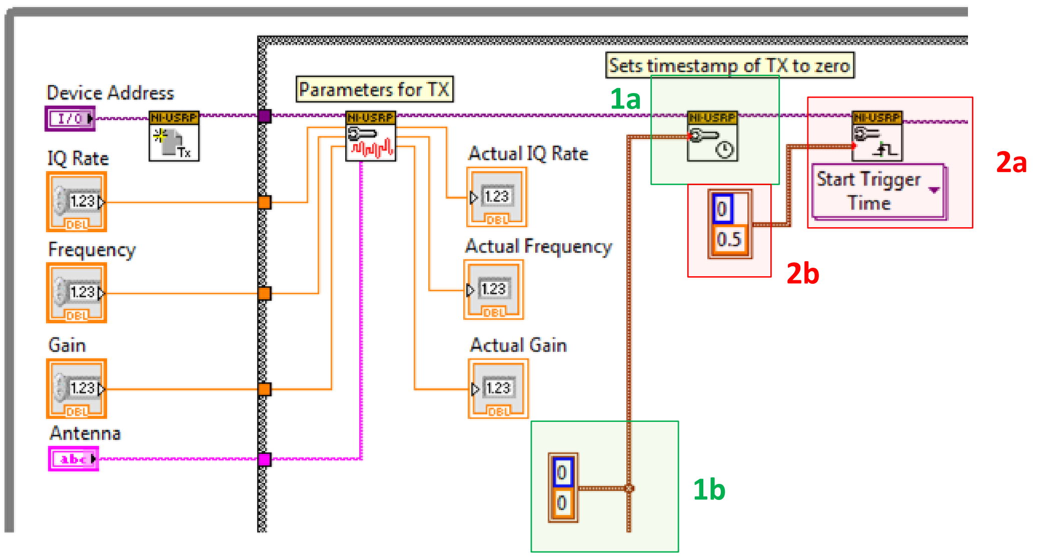


Fig. 103: Transmitter Setup Block Diagram

The second half of the transmitter block diagram (Fig. 104) generates the RADAR pulse (equation (1)). The block diagram uses the modulation toolkits FM Modulation VI and the Write Tx Data VI to generate and write a baseband signal to the transmitter USRP for transmission. The phase component of the instantaneous frequency, $f_i(t)$, for the triangular (Fig. 105) waveforms. The pulse height ($B\pi$) should be set to $\pi/2$ and a T_m of 1 second.

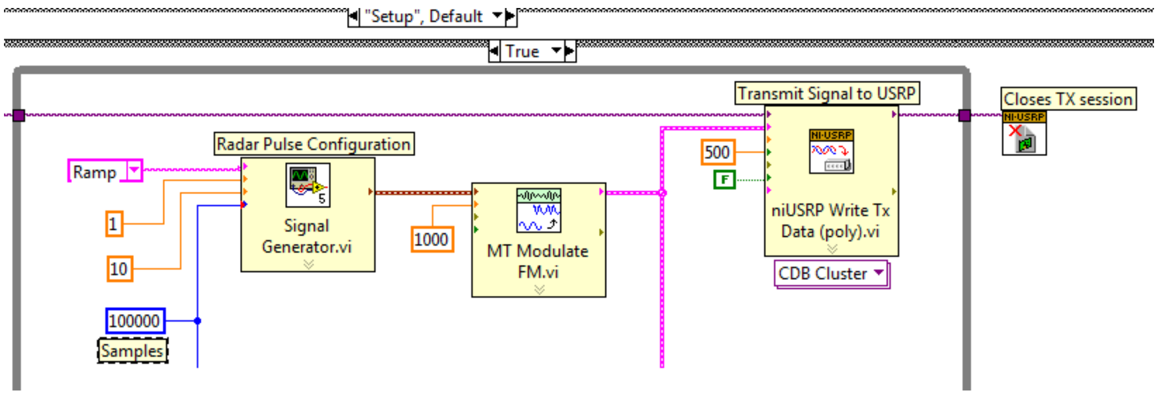


Fig. 104: Transmitter Pulse Generation and Transmitter Block Diagram

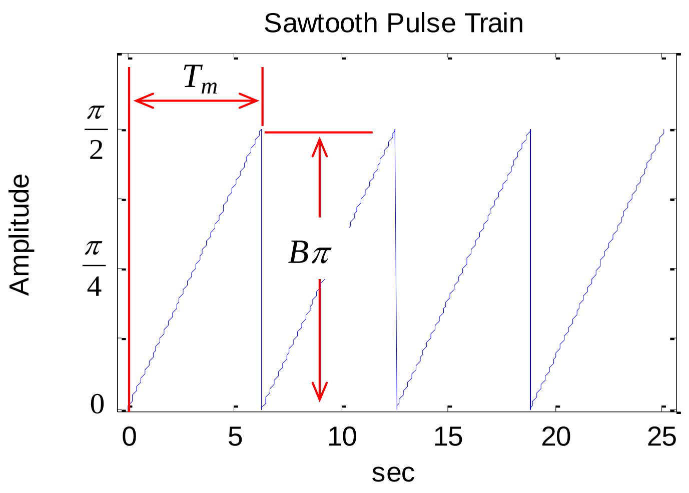


Fig. 105: Radar Pulse Train Block Diagram

8.3.2 RADAR Return Processing

Like the RADAR Transmitter, the Receiver portion is divided into two parts. The first part deals with the initialization of the receiver. The second part calculates the cross correlation between the transmitted and returned pulses ($r_{pr}(l)$) and the beat frequency (f_{beat}) PSD. The initialization of the transmitter uses the Open Rx Session VI, Configure Signal VI, and Rx Initiate VI as usual. The niUSRP Set Time VI (Fig. 106) resets the timer on the transmitting radio to 0. This additional block is necessary to ensure that the timing elements of the signals are consistent.

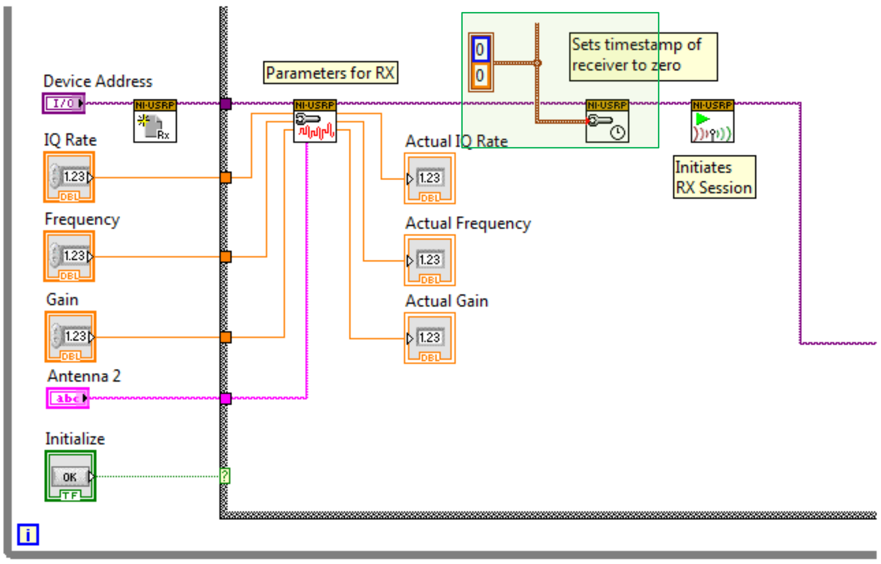


Fig. 106: Receiver Setup Block Diagram

The second half of the receiver block diagram (Fig. 107) processes the return signals. The block diagram uses the Fetch Rx Data VI and the MT FM Demodulation VI to convert the received signal into a baseband signal. In this lab you will be asked to compose both the PSD showing the beat frequency and the cross correlation.

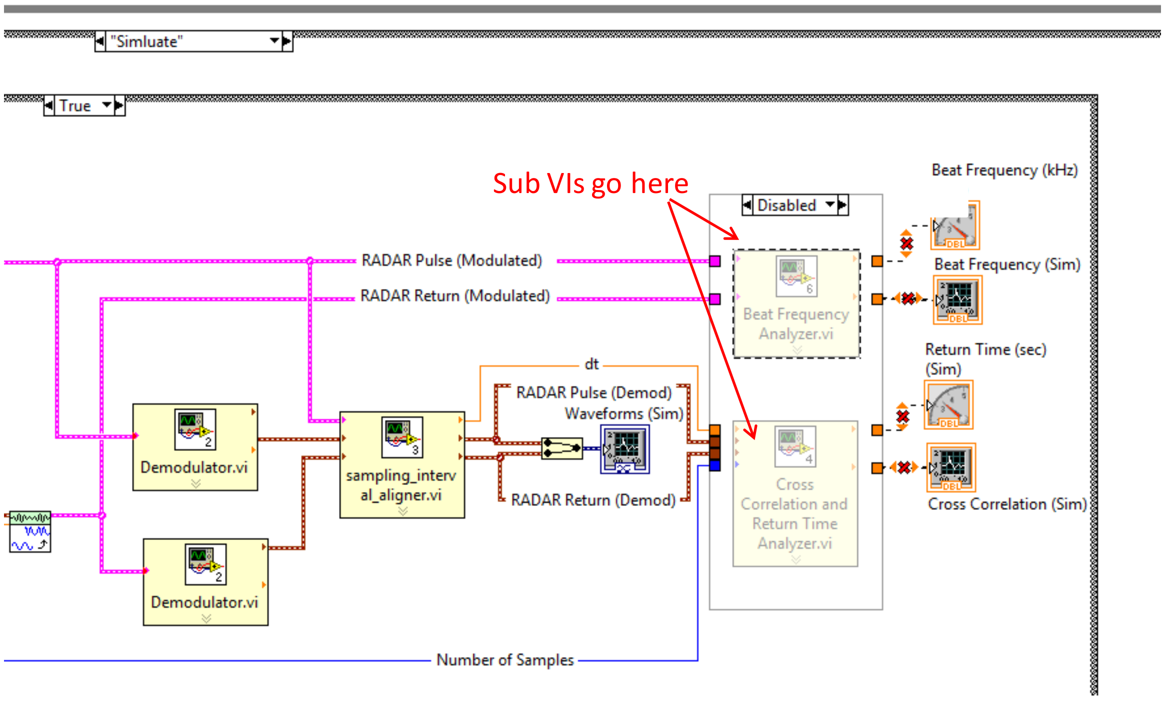


Fig. 107: Return Processing Block Diagrams

The beat frequency is determined by

a) First multiplying the return signal with RADAR pulse signal. This is equivalent to performing Amplitude modulation with these signals.
b) Next the product signal is passed through an **Envelope Detector** VI capturing the beat waveform. The envelope detector also needs the sampling frequency. For convenience we will use the number of samples in the signal to represent the sampling frequency. This can be recovered using the **Array Size** subVI from the array palette (Fig. 109).
c) The output of the envelope detector's output is then passed through a **Fast Fourier Transform** (FFT) subVI (Fig. 108). To properly isolate the beat frequency we need to use the correct size for the FFT.
d) This is done by connecting the output of the **Next Power of 2** VI to the FFT subVI. The input is the number of samples form the **Array Size** subVI.
e) The output of the FFT is passed through a low pass filter. The output is then converted to a magnitude and phase using the **Complex to Polar** VI (Fig. 110).
f) The filtered FFT magnitude is the beat frequency should be the frequency component with the largest magnitude. This frequency can been recovered by determining the index of the maximum value using the **Array Max and Min Array** VI (Fig. 109)
g) The FFT output sent to **Waveform Chart** indicator for display on the front panel. The maximum frequency Index is scaled by 0.001 and then sent to the **Gauge** indicator.

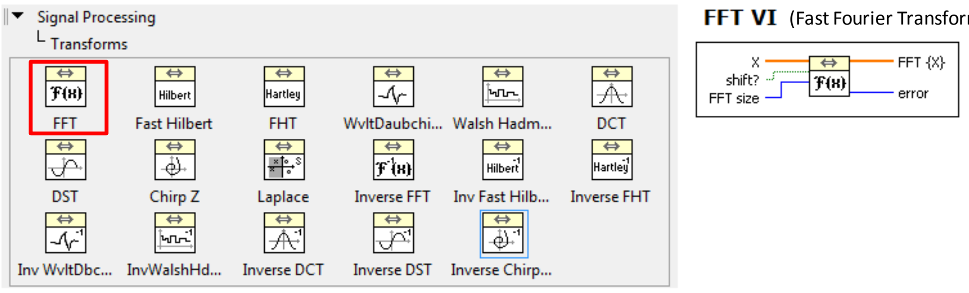


Fig. 108: Fast Fourier Transform On Signal Processing Palette

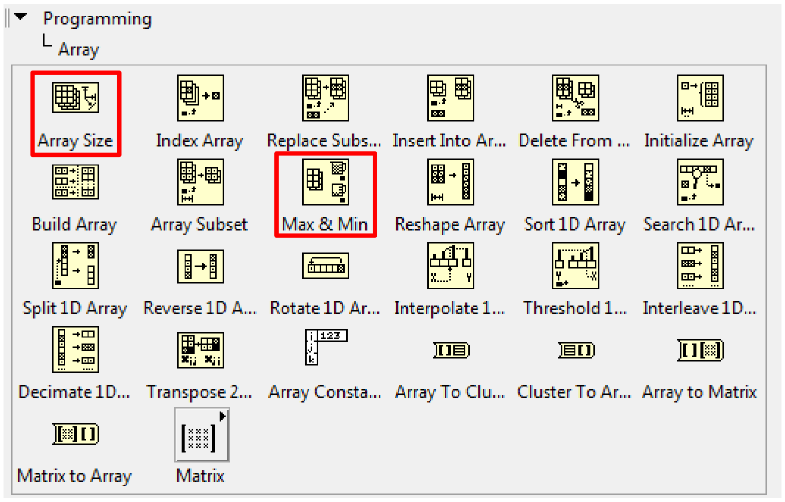


Fig. 109: Array Palette

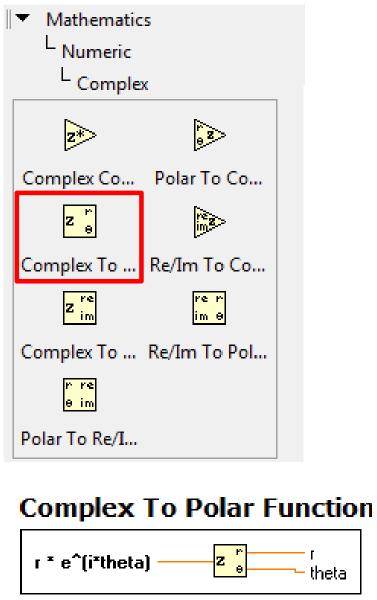


Fig. 110: Mathematics Palette

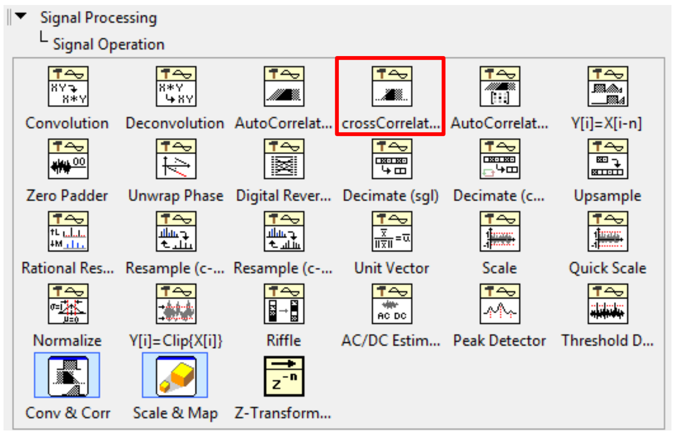
The cross-correlation function will use the **Cross Correlation and Return Time Analyzer** VI from the Signal Processing Palette (Fig. 111).

a) The x-input should be wired to receive the RADAR Pulse and the y-input should be connected to RADAR Return waveforms. (Note: A negative lag time will result if they are connected otherwise. The **algorithm** input to the VI specifies the correlation method to use. Configure the VI computes the cross correlation using an FFT-based technique. Since the waveforms have a large number of points, the **frequency domain** method is faster (use a constant with the value 1).

b) The VI outputs the cross-correlation coefficients. The correlation coefficients are displayed on **Waveform Chart** indicator.

c) The lag index corresponding to the largest coefficient (the lag where the two waveforms are most alike) can be found using **Array Max and Min Array** VI (Fig. 109).

d) The time delay can be recovered by subtracting the middle index (l_{center}) from (l_{max}) and multiplied by the sampling period ($T_{sampling}$). It should be noted that the cross-correlation has 2 times the number of samples of the signals. So we can use the number of samples as the value of l_{center}.



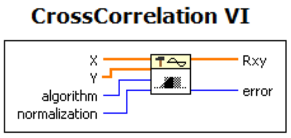


Fig. 111: Signal Processing Palette

8.3.3 Radar Simulation (Debugging of sub-VIs)

Like the RADAR receiver, the simulation uses the same VIs developed in section 2.2 (Fig. 112).

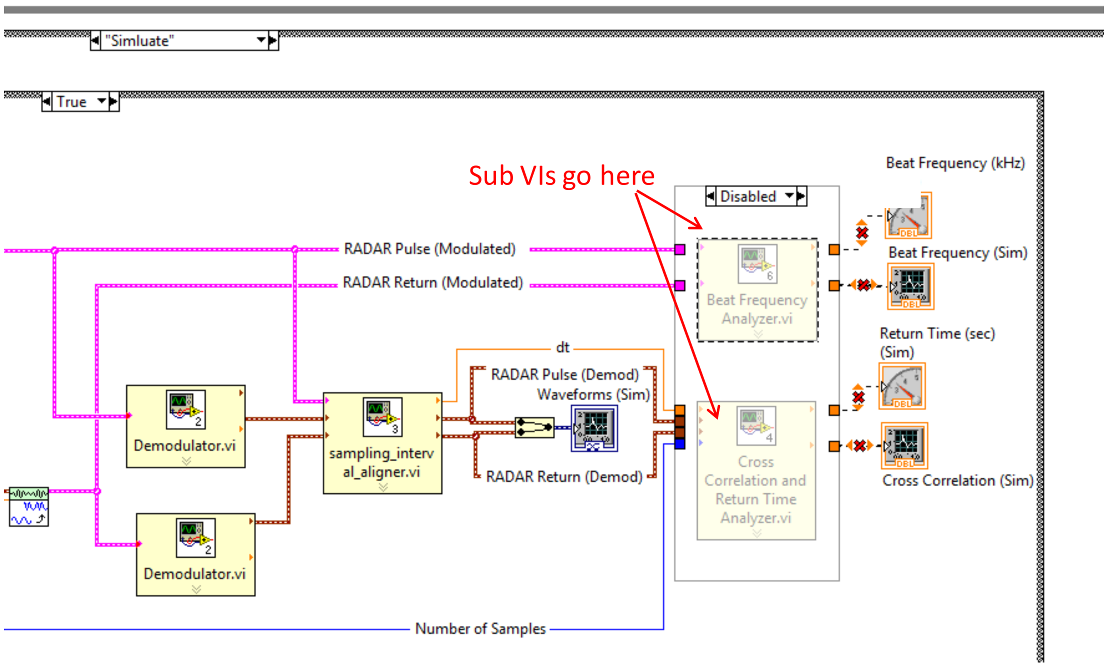


Fig. 112: Radar Return Processing Block Diagram

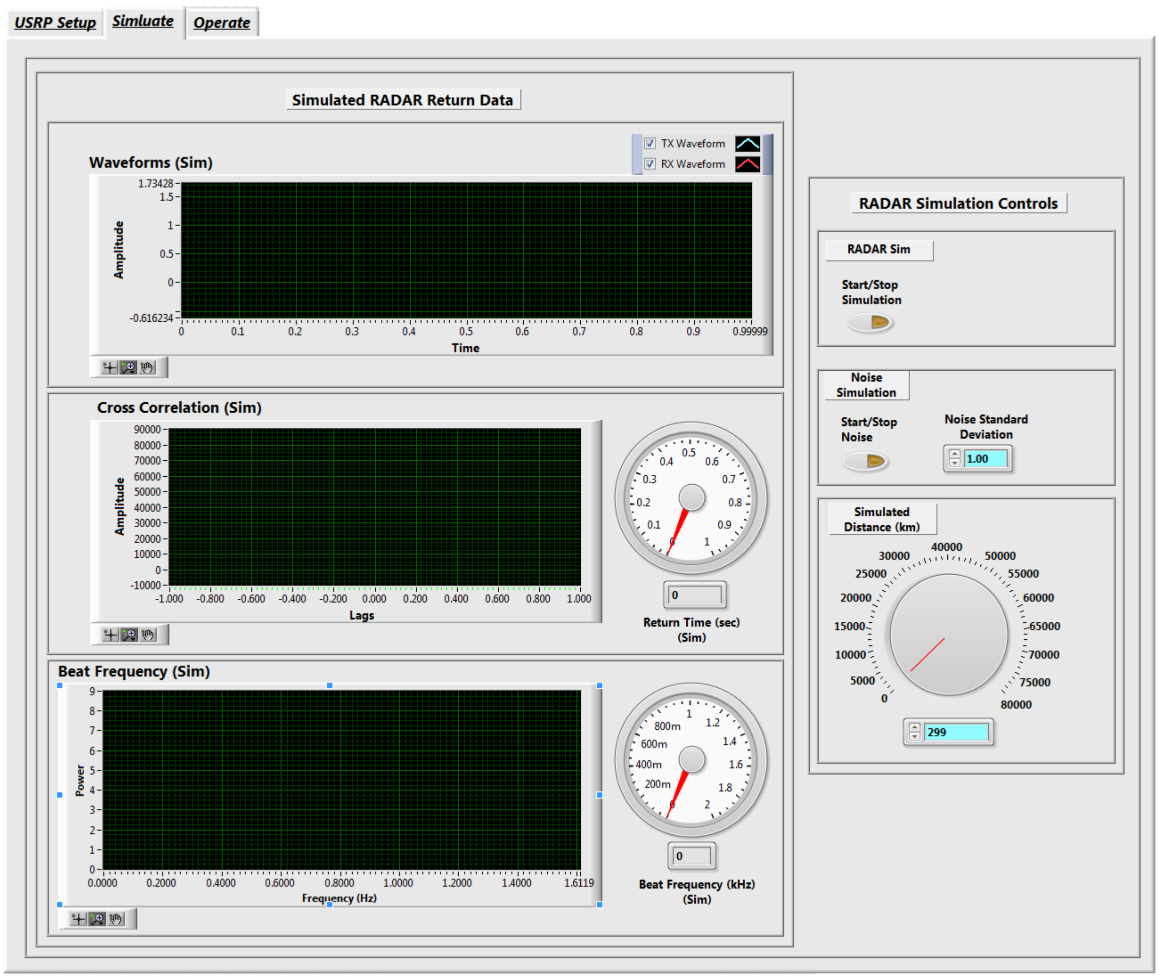


Fig. 113: RADAR Simulation Page Front Panel

Since the display measurement sub-VIs are the same, it is possible to use the simulator to check your work. A reference point you can use is given in Table XVIII.

Table XVIII – 20,000km Test Case Reference

Simulated Distance to Target. (km)	*Return Signal Ramp Reset Time (Sec)*	Return Time (Sec)	Beat Frequency (Hz)
20000	0.86728 (see **Fig. 115**)	0.13272 (see Fig. 116)	0.337 (see Fig. 117)

8.4 Lab Procedure

8.4.1 Worksheet: The Effect of Distance to Target on Cross-correlation and Beat Frequency Measurements

Note: The lab space is at most 20 meters in length. Using (53), a RADAR target at 20 meters would imply a delay of 0.00667μs (20m/299,792,458m/s) or 14.99GHz this is above the frequency threshold for the radios. A RADAR simulation (Fig. 113) has been provided to better understand how the cross correlation and beat frequency processing perform.

1. Open the RADAR Simulator page of the **J2 RADAR v2.vi**. The page should look like Fig. 113.
2. Start LabVIEW and then click on the start simulation button (see Fig. 114).

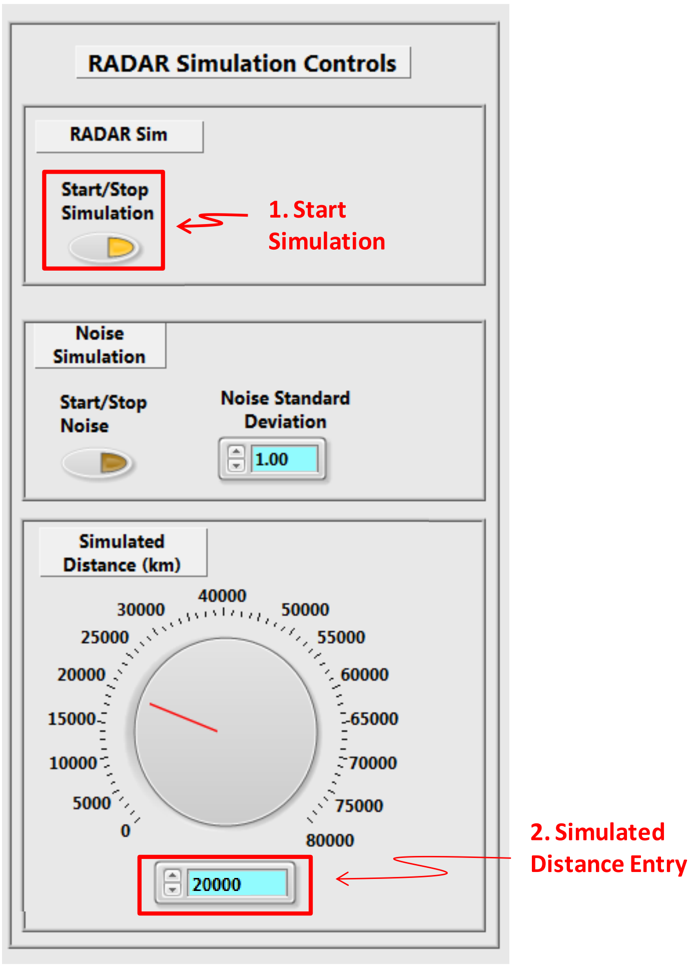


Fig. 114: RADAR Simulation Controls

3. Set the Simulated distance to the first value in Table XIX.
4. Observe the Waveforms (Sim) display (**Fig. 115**) and record the **Return Signal Ramp Reset Time** in Table XIX. You will have to use the waveform graph display tools to magnify the x-axis to do this.
5. Observe the Sim Cross Correlation display (Fig. 116) and record the **Return Time (Sec)** in Table XIX.
6. Observe the Beat Frequency (Sim) display (Fig. 117) and record the **Beat Frequency (Hz)** in Table XIX and calculate the return time using calibration equation.

$$\tau = \frac{15}{31}(f_{beat} - 0.0416) \qquad (59)$$

For example, at 20,000 km the beat frequency is 1.956 Khz so 1.956×10^4 multiplied by 0.25 is 0.1327 seconds

7. Repeat steps 4 through 6 for each distance entry in Table XIX.

Table XIX – Distance to Target Measurements

Simulated Distance to Target. (km)	***Return Signal Ramp Reset Time (Sec)***	**Return Time (Sec)**	**Beat Frequency (Hz)**	Beat Frequency Return Time (sec)
1000				
2000				
4000				
8000				
16000				
28000				
30000				

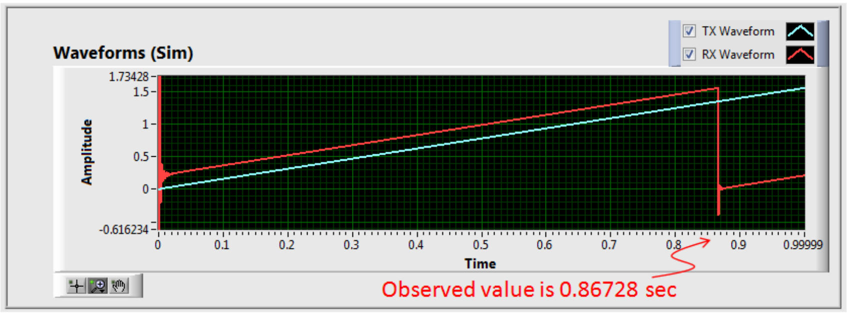


Fig. 115: Reading Waveforms (Sim) Display

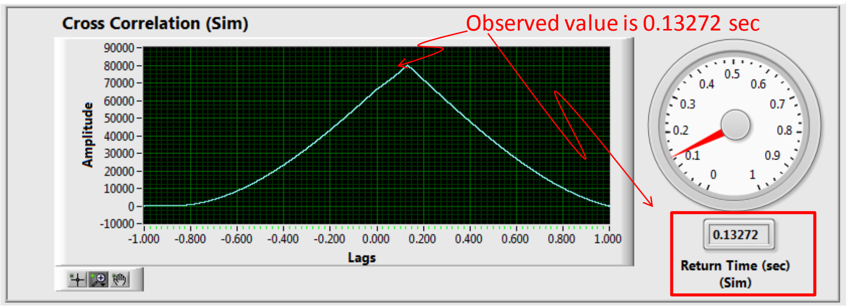


Fig. 116: Sim Cross Correlation Display

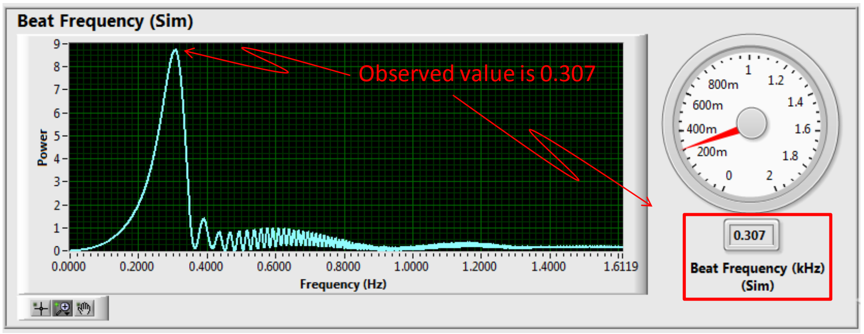


Fig. 117: Beat Frequency Display

8.4.2 Worksheet: The Effect of Noise on Cross-correlation and Beat Frequency Measurements

Note: The lab space is at most 20 meters in length. Using (53), a RADAR target at 20 meters would imply a delay of 0.00667μs (20m/299,792,458m/s) or 14.99GHz this is above the frequency threshold for the radios. A RADAR simulation (Fig. 113) has been provided to better understand how the cross correlation and beat frequency processing perform.

1. Open the RADAR Simulate page of the J2 RADAR v2.vi. The page should look like Fig. 113.
2. Start LabVIEW and then click on the start simulation button (see Fig. 118)

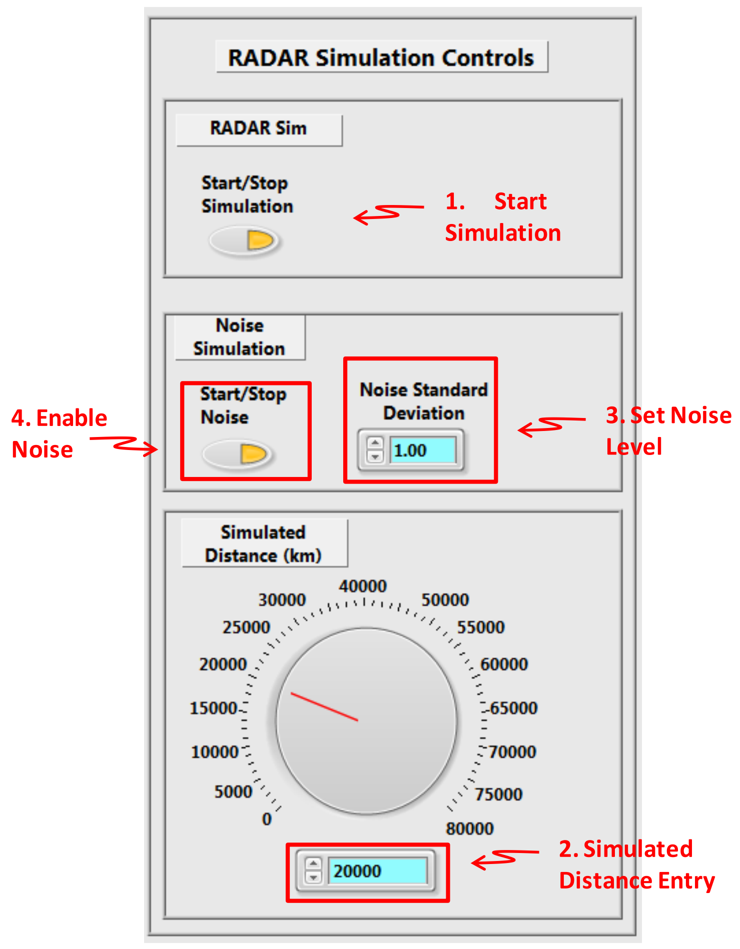


Fig. 118: RADAR Simulation Controls

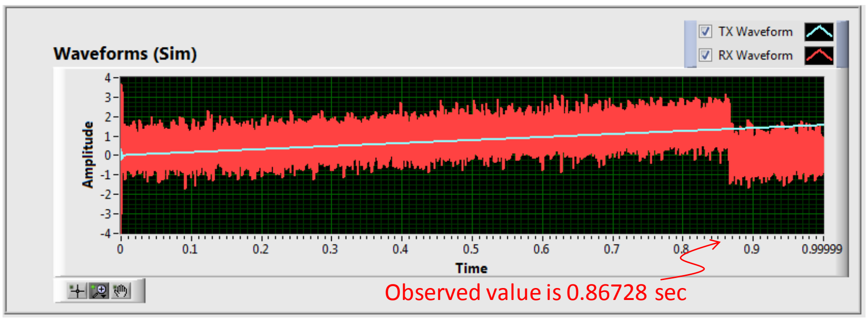
3. Set the noise level to 1 standard deviation (Fig. 118).
4. Enable noise injection into the return signal by clicking on "Star t/Stop" Noise button (Fig. 118).
5. Set the Simulated distance to the first value in Table XX.
6. Observe the Waveforms (Sim) display (Fig. 119) and record the **Return Signal Ramp Reset Time** in Table XX \. You will have to use the waveform graph display tools to magnify the x-axis to do this.
7. Observe the Sim Cross Correlation display (Fig. 120) and record the **Return Time (Sec)** in Table XX.
8. Observe the Beat Frequency (Sim) display (Fig. 121) and record the **Beat Frequency (Hz)** in Table XX and calculate the return time using calibration equation (59). For example, at

20,000 km the beat frequency is 1.956 Khz so 1.956×10^4 multiplied by 0.25 is 0.1327 seconds.

9. Repeat steps 4 through 6 for each distance entry in Table XX.

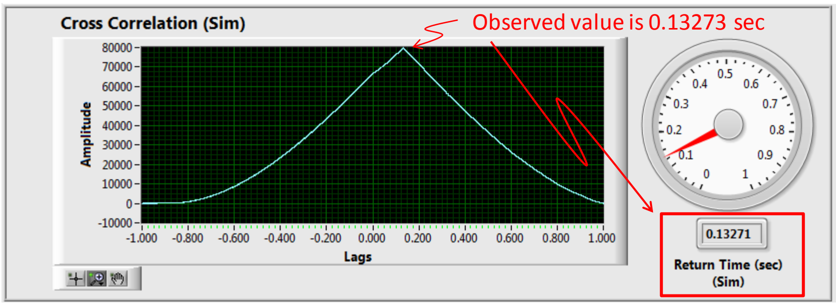
Table XX – Distance to Target With Noise Measurements

Simulated Distance to Target. (km)	*Return Signal Ramp Reset Time (Sec)*	Return Time (Sec)	Beat Frequency (Hz)	Beat Frequency Return Time (sec)
1000				
2000				
4000				
8000				
16000				
28000				
30000				



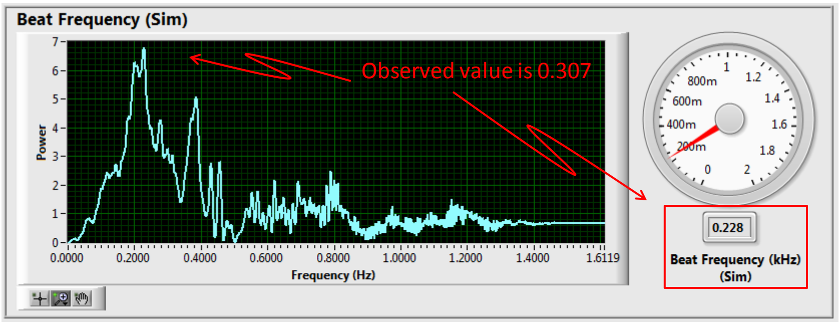
Observed value is 0.86728 sec

Fig. 119: Reading Waveforms (Sim) Display



Observed value is 0.13273 sec

Fig. 120: Sim Cross Correlation Display



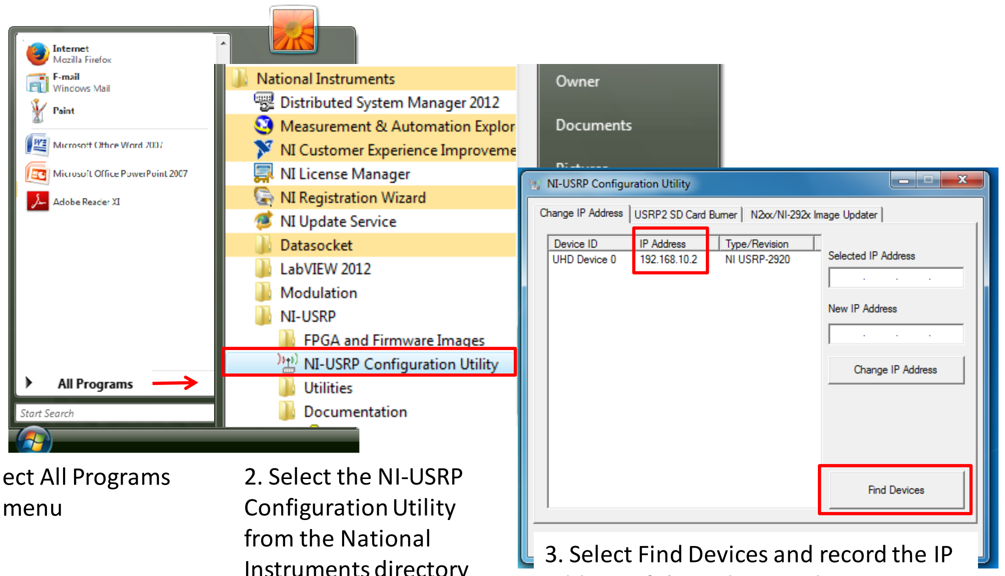
Observed value is 0.307

Fig. 121: Beat Frequency Display

8.4.3 Worksheet: RADAR Operation

Note: The lab space is at most 20 meters in length. Using (53), a RADAR target at 20 meters would imply a delay of $0.00667\mu s$ (20m/299,792,458m/s) or 14.99GHz this is above the frequency threshold for the radios.

1. Connect the computer to the USRP using an Ethernet cable.
2. Open the NI-USRP Configuration Utility found in the National Instruments directory under programs files as shown in Fig. 2. Be sure to record the IP addresses since you will need them to configure your software.



Device ID	IP Address	Type/Revision
UHD Device 0	192.168.10.2	NI USRP-2920

1. Select All Programs from menu

2. Select the NI-USRP Configuration Utility from the National Instruments directory

3. Select Find Devices and record the IP address of the radio or radios since you will need them to configure the software in the lab.

Fig. 122: Finding the IP Address Radio Connectivity Test

If the IP address does not appear in the window then check your connections and ask the Teaching Assistant (TA) to verify that the LAN card has been configured correctly.

Important set-up notes:
- ✓ Determine the IP addresses for your radios using the NI-USRP configuration utility..
- ✓ Make sure the Tx and Rx VIs are always set to the same carrier frequency whenever you pair them up to communicate.
- ✓ Transmission should start after receiving workstations are ready to receive.
- ✓ Verify that device name fields in both Tx and Rx VIs are set to the IP address of the URSP in use (see Fig. 122).
- ✓ Make sure to connect the provided attenuator between the receiver USRP's Rx input and the antenna/loopback-cable. The attenuator is used to decrease the power level of the transmitted signal in order to avoid a high power signal at the receiver's end, due to the radios proximity.

3. Connect RX2 antenna to one of the two cantennas in the RADAR antenna fixture of the receiving USRP. Connect other cantennas to the TX1/RX1 antenna connector to the transmitting USRP.

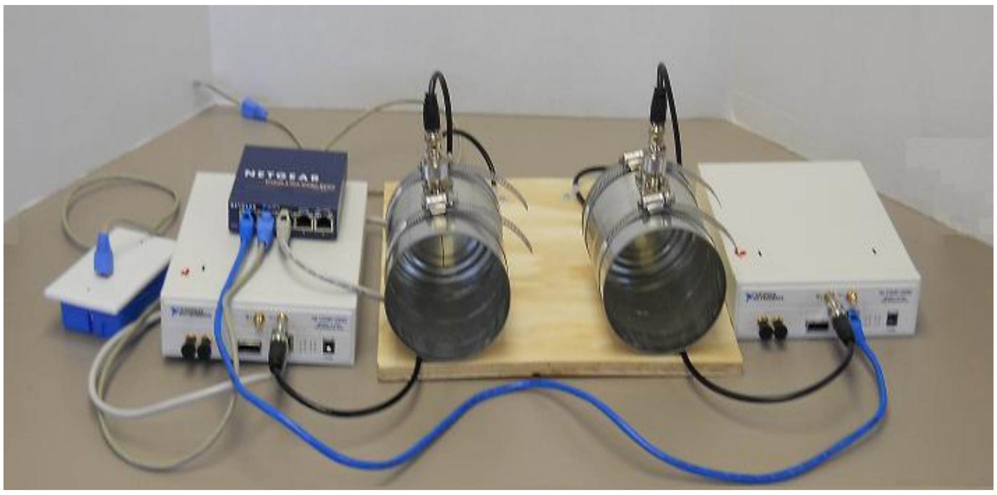
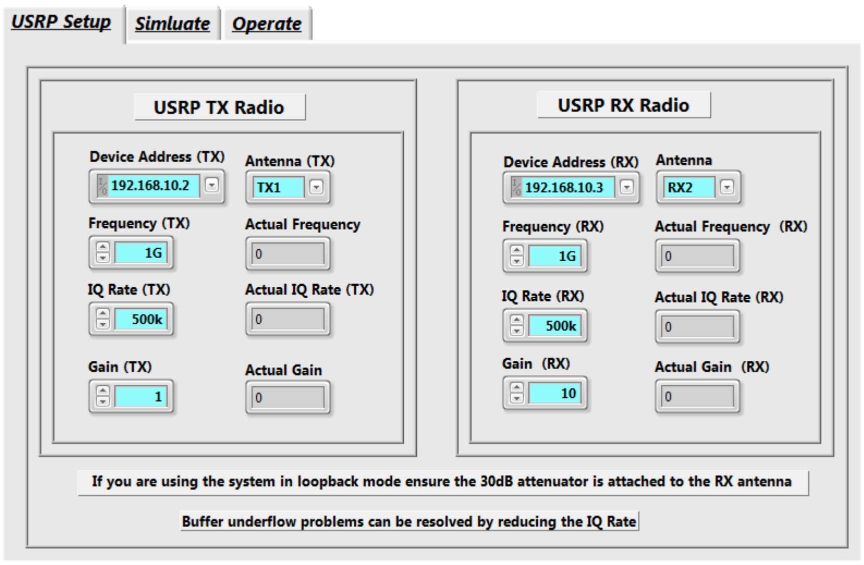


Fig. 123: RADAR Connectivity

4. Using the USRP Setup page (Fig. 124) ensure that the transmitter USRP is setup according to Table XXI.

Table XXI – Transmitter Settings

Field	Setting
Device Name:	192.168.10.*
Carrier Frequency:	1GHz
IQ Rate:	500kHz
Gain:	1
Active Antenna:	TX1



USRP TX Radio

Device Address (TX): 192.168.10.2
Antenna (TX): TX1
Frequency (TX): 1G
Actual Frequency: 0
IQ Rate (TX): 500k
Actual IQ Rate (TX): 0
Gain (TX): 1
Actual Gain: 0

USRP RX Radio

Device Address (RX): 192.168.10.3
Antenna: RX2
Frequency (RX): 1G
Actual Frequency (RX): 0
IQ Rate (RX): 500k
Actual IQ Rate (RX): 0
Gain (RX): 10
Actual Gain (RX): 0

If you are using the system in loopback mode ensure the 30dB attenuator is attached to the RX antenna

Buffer underflow problems can be resolved by reducing the IQ Rate

Fig. 124: USRP Setup Page

5. Again using the RADAR Setup Page (Fig. 124), setup the USRP according to Table XXII.

Table XXII – Receiver Settings

Field	Setting
Device Name:	192.168.10.*
Carrier Frequency:	1GHz
IQ Rate:	500kHz
Gain:	10
Active Antenna:	RX2

6. Open the RADAR Operate page of the **J2 RADAR v2.vi**. The page should look like Fig. 125.
7. Start LABView and then click on the start Start TX button (see Fig. 125)
8. Observe the Waveforms (Sim) display (Fig. 119) and record the **Return Signal Ramp Reset Time** in **Table XXIII**. You will have to use the waveform graph display tools to magnify the x-axis to do this.
9. Observe the Sim Cross Correlation display (Fig. 120) and record the **Return Time (Sec)** in **Table XXIII**.
10. Observe the Beat Frequency (Sim) display (Fig. 121) and record the **Beat Frequency (Hz)** in **Table XXIII**.

Table XXIII – Actual RADAR Observations

Return Signal Ramp Reset Time (Sec)	Return Time (Sec)	Beat Frequency (Hz)

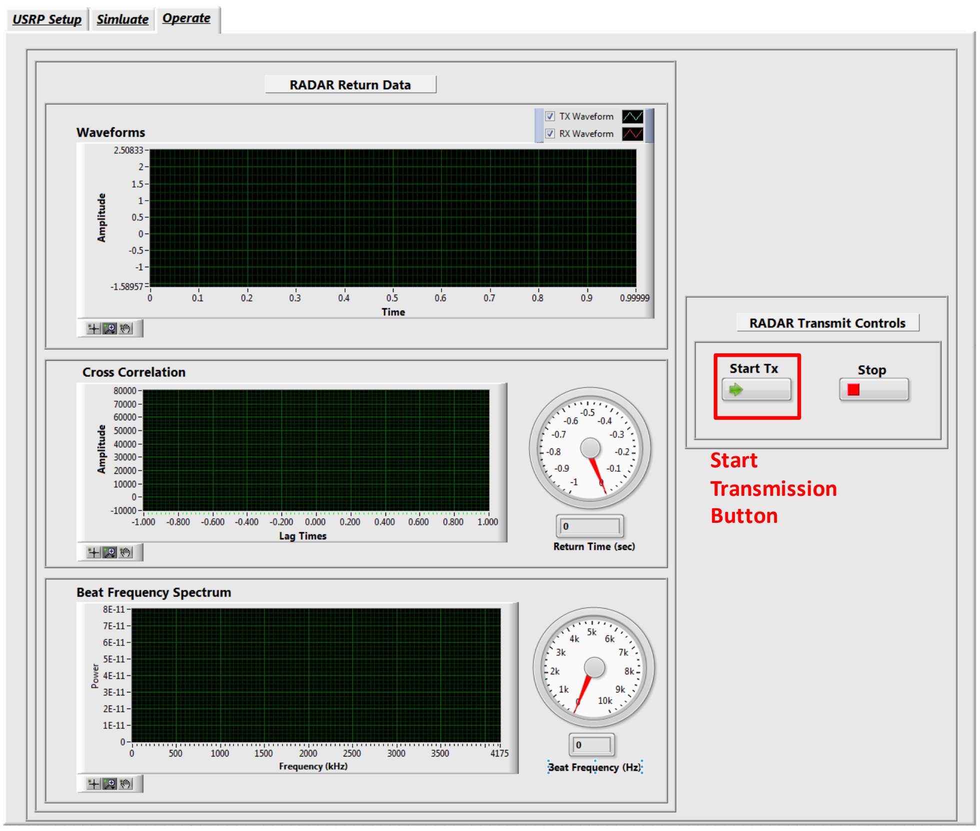


Fig. 125: RADAR Operate Page Front Panel

8.5 Lab Write-up

Short Answer Questions

1. What are the benefits of using cross correlation or beat frequency to measure distances?

2. Discuss possible reasons for having a constant of set in the RADAR calibration equation (59) which is not predicted by the beat frequency equation given by equation (57)?

3. What is the minimum range of detection for your RADAR system using cross correlation?

Performance Measures

Task	**Standards**	**Sat/Unsat**
Hardware Setup	Working setup for all with Loopback-cable.	
Running VIs	Successful transmission and reception of square wave signal.	
Determine Range and Velocity	Setup VI's to determine range and velocity to targets.	

Discussion

Did all configurations perform as expected?
Did you have any difficulties completing the lab?
Did your TA provide enough guidance?
Do you have any recommendations to improve the lab?

8.6 References

[1] John G. Proakis and Dimitris G. Manolakis, Chapter #:, In “Digital Signal Processing”, 3rd Edition.
[2] R.E. Ziemer and W.H. Tranter, Chapter 2: Signal and Linear System Analysis, In “Principles of Communications: Systems, Modulation and Noise”, Fifth Edition.
[3] Wolfram Demonstrations Project, Marshall Bradley. “Frequency-Modulated Continuous-Wave (FMCW) Radar” retrieved August 11, 2014, from http://demonstrations.wolfram.com/FrequencyModulatedContinuousWaveFMCWRadar.
[4] Cantenna Design Tutorial, Jeffrey Love, University of New Mexico.

9 Amplitude Modulation with Additive Gaussian White Noise

9.1 Summary

This laboratory exercise has two objectives. The first is to gain experience in implementing a white noise source in LabView. The second is to investigate classical analog amplitude modulation [1] and the effects of noise on the modulated signal envelope.

9.2 Background

9.2.1 Additive Gaussian White Noise

Additive white Gaussian noise (**AWGN**) is used to simulate the effect of many random processes too complicated to model explicitly. These random processes are the result of many natural sources, such as:

- Thermal noise is the result of vibrations of atoms in conductors resulting thermal energy;
- Shot noise is the result of random fluctuations in the movement of current in discrete electric charge quanta or electrons.
- Electromagnetic radiation emitted by the sun, earth and other large masses in thermal equilibrium.
- In the case of this lab, the distance between the transmitter and receiver, and background radiation from other nearby transmitters.

AWGN is also used to simulate other types of noise such as background noise and interference between other transceivers in the network.

The model is assumed to be linear so that the noise can be super imposed or added to the message or modulated signal. A white noise process is assumed to uniformly affect all frequencies in the signal's spectrum. When the noise only affects a part of the spectrum it is referred to as colored noise. The noise in the time-domain results in a sequence of random terms that are added to the signal's amplitude. These values are determined by sampling a random process with a zero-mean-normal distribution. A mean of zero is used since the process is not expected add a dc bias. A normal distribution is used because of the central limit theorem [2]. This theorem states that a random process that is sum of a large number of random variables tends toward a normally distribution random process.

All this said, for the purposes of this lab it is a good enough approximation of background radiation sources and can easily illustrate its effects on an AM radio station channel.

The AGWN is simulated using a pseudorandom number generator whose statistical profile is $N(0,\sigma^2)$, where σ^2 is the variance of random number generator whose output conforms to the following probability density function.

$$f_N(N=n)=\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{n^2}{2\sigma^2}\right) \tag{60}$$

A "pseudorandom number generator" is an algorithm for generating a sequence of numbers that approximates the properties of random numbers. In terms of random number generation, the sequence is not truly random in that it is completely determined by a relatively small set of initial values, called the pseudorandom number generator *state*, which includes a truly random seed. Although sequences that are closer to truly random can be generated using hardware random number generators, *pseudorandom* numbers are important in practice for their speed in number generation and their reproducibility.

This is done using the Box-Muller Transformation, which uses two random variables to represent the angle and the radius of a circle. The radius and the angle (U_1 and U_2) are generated as independent random variables that are uniformly distributed over the interval (0, 1]. When transformed back into a two-axis Cartesian coordinate (61), the coordinates represent two independent random variables with a standard normal distribution on the interval (0, 1].

$$\begin{aligned} Z_0 &= \sqrt{-2\ln(U_1)}\cos(2\pi U_2) \\ Z_1 &= \sqrt{-2\ln(U_1)}\sin(2\pi U_2) \end{aligned} \tag{61}$$

In LABVIEW this is implemented with the Gaussian White Noise Waveform VI (Fig. 128).
The standard deviation can be estimated using a spectrum analyzer to observe the noise floor (Fig. 126). The standard deviation is calculated using (19). For the purposes of the lab you will want to set this at specific values to assess the impact of noise on the AM receiver.

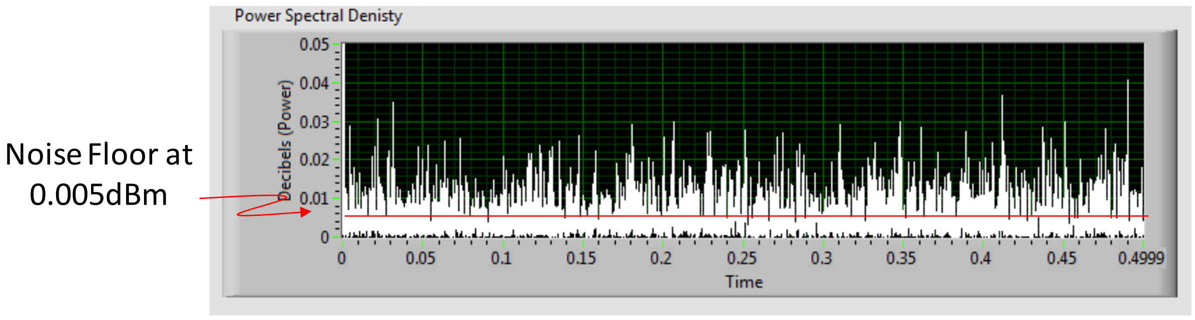


Fig. 126: Noise Floor Measurement

$$\sigma = \sqrt{2\cdot(Noise\ Floor)} \tag{62}$$

In addition to measuring the noise floor, the signal-to-noise and distortion ratio (**SINAD**) can be used to measure the residual noise and distortion in a signal after being filtered or, in the case of a transmitter, the amount of noise and distortion introduced as a result of modulation. SINAD is defined as the ratio (63) of the RMS energy of the received noisy signal to the RMS energy of received noisy signal less the energy in the fundamental frequency, expressed in dB. This implies that the higher the ratio is, the lower the power of the noise would be. As you will see in the lab, this implies the receiver will have a better chance of detecting the envelope.

$$SINAD = \frac{P_{signal} + P_{noise} + P_{distortion}}{P_{noise} + P_{distortion}} \tag{63}$$

9.3 Pre-Lab

9.3.1 Transmitter

Your task is to add a data entry pane to the transmitter template (Fig. 127) to allow addition of white Gaussian noise to the amplitude modulated signal in AM_Noise_Tx_Template.vi.

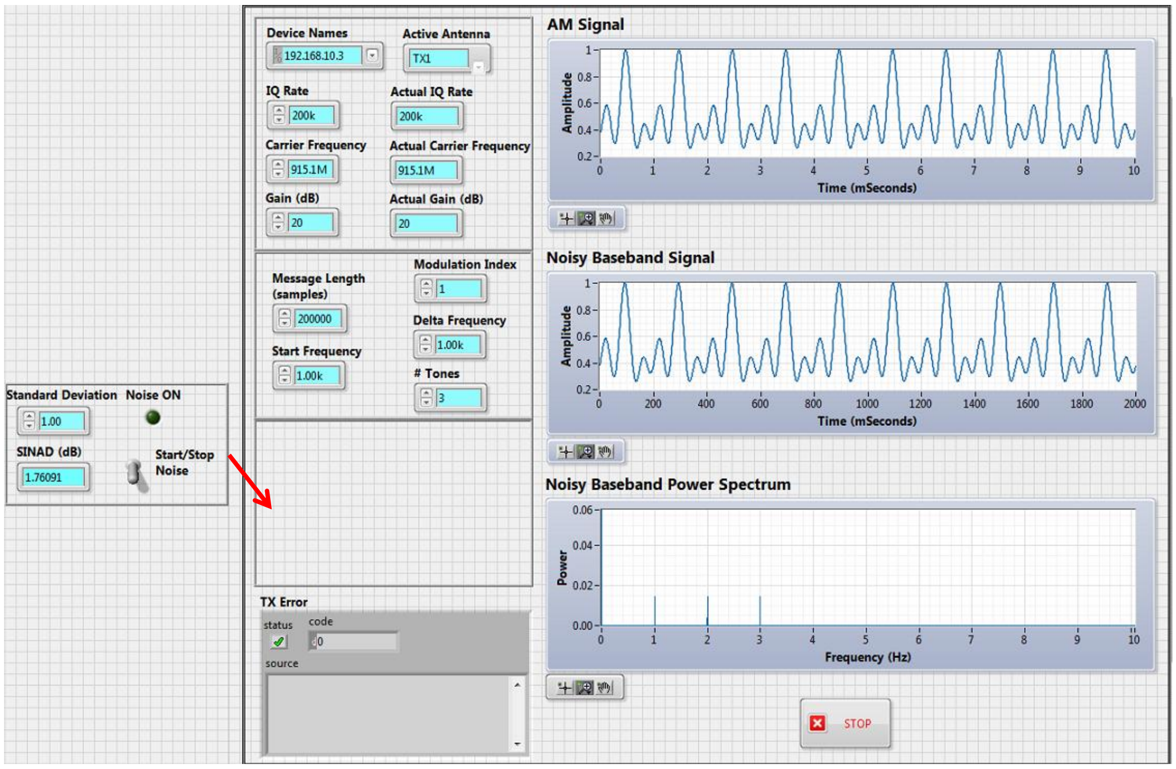
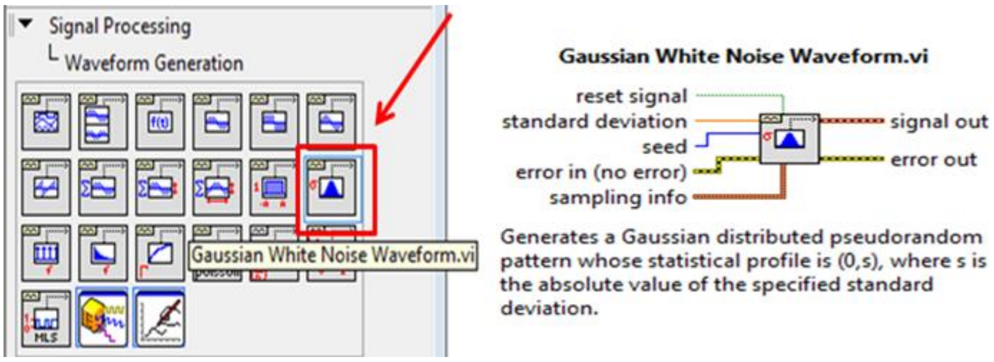

Device Names: 192.168.10.3
Active Antenna: TX1
IQ Rate: 200k
Actual IQ Rate: 200k
Carrier Frequency: 915.1M
Actual Carrier Frequency: 915.1M
Gain (dB): 20
Actual Gain (dB): 20
Message Length (samples): 200000
Modulation Index: 1
Delta Frequency: 1.00k
Start Frequency: 1.00k
Tones: 3
Standard Deviation: 1.00
SINAD (dB): 1.76091
TX Error: status ✓, code 0, source

Fig. 127: Transmitter VI Template Front Panel

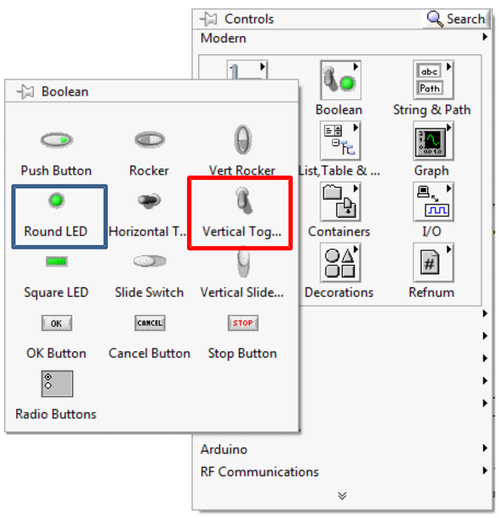
a) Generate a white noise source using the “Gaussian White Noise Waveform” VI (**Fig. 128**). The VI generates a Gaussian distributed pseudorandom pattern with a mean of 0 and a standard deviation (σ), where σ is the absolute value of the chosen standard deviation. The value of the standard deviation input is controlled by a switch and case structure as explained in the coming paragraphs. The sampling information for the generated white noise should be kept the same as the message signal generated by the basic multi-tone.



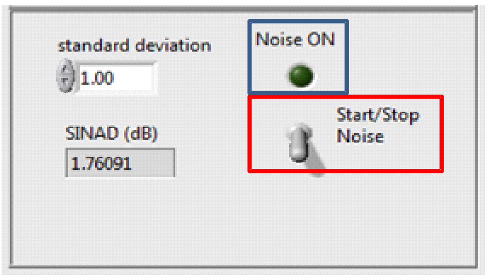
Generates a Gaussian distributed pseudorandom pattern whose statistical profile is (0,s), where s is the absolute value of the specified standard deviation.

Fig. 128: Gaussian White Noise Waveform VI

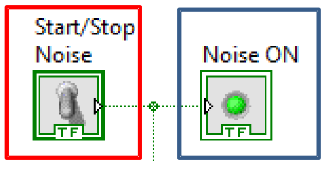
b) Insert a switch and round LED indicator on the front panel to activate or deactivate noise addition from the front panel. So when you want the noise to be added, you would turn the switch to the “On” position and the LED indicator would turn “Green”. These controls can be found in the Front Panel Boolean Controls Palette and wired as shown in Fig. 129.



1) Select Switch and Round LED from Front Panel Controls Menu



2) Arrange the LED and switch on the front panel



To Case Statement

3) Arrange the LED and switch in the block diagram

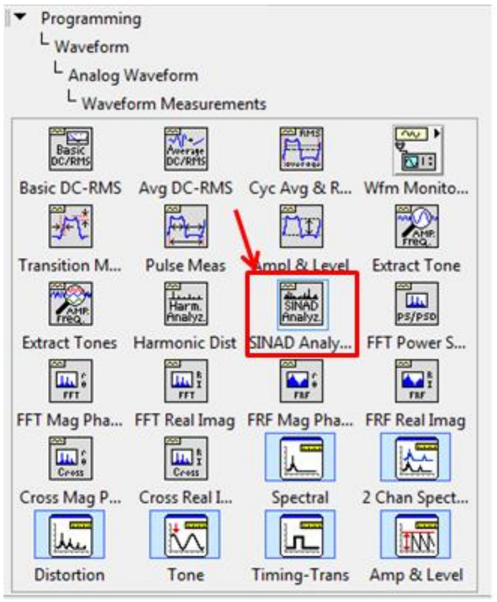
Fig. 129: Boolean Switch and LED Palette Location and Wiring

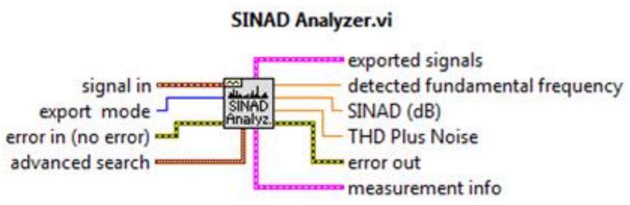
c) Design a case structure that relates to the “On” and “Off” positions of the switch. In the “On” position, the standard deviation terminal of the Gaussian White Noise Waveform Generator should receive the value of “Standard Deviation” set up from the front panel, and zero otherwise.

d) Get the data values of the generated noise using a “Get Waveform Components” VI and add them to the complex form of the scaled baseband signal.

e) Write the noisy signal to the “niUSRP Write Tx Data” VI buffer for transmission.

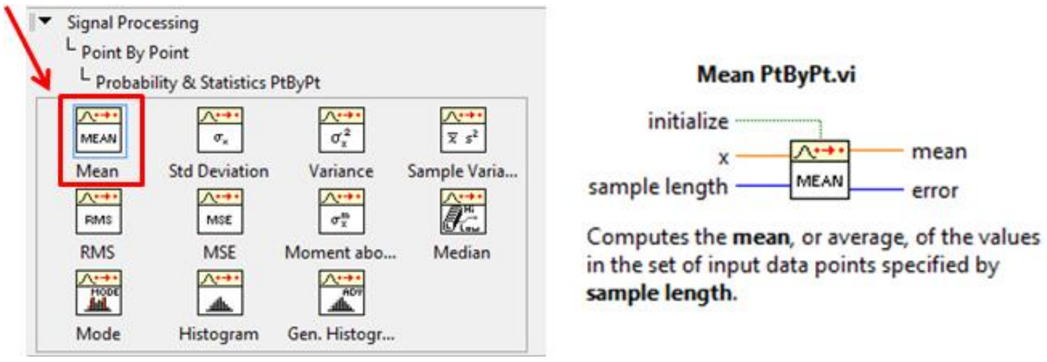
f) Do a quantitative noise analysis by feeding the noisy signal to the “signal in” input of the “SINAD Analyzer” **(**Fig. 130**)**. Set the export mode value to “input signal”. For output, display the mean of 100 SINAD values using the “Mean PtByPt” VI **(**Fig. 131**)**.





Takes a signal in and performs a full Signal in Noise and Distortion (SINAD) analysis, including measuring the fundamental frequency tone and returning the fundamental frequency and SINAD level in dB. Wire data to the **signal in** input to determine the polymorphic instance to use or manually select the instance.

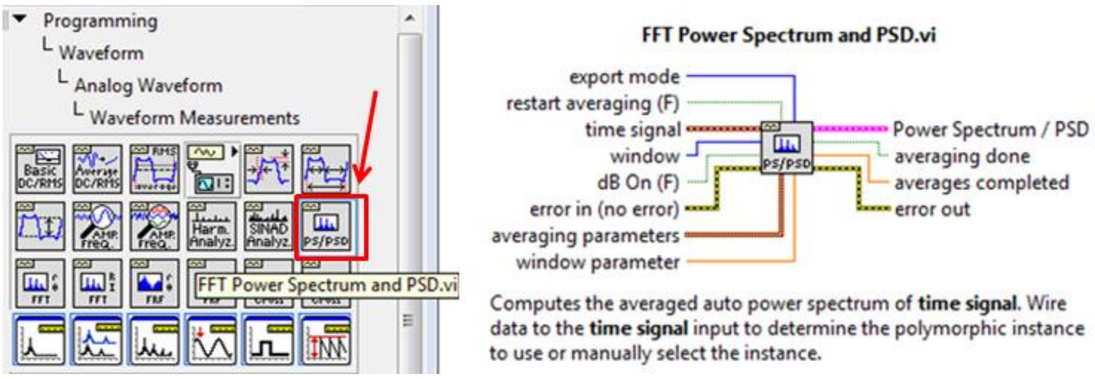
Fig. 130: SINAD Analyzer VI



Computes the **mean**, or average, of the values in the set of input data points specified by **sample length**.

Fig. 131: Mean PtByPt VI

g) Plot the noisy signal in time and frequency domains using the waveform charts and "FFT Power Spectrum and PSD" VI **(**Fig. 132**)** provided.



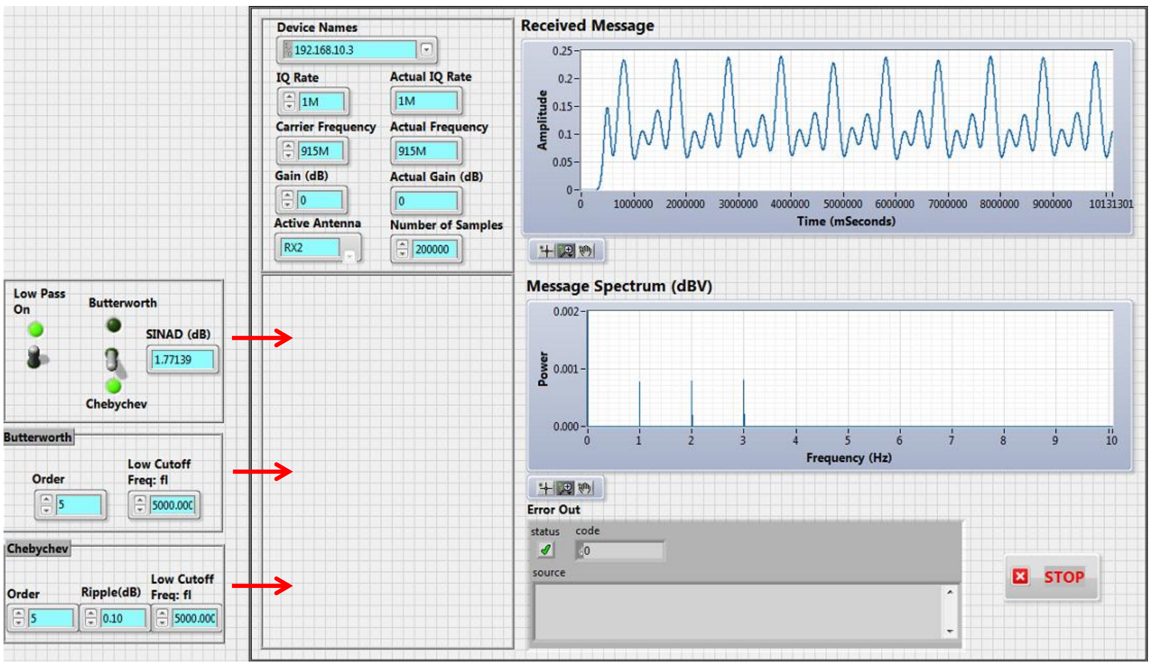
Computes the averaged auto power spectrum of **time signal**. Wire data to the **time signal** input to determine the polymorphic instance to use or manually select the instance.

Fig. 132: FFT Power Spectrum and PSD VI

e) Save your transmitter in a file whose name includes the letters "AM_Tx_Noise" and your initials.

9.3.2 Receiver

The task is to design a Boolean or digital circuit that allows the selection of no filter, a Chebyshev low-pass filter, or a Butterworth low-pass filter to extract the tones in the transmitted signal's envelope. A template for the receiver has been provided in the file AM_Noise_Rx_Template.vi (see Fig. 49).



Device Names: 192.168.10.3
IQ Rate: 1M
Actual IQ Rate: 1M
Carrier Frequency: 915M
Actual Frequency: 915M
Gain (dB): 0
Actual Gain (dB): 0
Active Antenna: RX2
Number of Samples: 200000
SINAD (dB): 1.77139

Butterworth
Order: 5
Low Cutoff Freq: fl: 5000.000

Chebychev
Order: 5
Ripple(dB): 0.10
Low Cutoff Freq: fl: 5000.000

Error Out
status: ✓
code: 0
source

Fig. 133: Receiver VI Template Front Panel

a) The Front Panel controls for the filter selection logic require you to construct the switching network shown in Fig. 134. The network implements the following logic table. The name given on the front panel will be assigned to the switch and indicator VIs.

Table XXIV – Switch and LED Settings

Switches		Indicator LEDs		
Low Pass on	**Filter Selector**	**Low Pass On**	**Chebyshev**	**Butterworth**
Off	Chebyshev	Off	On	Off
Off	Butterworth	Off	Off	On
On	Chebyshev	On	On	Off
On	Butterworth	On	Off	On

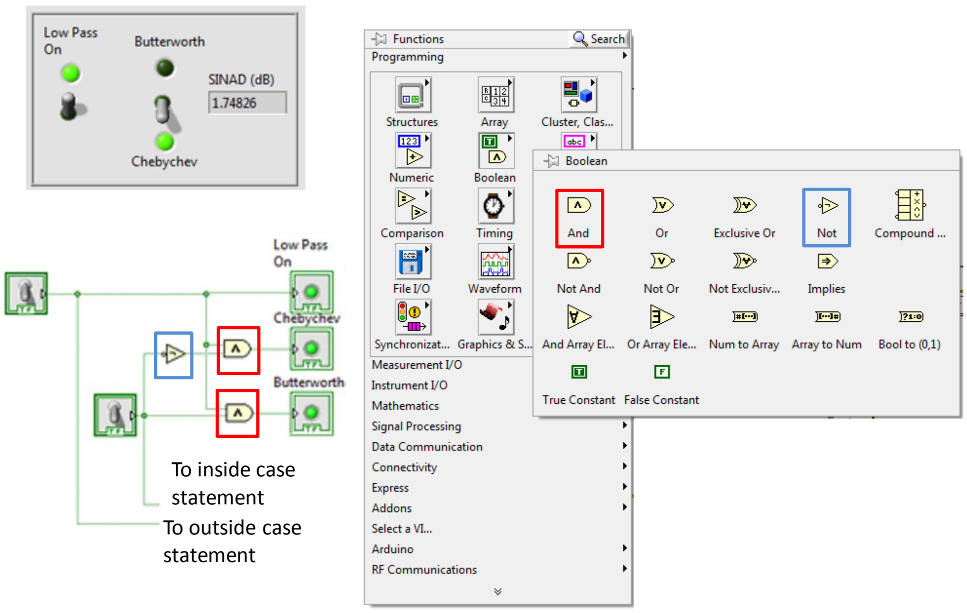


Fig. 134: Filter Selection Logic

The switches will be used to drive the nested Case Structures illustrated in Fig. 135. The nested Case structure is driven by the “Low-pass Filter On” switch. When the “Low-pass Filter On” is in the off position, the signal from the band-pass filter should be passed through with no filtering. When the “Low-pass Filter On” is in the on position, the signal is passed through the selected filter. Remember to put the right filter in the right case structure pane.

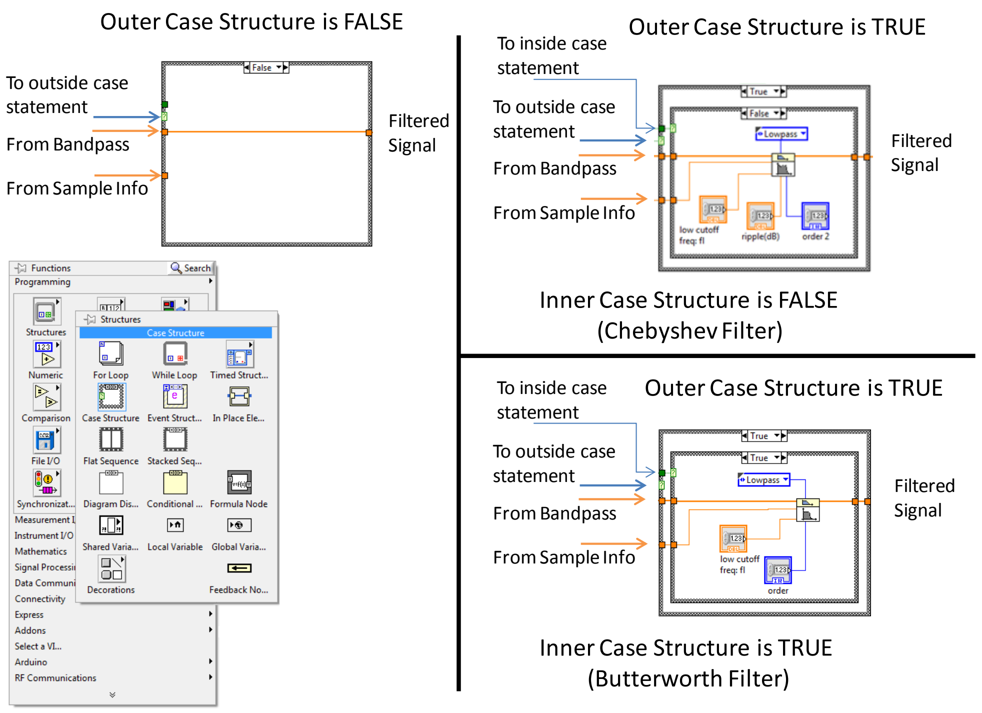
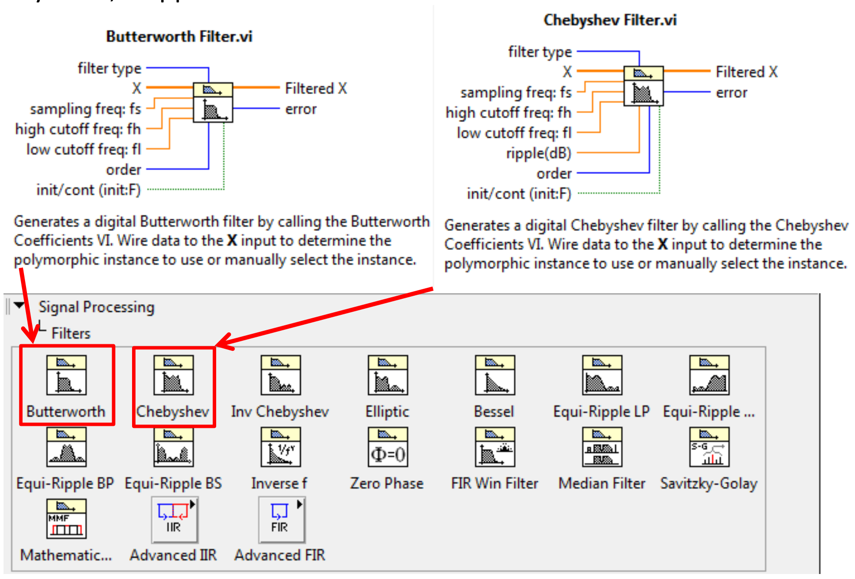


Fig. 135: Nested Case Structure for Filter Selection

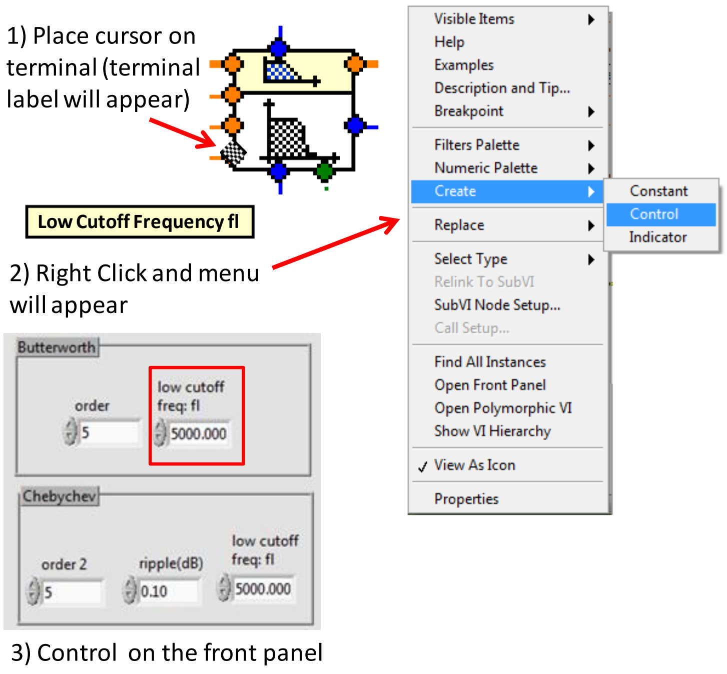
b) As in Lab 5 the filters will be found on the Filters palette shown in Fig. 136. Unlike before, you will make the “Order” and “Low Cutoff Frequency” for the Butterworth filter into control inputs from the main panel. This can be done by selecting the desired input for the VI on the left-hand side of the block and right clicking and selecting the create control option as shown in Fig. 137. For the Chebyshev filter, control inputs are the parameters “Order”, “Low Cutoff Frequency” and, “Ripple”.



Generates a digital Butterworth filter by calling the Butterworth Coefficients VI. Wire data to the **X** input to determine the polymorphic instance to use or manually select the instance.

Generates a digital Chebyshev filter by calling the Chebyshev Coefficients VI. Wire data to the **X** input to determine the polymorphic instance to use or manually select the instance.

Fig. 136: Filters Palette



Butterworth: order 5; low cutoff freq: fl 5000.000
Chebychev: order 2 5; ripple(dB) 0.10; low cutoff freq: fl 5000.000

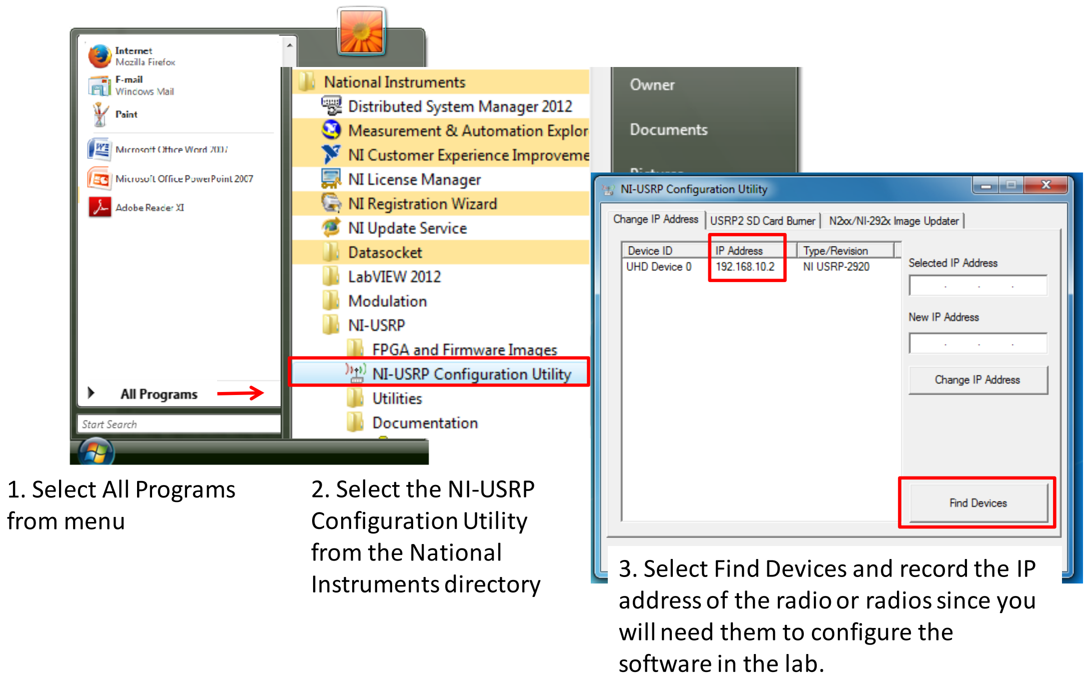
Fig. 137: Creating a Control for an Input

c) Save your receiver in a file whose name includes the letters "AM_Noise_Rx" and your initials.

9.4 Lab Procedure

Test your designed transmitter and receiver VIs for this lab by following the undermentioned steps.

1. Run LabVIEW and open the transmitter and receiver VIs that you created in the pre-lab.
2. Connect the computer to the USRP using an Ethernet cable.
3. Open the NI-USRP Configuration Utility found in the National Instruments directory under programs files as shown in Fig. 138. Be sure to record the IP addresses since you will need them to configure your software.


Internet Mozilla Firefox
E-mail Windows Mail
Paint
Microsoft Office Word 2007
Microsoft Office PowerPoint 2007
Adobe Reader XI
All Programs
Start Search
National Instruments
Distributed System Manager 2012
Measurement & Automation Explor
NI Customer Experience Improveme
NI License Manager
NI Registration Wizard
NI Update Service
Datasocket
LabVIEW 2012
Modulation
NI-USRP
FPGA and Firmware Images
NI-USRP Configuration Utility
Utilities
Documentation
Owner
Documents
NI-USRP Configuration Utility
Change IP Address
USRP2 SD Card Burner
N2xx/NI-292x Image Updater
Selected IP Address
New IP Address
Change IP Address
Find Devices
1. Select All Programs from menu
2. Select the NI-USRP Configuration Utility from the National Instruments directory
3. Select Find Devices and record the IP address of the radio or radios since you will need them to configure the software in the lab.

Device ID	IP Address	Type/Revision
UHD Device 0	192.168.10.2	NI USRP-2920

Fig. 138: Finding the IP Address: Radio Connectivity Test

If the IP address does not appear in the window then check your connections and ask the Teaching Assistant (TA) to verify that the LAN card has been configured correctly.

4. Connect a loopback cable between the TX 1 and RX 2 antenna connectors. Remember to connect the attenuator to the receiver end.

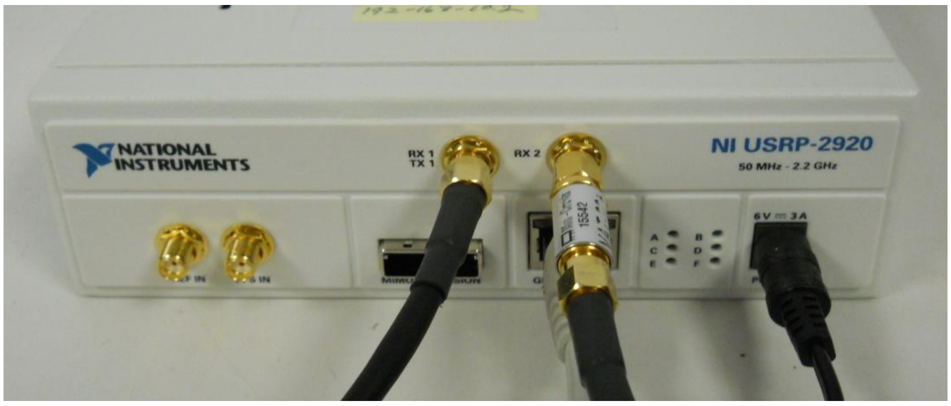


Fig. 139: Broadcast Setup

5. Ensure that the transmitter VI is set up correctly, using the following parameters:

Table XXV – Transmitter Parameters

Parameters	Values
Device Name:	192.168.10.x
Carrier Frequency:	915.1 MHz
IQ Rate:	200 kHz
Gain:	Use default.
Active Antenna:	TX1
Message Length:	200,000 samples
Modulation Index:	Start with 1.0.
Start Frequency, Delta Frequency, Number of Tones:	1kHz, 1Hz, and 3.

6. Ensure that the **receiver VI** is set up correctly, using the following parameters:

Table XXVI – Receiver Parameters

Parameters	Values
Device Name:	192.168.10.x
Carrier Frequency:	915 MHz
IQ Rate:	1 MHz
Gain:	0 dB
Active Antenna:	RX2
Number of Samples:	200,000 samples

7. Run the **receiver VI**. LED “C” will illuminate on the USRP if the radio is receiving data.
8. Run the **transmitter VI**. LED “A” will illuminate on the USRP if the radio is transmitting data.
9. After a few seconds, stop the receiver using the STOP button, then stop the transmitter. Note: Use the large STOP button on the front panel to stop transmission; otherwise the USRP may not be stopped cleanly.

10. Rescale the baseband spectrum graph to check the frequency content. It should be similar to the original message signal.

11. Use the horizontal zoom feature on the graph palette to expand the “message” waveform in the transmitter VI and the “baseband output” waveform in the receiver. Both waveforms should be identical, except for scaling and the fact that the receiver output has a DC offset.

Important set-up notes:

- ✓ Make sure the global set-up configuration has been performed before interfacing with the USRPs.
- ✓ Make sure the Tx and Rx VIs are always set to the same carrier frequency whenever you pair them up to communicate (see Table XXVand Table XXVI).
- ✓ Transmission should start after receiving workstations are ready to receive.
- ✓ Verify that the device name fields in both Tx and Rx VIs are set to the IP address of the URSP in use (see Fig. 49)
- ✓ **Make sure to connect the provided attenuator between the receiver USRP’s Rx input and the loopback-cable.** The attenuator is used to decrease the power level of the transmitted signal in order to avoid a high power signal at the receiver’s end, due to the radios’ proximity.

9.4.1 Worksheet: The Effect of Varying the Noise Level

Transmitter (**Fig**. 141) and receiver (Fig. 141) setups:

1. Set the transmitter to use <u>one</u> of the three tones. (Please note that using more than one tone will make it very hard to make the observations.)
2. Set the Start Frequency to 1 kHz.
3. Set the Standard Deviation (**Fig. 140**) to the first value in the worksheet table **Table II** below.

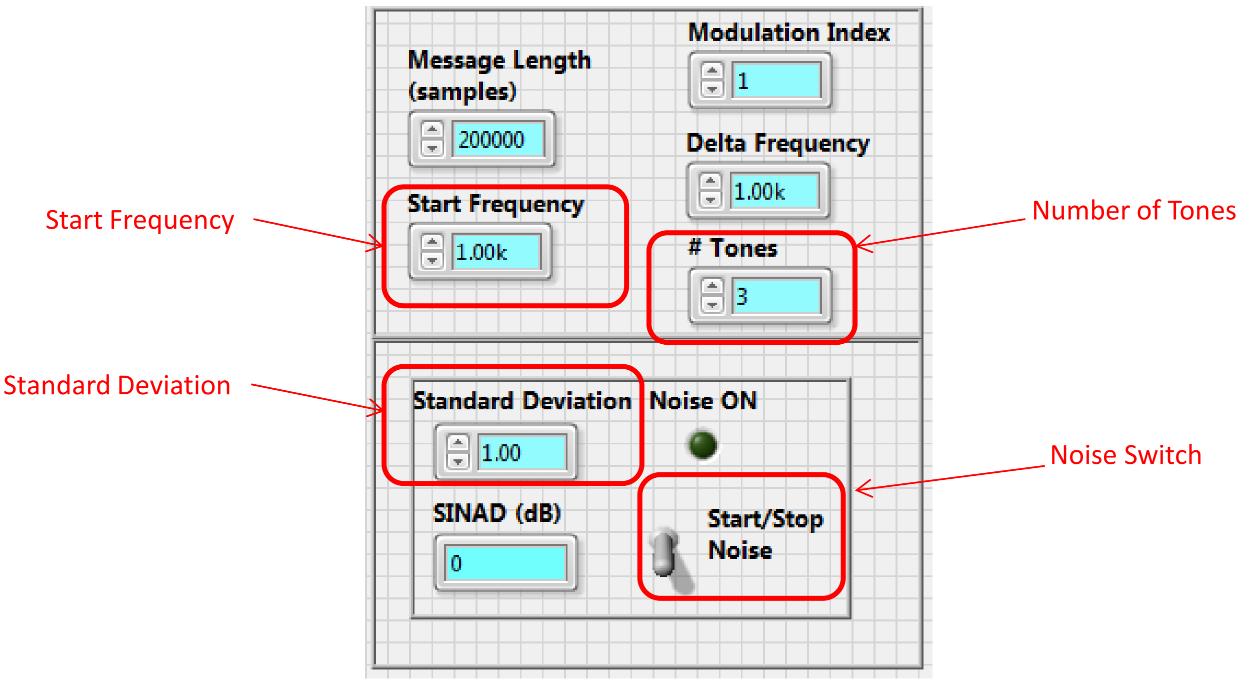


Fig. 140: TX Front Panel Noise Configuration Setup

4. Set the order of the Butterworth filter to 5.
5. Set the low-pass cut-off frequency to 5000Hz.
6. Set the order of the Chebyshev filter to 5.
7. Check the Ripple is set to 0.1 Hz.
8. Set the low-pass cut-off frequency to 5000Hz.
9. Start the receiver VI.
10. Start the transmitter VI.
11. Set the transmitter's Start/Stop Noise switch to the "On" position.
12. Set the low-pass switch to the "On" position.
13. Set the Filter Selection switch to the "Chebyshev" position.
14. Observe the SINAD value and record it on the data sheet.
15. Observe the wave received waveform and the spectrum and record whether the received wave form is recovered or not. Use a scale from 1 to 5, where 1 indicates that the waveform is entirely obscured by the noise, and 5 indicates that the waveform looks like the transmitted waveform.
16. Observe the noise floor and record it.

After each set of observations, reset the noise standard deviation to the next value in the table and record the measurements as described in steps 14, 15, 16.

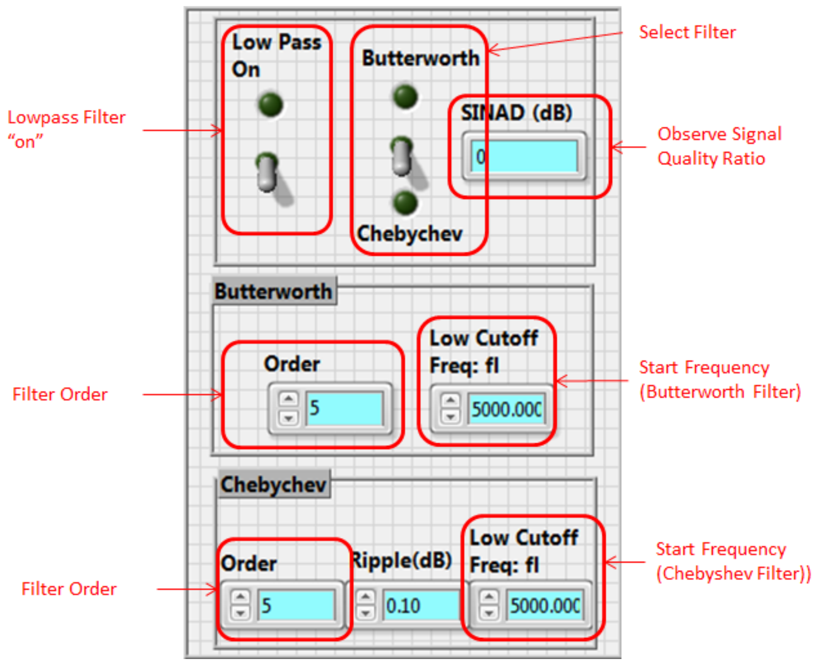


Fig. 141: RX Front Panel Filter Configuration

Table XXVII – Effect of Varying the Noise Level

Standard Deviation	SINAD	Signal Quality (1 to 5)	Noise Floor
0.0			
0.1			
0.2			
0.3			
0.4			
0.5			
0.6			
0.7			
0.8			
0.9			
1.0			

9.4.2 Worksheet: The Effect of Low-pass Order on Envelope Detection

Transmitter (**Fig**. 141) and receiver (Fig. 141) setups:

1. Set the transmitter to use one of the three tones. (Please note that using more than one tone will make it very hard to make the observations.)
2. Set the start frequency to 1 kHz.
3. Set the standard deviation of the transmitter VI (**Fig. 140**) to 0.1.
4. Set the order of the Butterworth filter to the first value in the worksheet table **(Table XXVIII)** below.
5. Set the low-pass cut-off frequency to 5000Hz.
6. Set the order of the Chebyshev filter to the first value in the worksheet table below.
7. Check the ripple is set to 0.1 Hz.
8. Set the low-pass Frequency Cutoff to 5000Hz.
9. Start the receiver VI.
10. Start the transmitter VI.
11. Set the transmitter's Start/Stop Noise switch to the "On" position.
12. Set the low-pass switch to the "On" position.
13. Set the filter selection switch to the "Chebyshev" position.
14. Observe the SINAD value and record it on the data sheet.
15. Observe the received waveform and the spectrum and record whether the received wave form is recovered or not. Use a scale from 1 to 5, where 1 indicates that the waveform is entirely obscured by the noise, and 5 indicates that the waveform looks like the transmitted waveform.
16. Set the Filter Selection switch to the "Butterworth" position.
17. Observe the SINAD value and record it on the data sheet.
18. Observe the received waveform and the spectrum and record whether the received wave form is recovered or not. Use a scale from 1 to 5, where 1 indicates that the waveform is entirely obscured by the noise, and 5 indicates that the waveform looks like the transmitted waveform.

After each set of observations, reset the filter order to the next value in the table, and record the measurements.

Table XXVIII – Effect of Low-Pass Filter Order on Envelope Detection

Filter Order	*Chebyshev SINAD*	*Chebyshev Quality (1 to 5)*	*Butterworth SINAD*	*Butterworth Quality (1 to 5)*
2				
5				
10				
15				
20				
25				
50				

9.5 Lab Write-up

Performance Checklist

Amplitude Modulation with noise

Short Answer Questions

1. What happens to the AM signal if the noise floor is high?

2. What is the relationship between increased filter order and type of filter, and recovering the signal?

3. What are other options, besides changing the low-pass filter, do you have to overcome the noise? (Hint: The Signal to Noise and Distortion Ratio)

Performance Measures

Task	Standards	Sat/Unsat
Worksheets and Short Answer Questions	Quality of answers and data collected.	
Running VIs	Successful transmission and reception of tones.	
Wiring of terminals	Clarity of Tx and Rx VI layouts.	

Discussion

Did all configurations perform as expected?
Did you have any difficulties completing the lab?
Did your TA provide enough guidance?
Do you have any recommendations to improve the lab?

9.6 References

[1] Lab 2: Amplitude Modulation, Bruce A. Black, Rose-Hulman Institute of Technology, July 2013.
[2] Random Samples: The Central Limit Theorem, "Virtual laboratories in Probability and Statistics", retrieved May 21, 2014, from http://www.math.uah.edu/stat/sample/CLT.html.

10 Frequency Modulation with Additive Gaussian White Noise

10.1 Summary

The objective of this laboratory exercise is to utilize the USRP to modulate an FM signal with additive White Gaussian Noise (AWGN) and demodulate the signal above the threshold operation.

10.2 Background

10.2.1 Noise Performance of Frequency-Modulation Systems

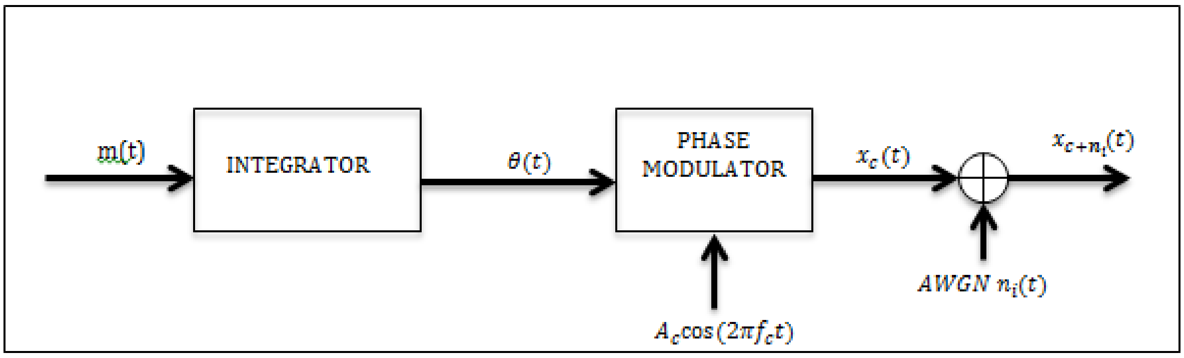


Fig. 142: Transmitter for an FM system with Added White Noise

Noise is present in all electronic systems and affects the quality of the desired signal recovered at the receiver. The "Frequency Modulation" lab explored angle modulation with a focus on frequency modulation, but did not take into account the noise and distortion of the message signal *m(t)* being received. The extent to which noise affects the performance of communication systems is measured by either the output signal-to-noise (SNR) power ratio or the probability of error. The SNR is used to measure the performance of analog communication systems, and the probability of error is used to measure the performance of digital communication systems.

AWGN is a basic noise model used to simulate the effect of many random processes that occur in the transmission of the signal. AWGN is often used as a channel model in which the only impairment to communication is a linear addition of wideband or white noise with a constant spectral density (expressed as watts per hertz of bandwidth) and a Gaussian distribution of amplitude. The purpose of this lab is to introduce noise to FM signals so as to mimic the effect of radiation sources on an FM signal transmission. The standard deviation of AWGM can be estimated using a spectrum analyzer to observe the noise floor, where the noise floor is the measure of the signal created from the sum of all AWGNs within a measurement system.

"Normally, noise is added to the signal as it traverses the transmission medium. In this experiment, however, the Additive White Gaussian Noise (AWGN) will be added at the end of the transmitted (frequency modulated) signal $x_c(t)$ (Fig. 1), where

$$x_c(t) = A_c \cos\,[w_c t + \theta(t)] \tag{64}$$

The excess phase $\theta(t)$ is given by

$$\theta(t) = 2\pi\Delta f_{max} \int_{-\infty}^{t} m_n(\alpha)d\alpha \tag{65}$$

where Δf_{max} is the peak frequency deviation, and $m_n(\alpha)$ is the normalized message signal (66),

$$m_n(t) = \frac{m(t)}{\max\ |m(t)|} \tag{66}$$

Adding the AWGN, represented by $n_i(t)$, to the original message signal produces an output signal $x_{c+n_i}(t)$, which can be written as

$$x_{c+n_i}(t) = A_c \cos[w_c t + \theta(t)] + n_i(\mathrm{t}) \tag{67}$$

The transmitted signal with noise, $x_{c+n_i}(t)$, arrives as a modulated signal with noise, $x_{r+n_i}(t)$, as shown in Fig. 143. The noisy signal is then filtered with a pre-detection filter (a differentiator) to get the differentiator output, $e(t)$, then a discriminator/FM demodulator to get the transient signal $y(t)$, and finally a post-detection filter to get the output of the discriminator $y_D(t)$. The detailed FM demodulation process has already been covered in the frequency modulation lab.

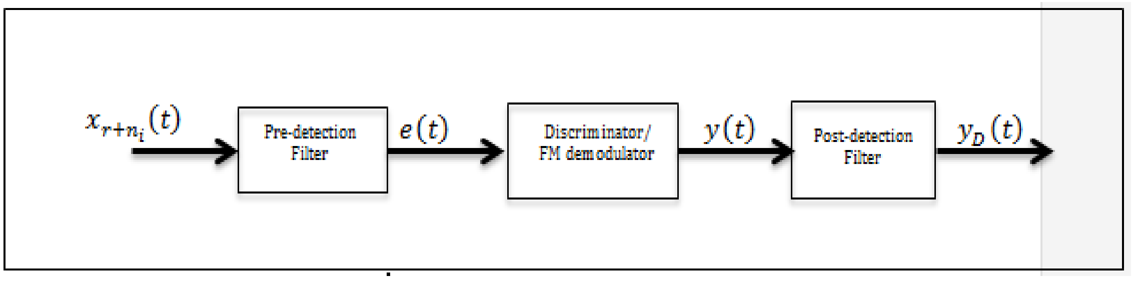


Fig. 143: FM demodulator

10.3 Pre-Lab

The task is for you to complete the transmitter and receiver VIs before coming to the lab.

10.3.1 Transmitter

Your task is to add a data entry pane to the transmitter template (**Fig. 127**) to allow addition of White Gaussian noise to the amplitude modulated signal in FM_Noise_Tx_Template.vi.

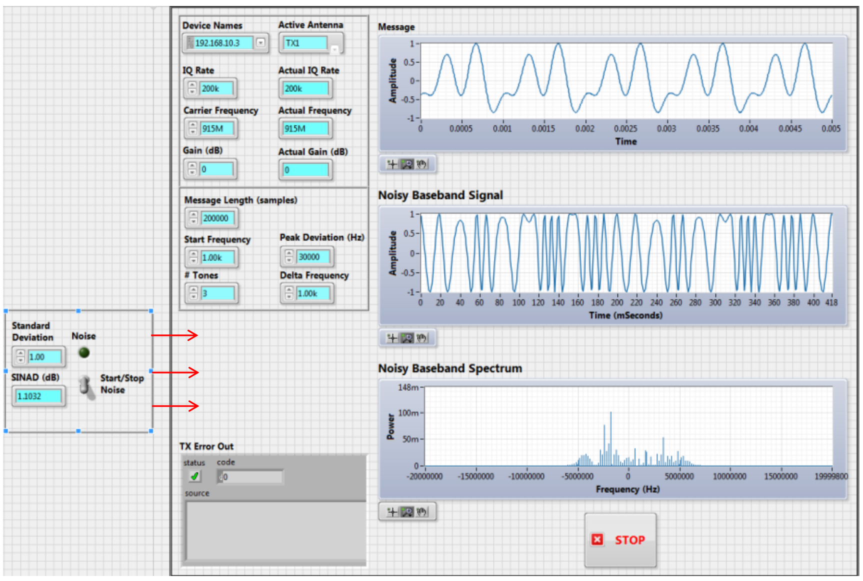
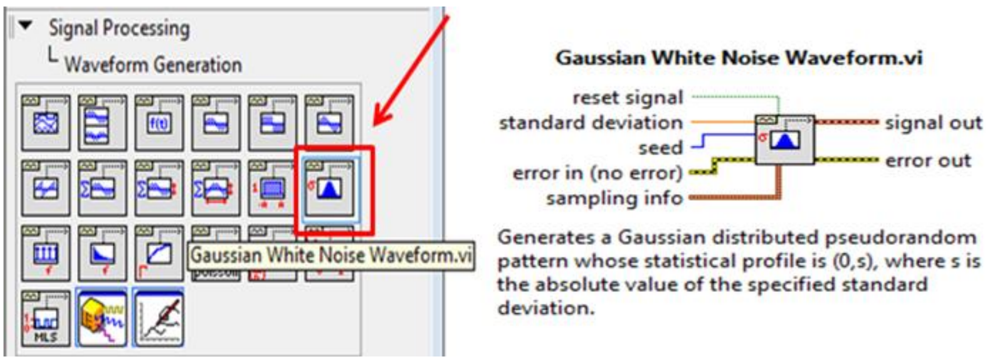

Device Names: 192.168.10.3
Active Antenna: TX1
IQ Rate: 200k
Actual IQ Rate: 200k
Carrier Frequency: 915M
Actual Frequency: 915M
Gain (dB): 0
Actual Gain (dB): 0
Message Length (samples): 200000
Start Frequency: 1.00k
Peak Deviation (Hz): 30000
Tones: 3
Delta Frequency: 1.00k
Standard Deviation: 1.00
SINAD (dB): 1.1032
TX Error Out: status ✓ code 0

Fig. 144: Transmitter VI Template Front Panel

a) Generate a white noise source using the “Gaussian White Noise Waveform” VI (**Fig. 128**). The VI generates a Gaussian distributed pseudorandom pattern with a mean of 0 and a standard deviation (σ), where σ is the absolute value of the chosen standard deviation. The value of the standard deviation input is controlled by a switch and case structure as explained in the coming steps. Set the sampling information for the generated white noise the same as the message signal generated by the basic multi-tone.



Generates a Gaussian distributed pseudorandom pattern whose statistical profile is (0,s), where s is the absolute value of the specified standard deviation.

Fig. 145: Gaussian White Noise Waveform VI

b) Insert a switch and round LED indicator on the front panel to activate or deactivate noise addition from the front panel. So when you want the noise to be added, you would turn the switch to the “On” position and the LED indicator would turn “Green”. These controls can be found in the Front Panel Boolean Controls Palette as shown in **Fig. 129**.

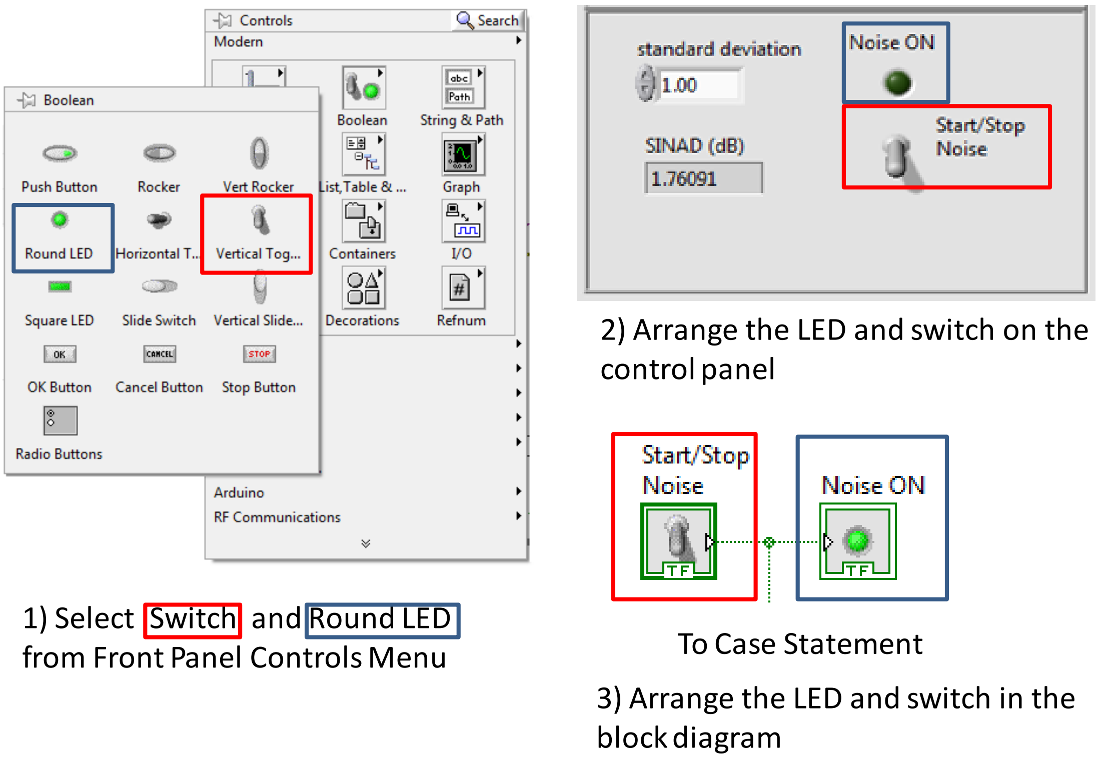
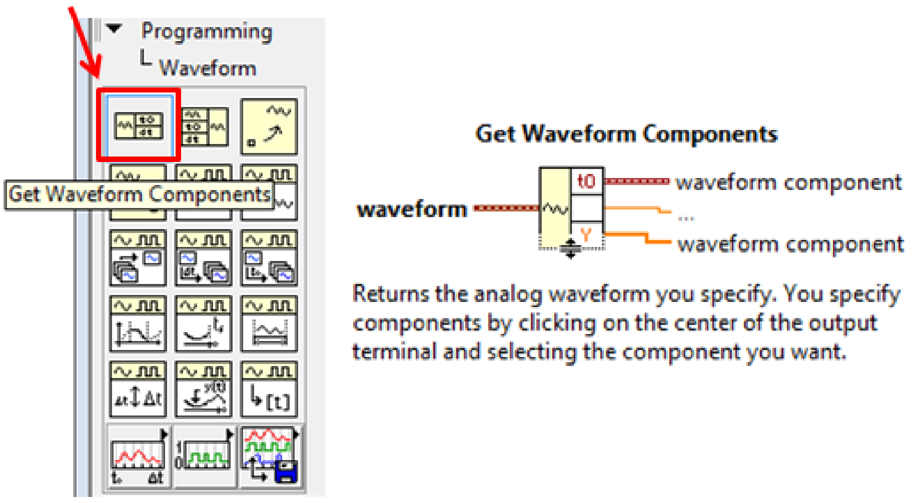


Fig. 146: Boolean Switch and LED Palette Location

c) Design a case structure that relates to the “On” and “Off” positions of the switch. When the switch is “On”, the standard deviation terminal of the Gaussian White Noise Waveform Generator should receive the value of “Standard Deviation” set up from the front panel, and zero otherwise.

d) Retrieve the data values of the generated noise using a “Get Waveform Components” VI and add them to the complex frequency modulated signal.

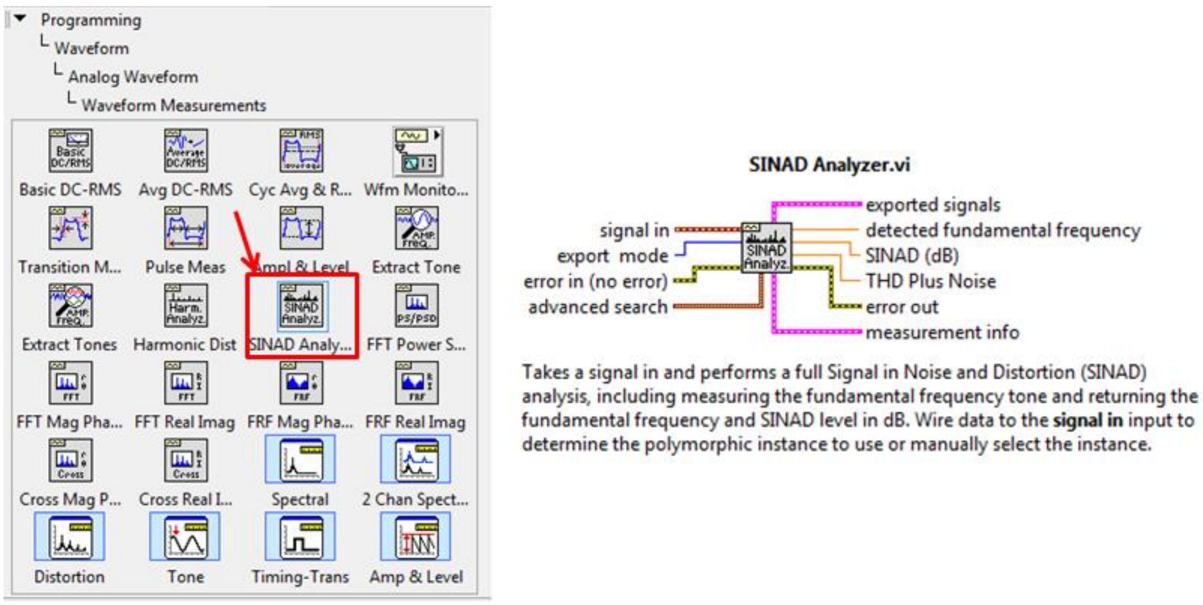


Returns the analog waveform you specify. You specify components by clicking on the center of the output terminal and selecting the component you want.

Fig. 147: Get waveform components VI

e) Pass the noisy FM signal data values to the “data” input of “niUSRP Write Tx Data.vi” for transmission.

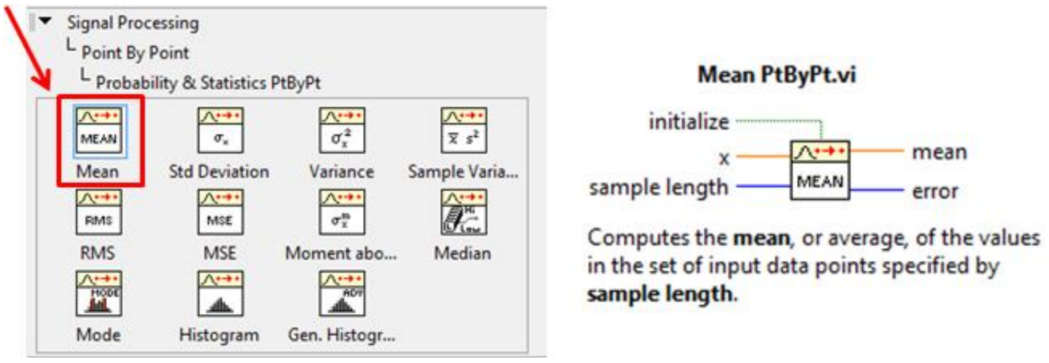
f) Pass the noisy FM signal to the “signal in” input of the “SINAD Analyzer” VI (Fig. 14) to perform a Signal in Noise and Distortion (SINAD) analysis on the analog waveform. Choose “Input signal” as the export mode.



Takes a signal in and performs a full Signal in Noise and Distortion (SINAD) analysis, including measuring the fundamental frequency tone and returning the fundamental frequency and SINAD level in dB. Wire data to the **signal in** input to determine the polymorphic instance to use or manually select the instance.

Fig. 148: SINAD Analyzer.vi

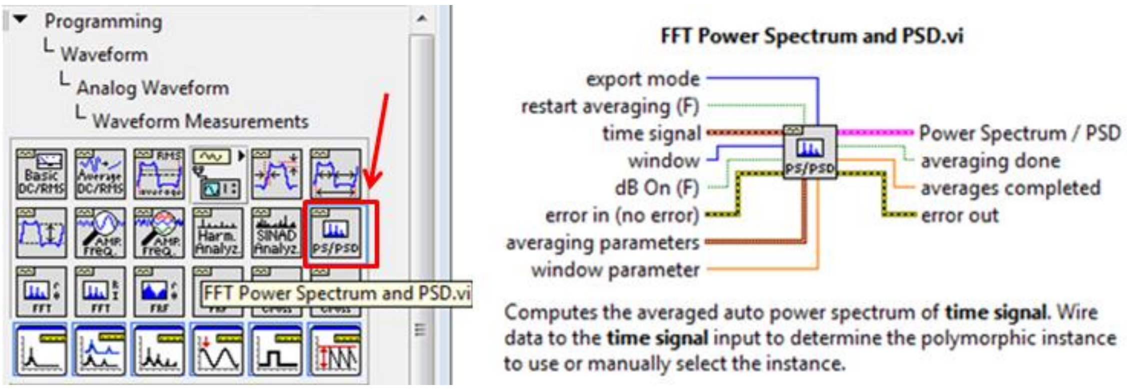
g) Calculate the average SINAD value by taking the mean of the last hundred values returned by the SINAD Analysis using the “Mean PtbyPt” VI (Fig. 149). Display the mean SINAD value by the indicator provided.



Computes the **mean**, or average, of the values in the set of input data points specified by **sample length**.

Fig. 149: Path to Mean PtByPt VI

h) Plot the noisy FM signal in time and frequency domain using waveform graphs. Use “FFT Power Spectrum and PSD” VI (Fig. 150) to get the spectrum of the time domain signal.



Computes the averaged auto power spectrum of **time signal**. Wire data to the **time signal** input to determine the polymorphic instance to use or manually select the instance.

Fig. 150: FFT Power Spectrum and PSD VI

i) Save your transmitter in a file whose name includes the letters “FM_Noise_Tx” and your initials.

10.3.2 Receiver

At the receiver, the FM modulated noisy data is fetched, the phase part is unwrapped and filtered, and lastly, the noise is filtered. The noise is filtered by a digital circuit that allows the selection of no filter, a Chebyshev low-pass filter or a Butterworth low-pass filter to extract the tones in the transmitted signal’s envelope.

Your task is to fetch the data, demodulate it, filter noise from it and then analyze the extracted signal in FM_Noise_Rx.vi. Most of the programming would be in the while-loop, the steps for which are described below.

a) Retrieve the data values and the sampling time interval of the data fetched by “NI USRP Fetch Rx Data” VI by using the “Get Waveform Components” VI (Fig. 147).

b) Extract the angular components by converting the received signal data values from complex form to polar form by using “Complex to Polar” VI (Fig. 151).

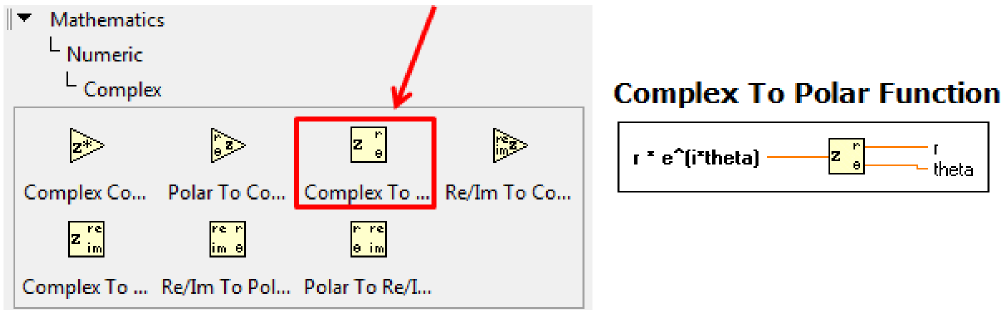


Fig. 151: Complex to Polar VI

c) Unwrap the phase by using the “Unwrap Phase” VI (Fig. 152).

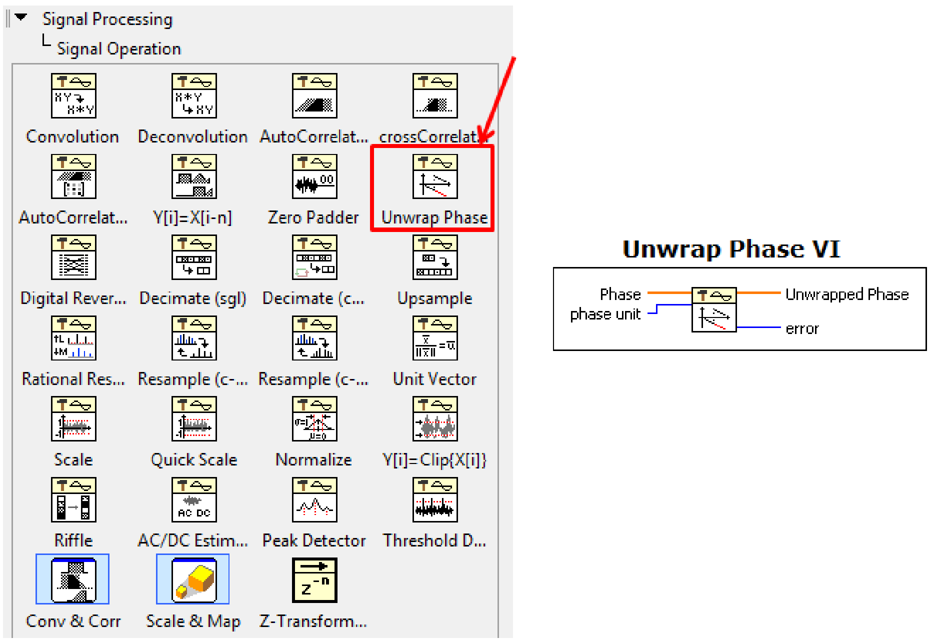
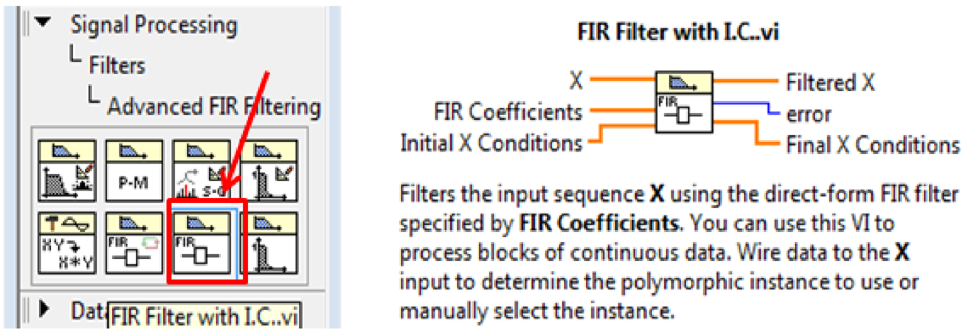


Fig. 152: Unwrap Phase VI

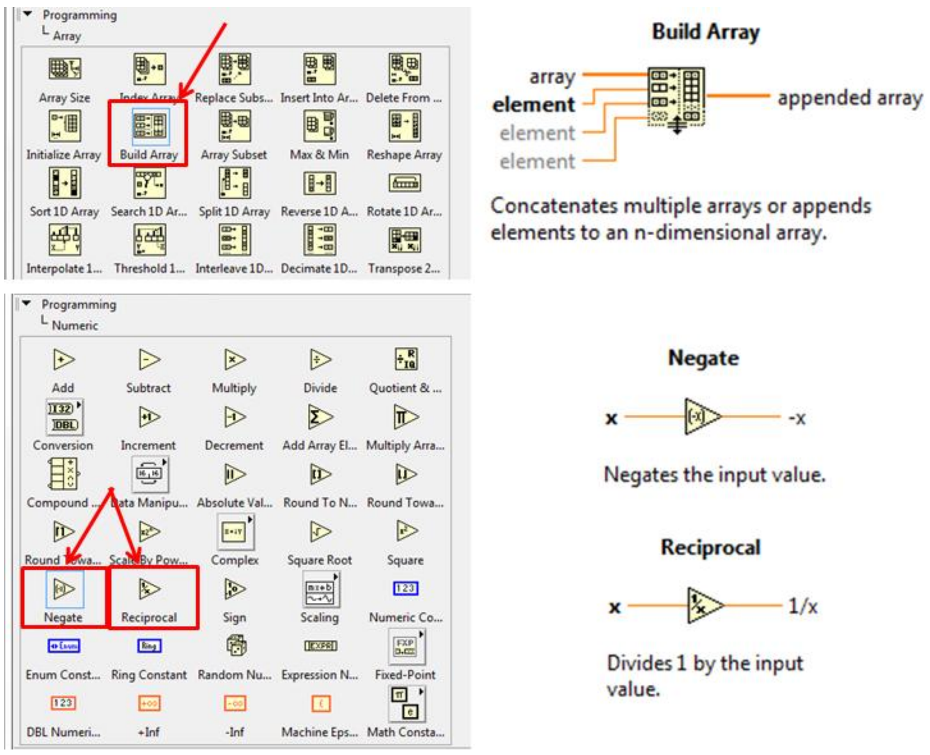
d) Filter the unwrapped phase component using a the “FIR filter with I.C.” VI (Fig. 153). The initial condition is set to zero and the final condition for a given loop iteration is set to be the initial condition for the next loop iteration by using a shift register. The FIR coefficients are calculated from the time interval of the fetched waveform by using “Build Array” VI (Fig. 154) to set up an array that appends the sampling frequency (the reciprocal of sampling time interval) and its negative in all loop iterations (Fig. 154).

Note: The FIR filter here is a differentiator with $\left(\frac{1}{T}, -\frac{1}{T}\right)$ as the “FIR coefficients” array. Please refer to the receiver section in the “Frequency Modulation” lab for details.



Filters the input sequence **X** using the direct-form FIR filter specified by **FIR Coefficients**. You can use this VI to process blocks of continuous data. Wire data to the **X** input to determine the polymorphic instance to use or manually select the instance.

Fig. 153: FIR Filter with I.C. VI



Concatenates multiple arrays or appends elements to an n-dimensional array.

Negates the input value.

Divides 1 by the input value.

Fig. 154: (a) Build Array VI, (b) Negate function, (c) Reciprocal function

e) Input the filtered signal into the envelope detector digital circuit provided to you for noise removal, keeping the sampling frequency the same as before (Fig. 155).

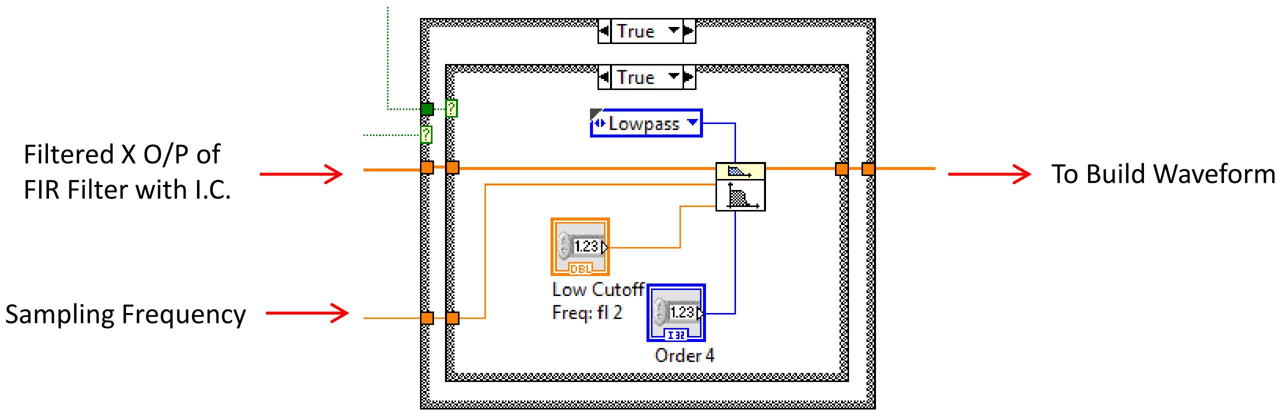
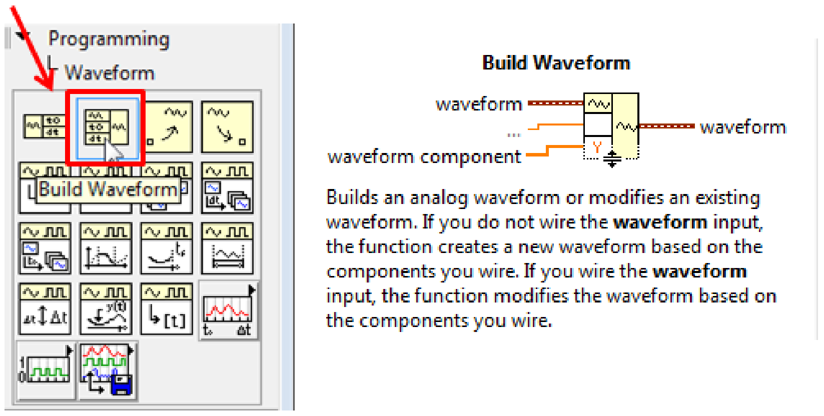


Fig. 155: I/P and sampling freq. connections to inner loop filters

Remember, the logic circuit's outer case structure toggles between turning the low-pass filters on/off, whereas the inner case structure toggles between the Butterworth and Chebychev filters.

f) The output of the envelope detector is the demodulated sequence (discrete data values). Build an analog waveform by bundling these filtered signal coefficients with the sampling time (same as that of the FM signal fetched by the receiver buffer) to get an analog waveform by using the "Build Waveform" VI (Fig. 156).



Builds an analog waveform or modifies an existing waveform. If you do not wire the **waveform** input, the function creates a new waveform based on the components you wire. If you wire the **waveform** input, the function modifies the waveform based on the components you wire.

Fig. 156: Build Waveform VI

g) Plot the noisy FM signal in time and frequency domain using the waveform graphs provided. Use "FFT Power Spectrum and PSD" VI (Fig. 150) to get the spectrum of the time domain signal.

h) Pass the demodulated waveform to the "signal in" input of the "SINAD Analyzer" VI (Fig. 148) to perform a Signal in Noise and Distortion (SINAD) analysis on the analog waveform. Choose "Input signal" as the export mode.

i) Calculate the average SINAD value by taking the mean of the last hundred values returned by the SINAD Analyzer using the "Mean PtbyPt" VI (Fig. 149). Display the mean SINAD value by the indicator provided.

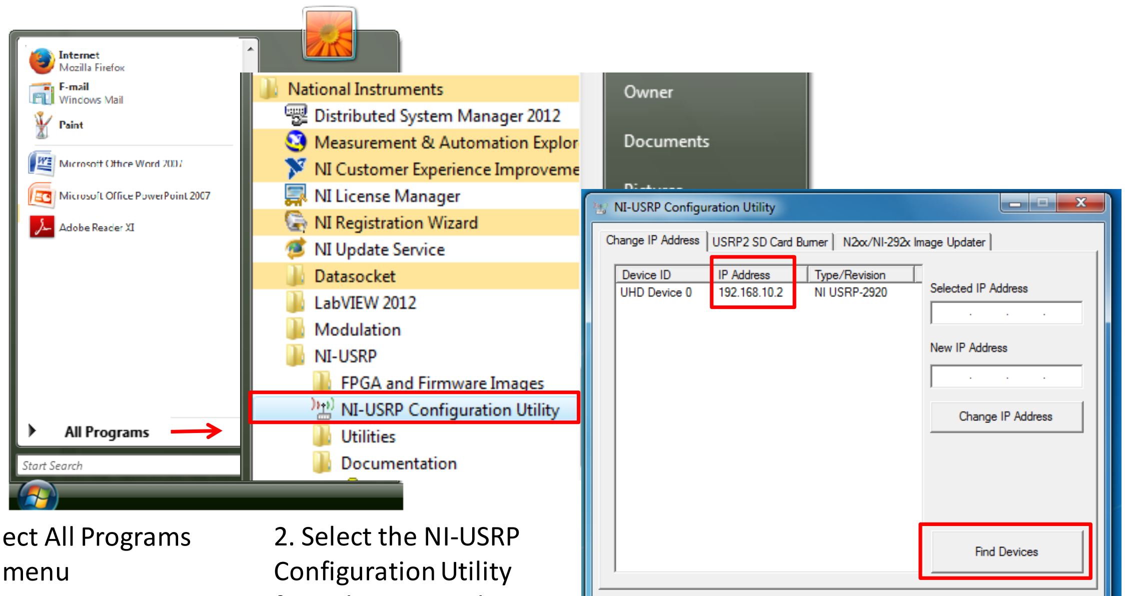
j) Save your receiver in a file whose name includes the letters "FM_Noise_Rx" and your initials.

10.4 Lab Procedure

10.4.1 Testing the Transmitter and Receiver

Test your designed transmitter and receiver VIs for this lab by following the undermentioned steps.

1. Run LabVIEW and open the transmitter and receiver VIs that you created.
2. Connect the computer to the USRP using an Ethernet cable.
3. Open the NI-USRP Configuration Utility found in the National Instruments directory under programs files as shown in Fig. 157. Be sure to record the IP addresses since you will need them to configure your software.



Device ID	IP Address	Type/Revision
UHD Device 0	192.168.10.2	NI USRP-2920

1. Select All Programs from menu

2. Select the NI-USRP Configuration Utility from the National Instruments directory

3. Select Find Devices and record the IP address of the radio or radios since you will need them to configure the software in the lab.

Fig. 157: Finding the IP Address: Radio Connectivity Test

If the IP address does not appear in the window then check your connections and ask the Teaching Assistant (TA) to verify that the LAN card has been configured correctly.

4. Connect a loopback cable between the TX 1 and RX 2 antenna connectors. Remember to connect the attenuator to the receiver end.

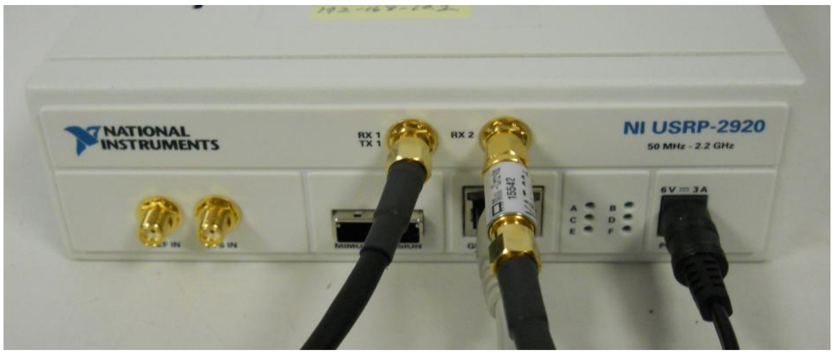


Fig. 158: Broadcast Setup

5. Ensure that the transmitter VI is set up to use the values in Table XXIX

Table XXIX – Transmitter Settings

Field	Setting
Device Name:	192.168.10.x
Carrier Frequency:	915 MHz
IQ Rate:	200 kHz
Gain:	0 dB
Active Antenna:	TX1

Table XXIX – Transmitter Settings

Field	Setting
Number of Samples:	200,000 samples
Standard Deviation	0.1
Peak Frequency Dev.	30 kHz
Start Frequency, Delta Frequency, Number of Tones:	1 kHz, 1 kHz, 3

6. Ensure that the receiver VI is set up to use the values in Table XXX.

Table XXX – Receiver Settings

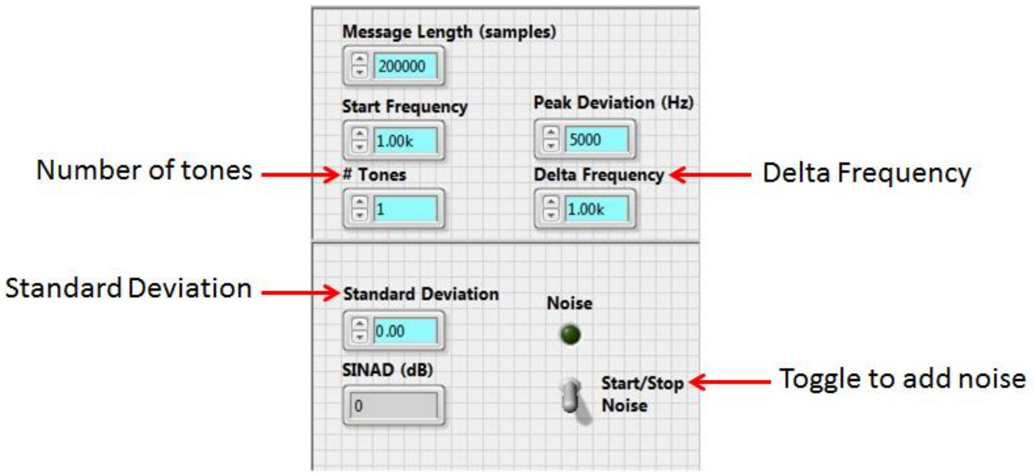
Field	Setting
Device Name:	192.168.10.x
Carrier Frequency:	915 MHz
IQ Rate:	200 kHz
Gain:	0 dB
Active Antenna:	RX2
Number of Samples:	200,000 samples
Filter Order:	5
Filter low cut-off frequency:	5000 Hz

7. Run the receiver (FM_Noise_Rx.vi) VI. LED “C” will illuminate on the USRP if the radio is receiving data.
8. Run the transmitter (FM_Noise_Tx.vi) VI. LED “A” will illuminate on the USRP if the radio is transmitting data.
9. After a few seconds, stop the receiver using the STOP button on the receiver VI, and then stop the transmitter using the STOP button on the transmitter VI. Use the large STOP button on the front panel of the VI to stop transmission/reception; otherwise the USRP may not be stopped cleanly.
10. Use the horizontal zoom feature on the graph palette to expand the message waveform in the transmitter VI and the demodulated message waveform in the receiver VI. Both waveforms should be identical. If not, review the steps to do in the transmitter and receiver setup. If they seem identical, compare and analyze the plotted spectra.
11. Once you get identical waveforms, change the number of tones, start frequency, delta frequency, and peak frequency deviation to different values and observe changes in the waveforms.

10.4.2 Worksheet: The Effect of Varying the Noise Level

Transmitter Setup:

1. Set the transmitter to use either two or three tones, the start frequency to 1 kHz, and the change in frequency tone (Delta frequency) to 1 kHz.
2. Set the standard deviation in FM_Noise_Tx VI to the first value in Table XXXI.



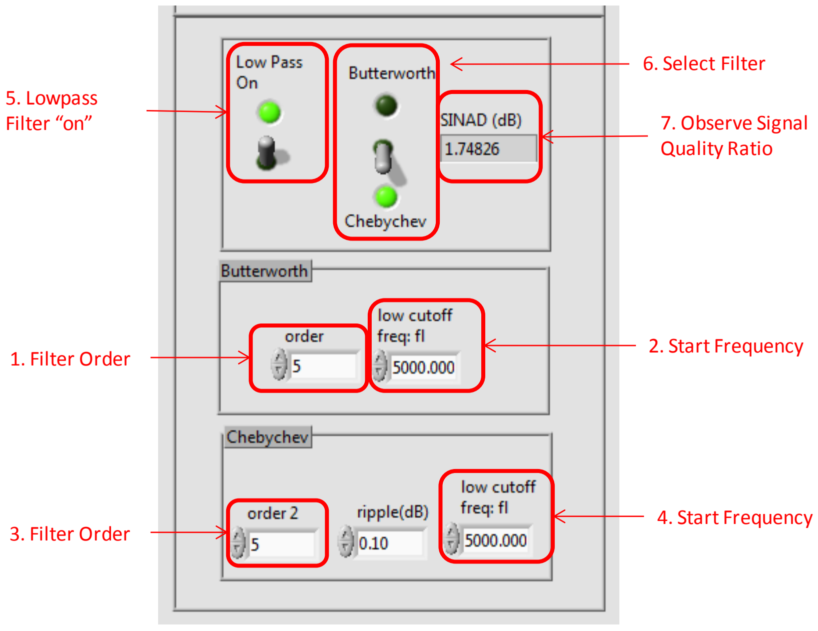
Message Length (samples): 200000
Start Frequency: 1.00k
Peak Deviation (Hz): 5000
Tones: 1
Delta Frequency: 1.00k
Standard Deviation: 0.00
SINAD (dB): 0

Fig. 159: TX Front Panel Noise Configuration Setup

3. Set the transmitter's Start/Stop Noise switch to the "On" position.
4. Do not start the transmitter VI until the receiver setup has been completed.

Receiver Setup:

1. Set the order of the Butterworth filter in Lab_10_FMRx.vi to 5.
2. Set the low-pass cut-off frequency to 5000Hz.
3. Set the order of the Chebyshev filter to 5.
4. Check the ripple is set to 0.1 Hz.
5. Set the low-pass frequency cutoff to 5000Hz.



SINAD (dB): 1.74826
Butterworth: order 5, low cutoff freq: fl 5000.000
Chebychev: order 2 5, ripple(dB) 0.10, low cutoff freq: fl 5000.000

Fig. 160: RX Front Panel Filter Configuration

6. Start the receiver VI, and then start the transmitter VI.
7. Set the low-pass filter switch to the "On" position.
8. Set the filter selection switch to the "Chebyshev" position.
9. Observe the SINAD value and record it in Table XXXI.
10. Observe the received waveform and its spectrum, and record whether the received wave form is recovered or not (Table XXXI). (Use a scale from 1 to 5, where 1 indicates that the waveform is entirely obscured by the noise, and 5 indicates that the waveform looks like the transmitted waveform).
11. Observe the noise floor and record it in Table XXXI.

Making the Observations

12. After each set of observations, reset the noise standard deviation to the next value in the table and record the measurements as described in steps 13 through 15.
13. After filling in with the data obtained using the Chebyshev filter, set the "Filter Selection" switch to the "Butterworth" position and perform the experiment again, recording your observations in

Table XXXI - Effect of Varying the Standard Deviation of Noise (Chebyshev Filter)

Standard Deviation	SINAD	Signal Quality (1 to 5)	Noise Floor
0.0			
0.2			
0.4			
0.6			
0.8			
1.0			

Table XXXII - Effect of Varying the Standard Deviation of Noise (Butterworth Filter)

Standard Deviation	SINAD	Signal Quality (1 to 5)	Noise Floor
0.0			
0.2			
0.4			
0.6			
0.8			
1.0			

10.4.3 Worksheet: The effect of low-pass filter order on envelope detection

Transmitter Setup:
1. Set the transmitter to use either two or three tones.
2. Change the standard deviation to 0.5."
3. Set the transmitter's Start/Stop Noise switch to the "On" position.

Receiver Setup:
1. Set the order of the Butterworth filter to the first value in Table XXXIII.
2. Set the low-pass cut-off frequency to 5000Hz.
3. Set the order of the Chebyshev filter to the first value in Table XXXIII.
4. Check the ripple is set to 0.1 Hz.
5. Set the low-pass cut-off frequency to 5000Hz.
6. Start the receiver VI and then start the transmitter VI.
7. Set the low-pass filter switch to the "On"position.
8. Set the filter selection switch to the "Chebyshev" position.
9. Observe the SINAD value and record it in Table XXXIII.
10. Observe the received waveform and its spectrum and record whether the received wave form is recovered or not. Use a scale from 1 to 5, where 1 indicates that the waveform is entirely obscured by the noise, and 5 indicates that the waveform looks like the transmitted waveform.
11. Set the filter selection switch to the "Butterworth" position.
12. Observe the SINAD value and record it in Table XXXIII.
13. Observe the received waveform and the spectrum and record whether the received waveform is recovered or not, using the scale from step 10.

Making the Observations

14. After each set of observations reset the Filter order to the next value in the table and record the measurements as described I steps 9 through 13.

Table XXXIII - Worksheet for Studying Effect of Varying the Filter Order On SINAD

Filter Order	Chebyshev SINAD	Chebyshev Quality (1 to 5)	Butterworth SINAD	Butterworth Quality (1 to 5)
2				
5				
10				
15				
20				
25				
50				

10.5 Lab Write-up

Performance checklist
Frequency Modulation with noise

Short Answer Questions

1. In comparison to Amplitude Modulation with AWGN, is Frequency Modulation with AWGN better for noise cancelation? Justify your answer.

2. What are other options besides changing the low-pass filter order to overcome the noise? (Hint: Signal to noise ratio)

Performance Measures

Task	Standards	Sat/Unsat
Worksheets and Short Answer Questions	Quality of answers and data collected.	
Running VIs	Successful transmission and reception of tones.	
Wiring of terminals	Clarity of Tx and Rx VI layouts.	

Discussion
Did all configurations perform as expected?
Did you have any difficulties completing the lab?
Did your TA provide enough guidance?
Do you have any recommendations to improve the lab?

10.6 References

[1] R.E. Ziemer and W.H. Tranter, Chapter 5: Random Signals and Noise, In “Principles of Communications: Systems, Modulation and Noise”, Fifth Edition.
[2] M.F. Mesiya, Chapter 5: Angle Modulation, In “Contemporary Communication Systems”, 2013 Edition.
[3] R.E. Ziemer and W.H. Tranter, Chapter 6: Noise in Modulation Systems, In “Principles of Communications: Systems, Modulation and Noise”, Fifth Edition.

11 Frequency Domain Multiplexing

11.1 Summary

In this laboratory exercise you will investigate sending multiple messages on a single carrier by frequency-division multiplexing [1]. Each individual message is modulated onto a separate sub-carrier and the modulated sub-carriers are summed before being sent to the USRP. At the receiver, the individual messages are separated by filtering and then demodulated. The purpose of this exercise is to

- provide additional practice in programming the USRP,
- introduce the concept of frequency-division multiplexing, and
- explore the concept of intermediate-frequency filtering in the receiver.

11.2 Background

11.2.1 Frequency-Division Multiplexing Basics

Frequency-division multiplexing is widely used in telemetry, in the satellite relaying of television signals, and, until the widespread adoption of fiber optics, was the standard transmission method for long-distance telephone signals. Frequency-division multiplexing also plays an important role in the OFDM technique used in DSL and in fourth-generation cellular telephone systems.

If $m_i(t)$ are the complex QAM symbol, N_s is the number of subcarriers, T the symbol duration, and $f_i = f_0 + i/T$ the carrier frequency, then one OFDM symbol starting at $t = t_s$ can be written as:

$$s(t) = \begin{cases} RE\left\{\sum_{i=0}^{N_s-1} m_i(t)\, e^{j2\pi f_i(t-t_s)}\right\}, & t_s \le t \le t_s + T \\ 0, & \text{otherwise} \end{cases} \tag{68}$$

Often the equivalent complex notation is used, which is given by (69). In this representation, the real and imaginary parts correspond to the in-phase and quadrature parts of the OFDM signal, which have to be multiplied by a cosine and sine of the desired carrier frequency to produce the final OFDM signal.

$$s(t) = \begin{cases} m_i(t)\left[\cos\left(2\pi f_i(t-t_s)\right) + j\sin\left(2\pi f_i(t-t_s)\right)\right], & t_s \le t \le t_s + T \\ 0, & \text{otherwise} \end{cases} \tag{69}$$

Fig. 9 shows the operation of the OFDM modular in block diagram.

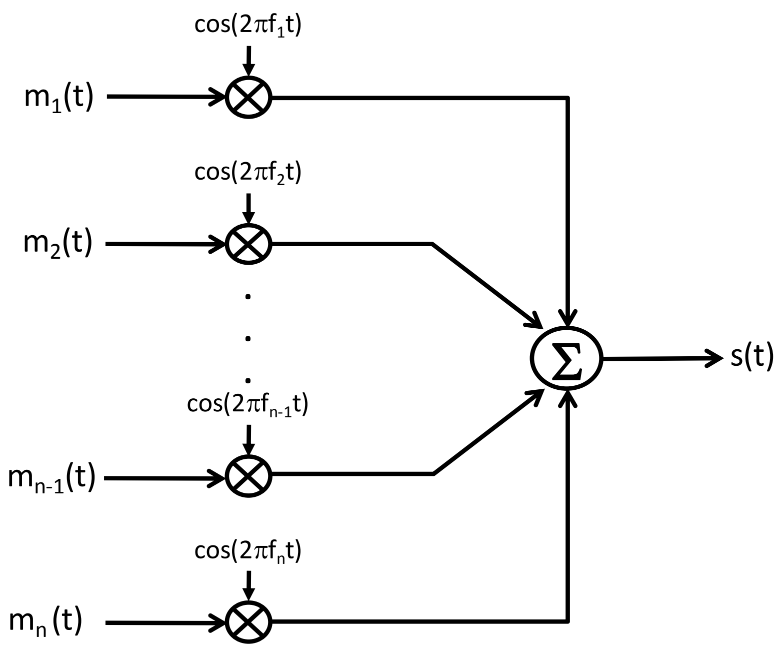


Fig. 161. Typical OFDM Modulation Schema

As an example, Fig. 162 shows four sub-carriers from one OFDM signal. In this example, all sub-carriers have the same phase and amplitude, but in practice the amplitudes and phases may be modulated differently for each sub-carrier. Note that each sub-carrier has exactly an integer number of cycles in the interval T, and the number of cycles between adjacent sub-carriers differs by exactly one. This properly accounts for the orthogonality between sub-carriers.

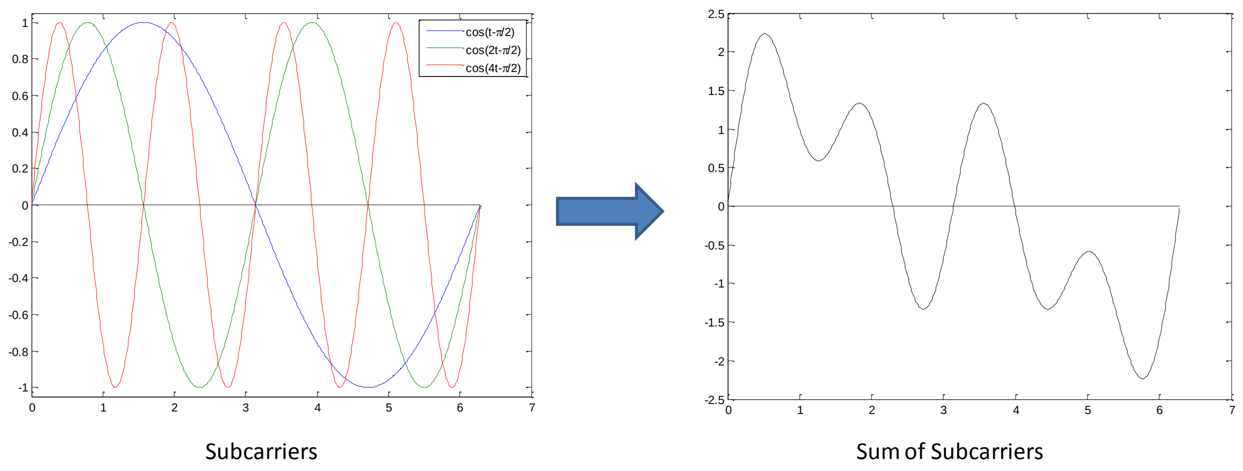


Fig. 162. OFDM Sub-carriers[2]

For instance, if the k^{th} sub-carrier from (69) is demodulated by down converting the signal with a frequency of $f_k = f_0 + k/T$ and then integrating the signal over T seconds, the result is as written in (3). By looking at the intermediate result, it can be seen that a complex carrier is integrated over T seconds. For the demodulated sub-carrier k, this integration gives the desired output $m_k(t)$ (multiplied by a constant factor T), which is the QAM value for that particular sub-carriers. For all other sub-carriers, this integration is zero, because the frequency difference $(i-k)/T$ produce an integer number of cycles within the integration interval T, such that the integration result is always zero.

In this lab we will suppose $m_1(t)$ and $m_2(t)$ are message signals. Let f_1 and f_2 be corresponding sub-carrier frequencies, $m_{i,p}$ be the maximum value of the signals, and μ_i be their respective modulation indexes. This gives us the modulated sub-carrier signals

$$\begin{aligned} g_1(t) &= A_1\left[1+\mu_1\frac{m_1}{m_{1,p}}\right]\cos(2\pi \mathrm{f}_1 \mathrm{t}) \\ g_2(t) &= A_2\left[1+\mu_2\frac{m_2}{m_{2,p}}\right]\cos(2\pi \mathrm{f}_2 \mathrm{t}) \end{aligned} \tag{70}$$

It is not necessary to use amplitude modulation to modulate the sub-carriers. We are using AM in this lab exercise because it is familiar from the Amplitude Modulation lab, and because it is easy to demodulate. Note that the sub-carrier frequencies f_1 and f_2 must be spaced sufficiently apart in frequency so that the spectra of $g_1(t)$ and $g_2(t)$ do not overlap. The signals $g_1(t)$ and $g_2(t)$ are combined using equation (71). The resulting term $g_I(t)$ is the in-phase component of the baseband signal.

$$g_I(t) = g_1(t) + g_2(t) \tag{71}$$

As in Lab 2 we will let the quadrature component, $g_Q(t)$ equal zero. So that the signal sent to the USRP is

$$\tilde{g}(t) = g_I(t) + j\, g_Q(t) = g_I(t) = g_1(t) + g_2(t) \tag{72}$$

The signal actually transmitted by the USRP is

$$\begin{aligned} g(t) &= A g_I(t)\cos(2\pi f_c t) - j\, g_Q(t)\sin(2\pi f_c t) \\ &= A[g_1(t) + g_2(t)]\cos(2\pi f_c t) \end{aligned} \tag{73}$$

where A is set by the "gain" parameter and $f_c$is the USRP carrier frequency.
On the receiving side, the USRP receiver provides the output

$$\tilde{r}(t) = A_r[g_1(t) + g_2(t)]e^{j\theta} \tag{74}$$

where θ is the phase difference between the transmitter and receiver oscillator signals. This complex-valued signal can be sent to a bank of two bandpass filters centered on the sub-carrier frequencies f_1 and f_2. The filter outputs are the individual signals given in (70). These can now be demodulated using envelope detectors as in the Amplitude Modulation lab.

11.3 Pre-Lab

The task is to add blocks as needed to produce an OFDM signal composed of two AM messages, and then to pass the OFDM signal into the *while* loop to the Write Tx Data block.
A template for the transmitter has been provided in the file FDM_Tx_Template.vi. To complete the transmitter you will be asked to perform two tasks:

- Create a sub-vi that modulates a message using Amplitude Modulation.
- Update the transmitter template to combine the modulated messages to form the OFDM signal.

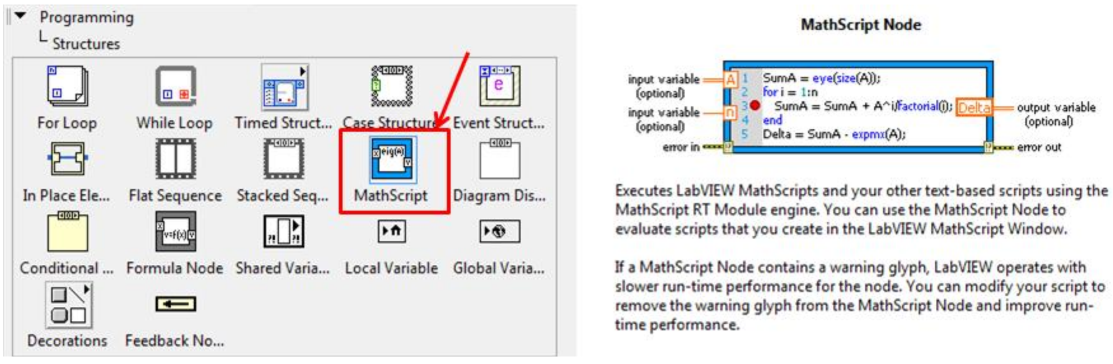
A template for the receiver is also provided, FDM_Rx_Template.vi. To complete the lab, you will need to

- Design a band pass filter to isolate each message signal.
- Create an envelope detector similar to the one designed in Amplitude Modulation Lab.

Transmitter Tasks

Creating "AM_on_Sub-carrier" sub-VI:

a) Create the signals $g_1(t)$ or $g_2(t)$ by using a MathScript Node **()**.



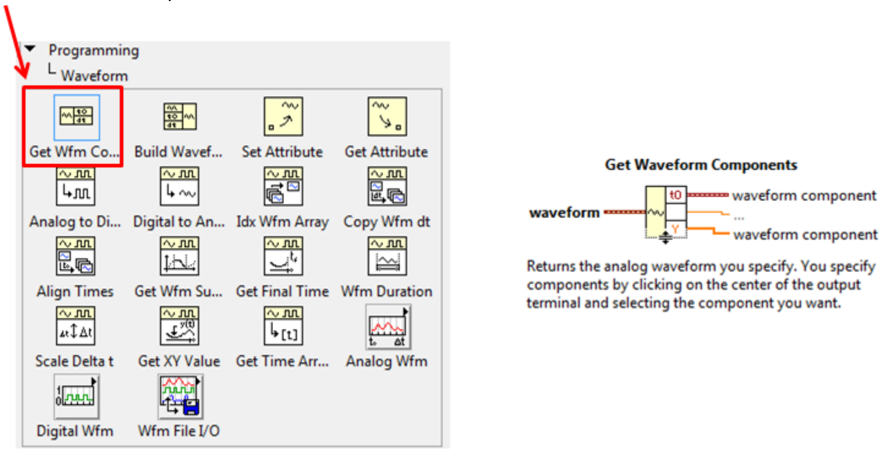
```
SumA = eye(size(A));
for i = 1:n
    SumA = SumA + A^i/factorial(i);
end
Delta = SumA - expm(A);
```

Executes LabVIEW MathScripts and your other text-based scripts using the MathScript RT Module engine. You can use the MathScript Node to evaluate scripts that you create in the LabVIEW MathScript Window.

If a MathScript Node contains a warning glyph, LabVIEW operates with slower run-time performance for the node. You can modify your script to remove the warning glyph from the MathScript Node and improve run-time performance.

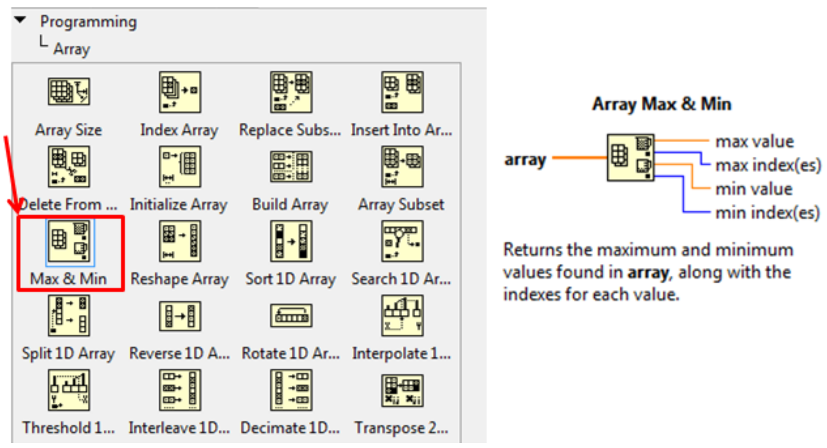
Fig. 163. MathScript Node

b) The MathScript Node would use the message waveform data values, the maximum of the message waveform data values, modulation index, and carrier level as inputs to output the baseband signal. Use "Get waveform components" VI (Fig. 164) to get the data values of the message waveform, the "Max and Min Array" VI (Fig. 165) to get the maximum value of the input signal data values, and indicators for modulation index and carrier level.



Returns the analog waveform you specify. You specify components by clicking on the center of the output terminal and selecting the component you want.

Fig. 164. Get waveform components VI



Returns the maximum and minimum values found in **array**, along with the indexes for each value.

Fig. 165. Max and Min Array VI

c) Next, generate the cosine waveform using the "Sine Waveform" VI **(Fig. 166)** with a 90 degree phase shift. Get the data values of the generated waveform by using "Get waveform components" VI **(Fig. 164)**. Modulate the cosine waveform data values with the baseband signal (output of MathScript Node) to get the modulated signal. Use indicators to output the modulated signal.

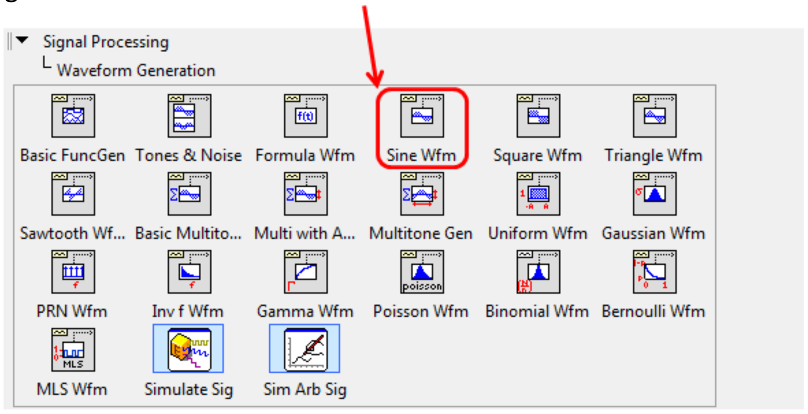


Fig. 166. Sine Waveform VI

The required inputs and outputs for this sub-VI are given in Table XXXIV.

Table XXXIV – AM_on_Subcarrier VI Inputs and Outputs

Inputs	**Function**	**Data Type**
Message waveform in	$m_i(t)$	waveform (double)
Modulation index	μ_i	double
Carrier level	A_i	double
Sampling information for subcarrier	NA	cluster
Subcarrier frequency	f_i	double
Outputs		
Modulated signal out	$g_i(t)$	array (double)

Form the OFDM signal.
A template for the transmitter has been provided in the file FDM_Tx_Template.vi. This template contains the four VIs for interfacing with the USRP along with two message generators that will generate waveforms $m_1(t)$ and $m_2(t)$.

a) Generate the two message waveforms by setting up two instances of the "Basic Multi-tone" VI **(Fig. 167)**. Set the "Number of tones", "Start frequency" and "Delta frequency" as controls for values to be inputted from the front panel. Set the "Coerce frequencies" input to be "True" and the "Sampling information" same as the sampling information cluster provided for both instances.

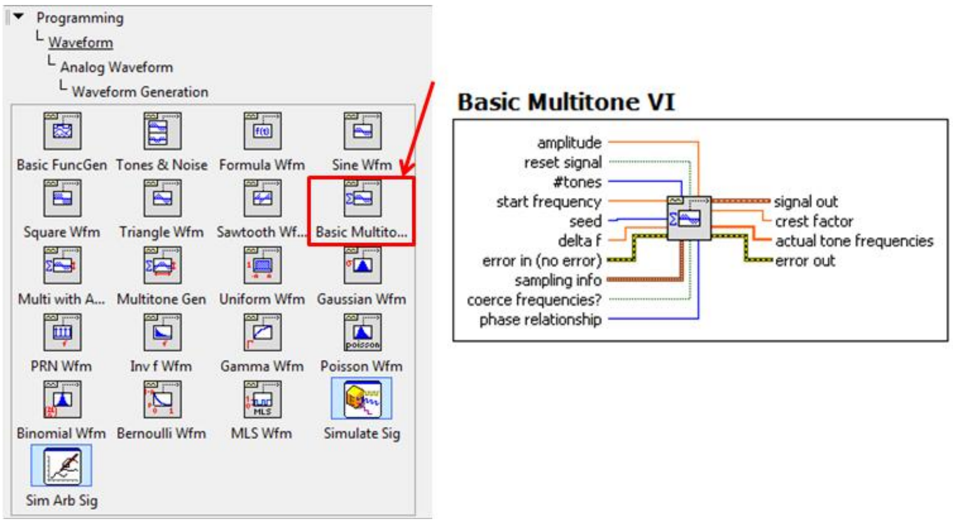


Fig. 167. Basic Multitone VI

b) Build an array from the two generated messages using the “Build Array” VI **(Fig. 168)**, and display the appended message waveform by a waveform graph.

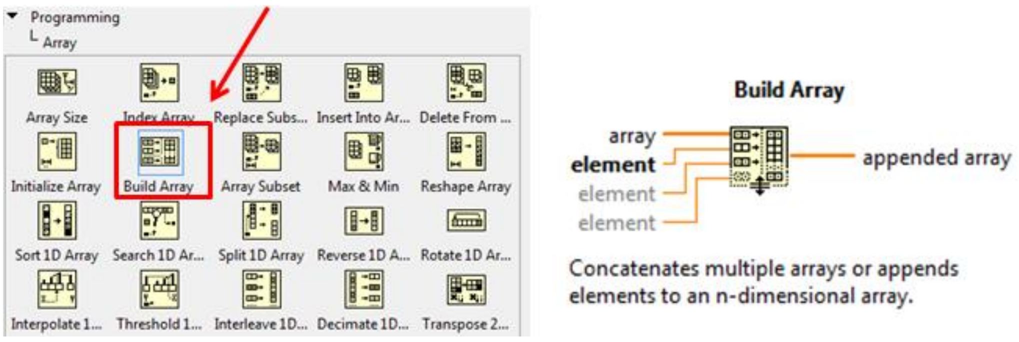


Fig. 168. Build Array VI

c) Feed the generated messages to the two instances of the “AM_on_Sub-carrier" sub-VI you created to create the amplitude modulated signals. Set the “Modulation index” and “Sub-carrier frequency” for both instances of the AM_on_Sub-carrier sub-VI as controls from the front panel. Set the “Carrier level” to be “one”, and the “Sampling information” same as the sampling information cluster provided for both the instances.

d) Add both amplitude modulated signals. Next, scale the sum by using “Quick Scale” VI **(**Fig. 169**)**. Display the scaled AM signal by using the waveform graph provided.

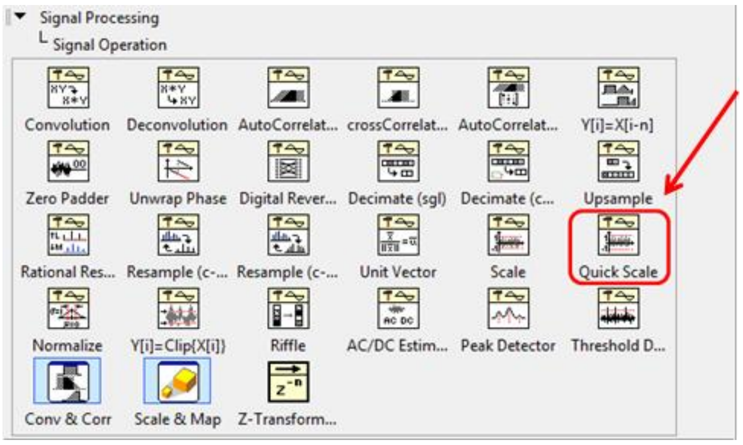


Fig. 169. Quick Scale VI

e) Convert the scaled AM signal to complex form by using “Re/Im to Complex” VI **(Fig. 170)**. For the imaginary input, create an array having the same length as that of the AM signal and all values set to zero by using the “Initialize Array” VI **(Fig. 171)**. Use “Array Size” VI **(Fig. 171)** to find the size of the AM signal.

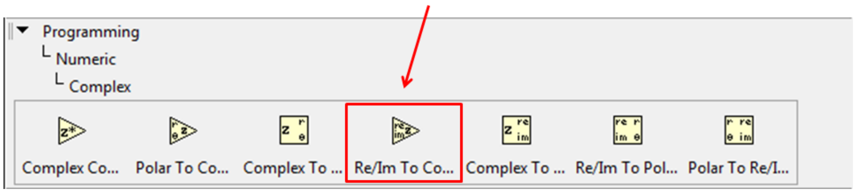


Fig. 170. Re/Im to Complex VI

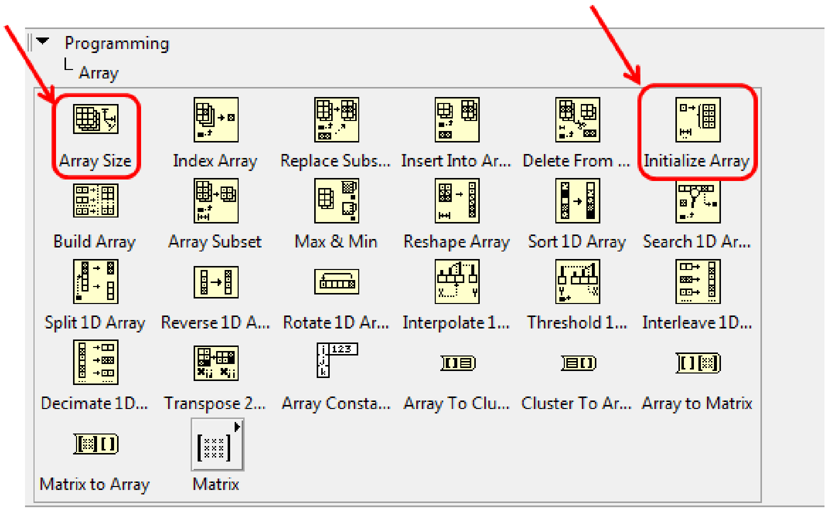


Fig. 171. Array Size and Initialize Array VIs

f) Use “Build Waveform” VI **(Fig. 172)** to build a waveform from the complex data values by setting the sampling time equal to the reciprocal of the “Coerced IQ rate” by using the “Reciprocal” VI **(Fig. 173)**, and pass it onto the buffer for transmission.

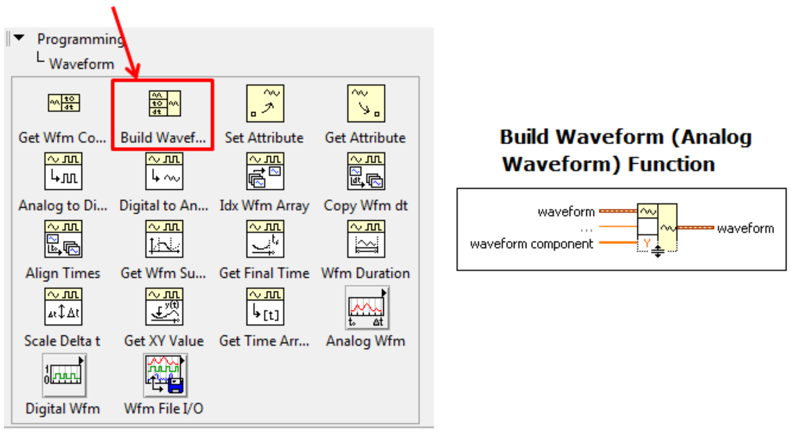


Fig. 172. Build Waveform VI

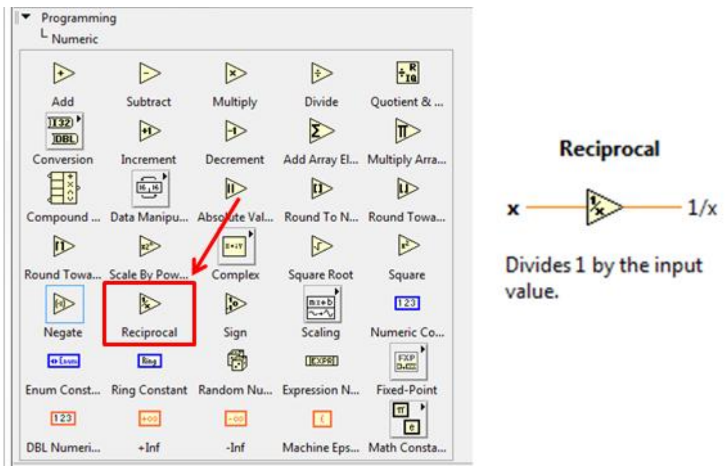


Fig. 173. Reciprocal VI

g) Save your transmitter in a file whose name includes the letters “FDM_Tx” and your initials.

Receiver Tasks

Design a band pass filter

A template for the receiver has been provided in the file FDM_Rx_Template.vi. This template contains the six VIs for interfacing with the USRP along with two waveform graphs on which to display your demodulated output signals.

a. Get the data values of the waveform received by the buffer by using “Get Waveform Components” VI **(Fig. 174).**

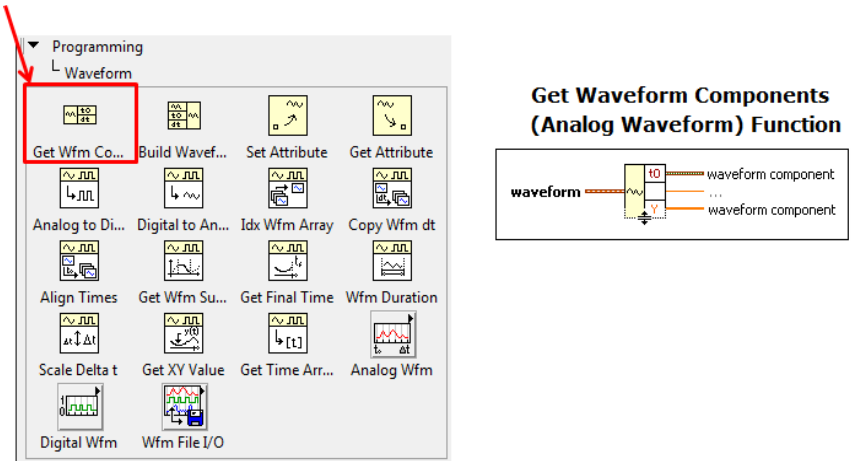


Fig. 174. Get Waveform Components VI

b. Feed the received array into two “Chebyshev” band-pass filters **(Fig. 175)**. Set one of the filters to have a “high cutoff frequency” of 505 kHz and a “low cutoff frequency” of 495 kHz. Set the other filter to have a “high cutoff frequency” of 515 kHz and a “low cutoff frequency” of 505 kHz. Set the filter order to 5 and sampling frequency input as the reciprocal of the sampling time interval using the “Reciprocal” VI **(Fig. 173)** for both filters. The default pass-band ripple value of 0.1 dB is acceptable for both instances of the filter.

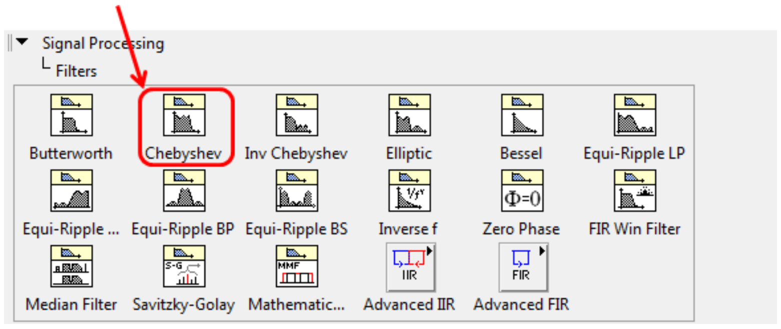


Fig. 175. Chebyshev VI

c. Pass the output of each band-pass filter through an envelope detector. Each envelope detector consists of an absolute value VI (Fig. 176) followed by a “Chebyshev” low-pass filter **(**Fig. 175**)**. Set up the low-pass filters with low-cut-off frequency of 5 kHz and filter order of 5. Set the sampling frequency input as the reciprocal of the sampling time interval using the reciprocal VI **(**Fig. 173) for both filters.

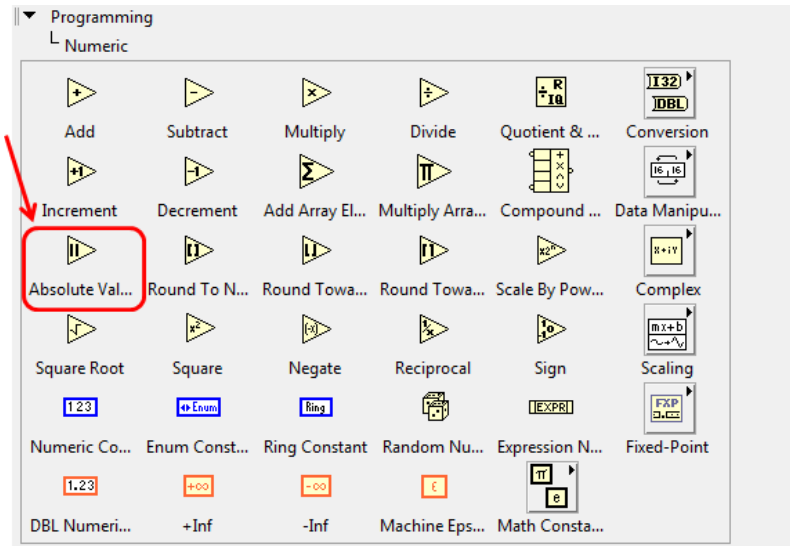


Fig. 176. Absolute Value VI

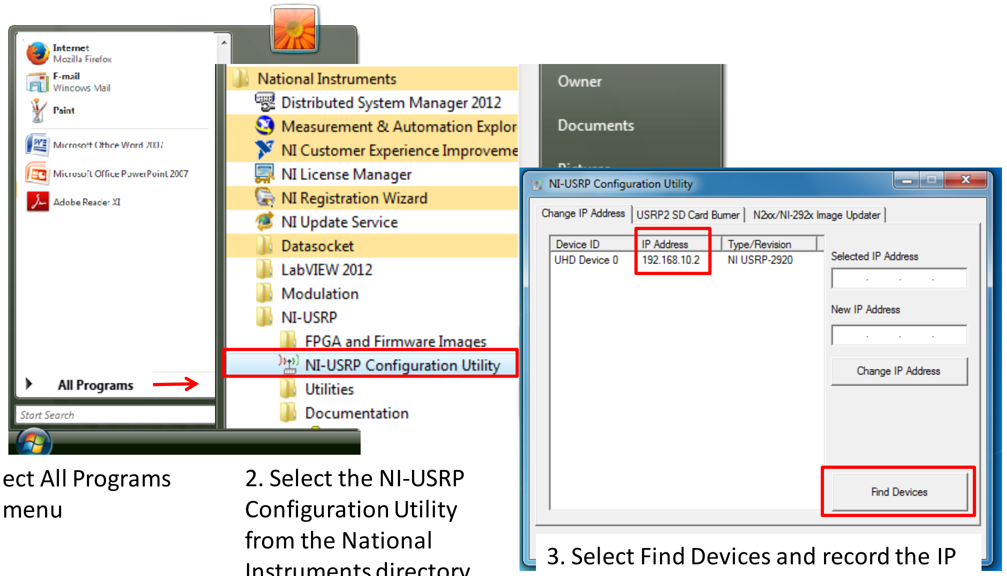
d. Build the two demodulated message waveforms from the low-pass filter outputs by setting the sampling time by using the Build Waveform VI **(Fig. 172)**.
e. Plot the demodulated waveforms by using waveform graphs provided.
f. Save your receiver in a file whose name includes the letters "FDM_Rx" and your initials.

11.4 Lab Procedure

11.4.1 Testing the Transmitter and Receiver

Test your designed transmitter and receiver VIs for this lab by following the undermentioned steps.

1. Run LabVIEW and open the transmitter and receiver VIs that you created.
2. Connect the computer to the USRP using an Ethernet cable.
3. Open the NI-USRP Configuration Utility found in the National Instruments directory under programs files as shown in **Fig. 2**. Be sure to record the IP addresses since you will need them to configure your software.



1. Select All Programs from menu

2. Select the NI-USRP Configuration Utility from the National Instruments directory

3. Select Find Devices and record the IP address of the radio or radios since you will need them to configure the software in the lab.

Fig. 177. Finding the IP Address: Radio Connectivity Test

If the IP address does not appear in the window then check your connections and ask the Teaching Assistant (TA) to verify that the LAN card has been configured correctly.

4. Connect a loopback cable between the TX 1 and RX 2 antenna connectors. Remember to connect the attenuator to the receiver end.

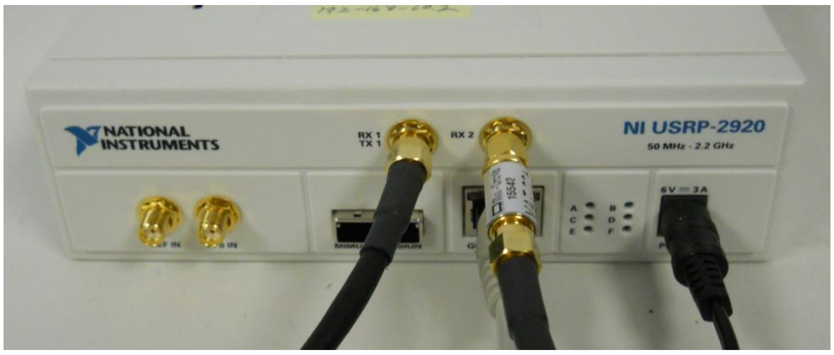


Fig. 178. Broadcast Setup

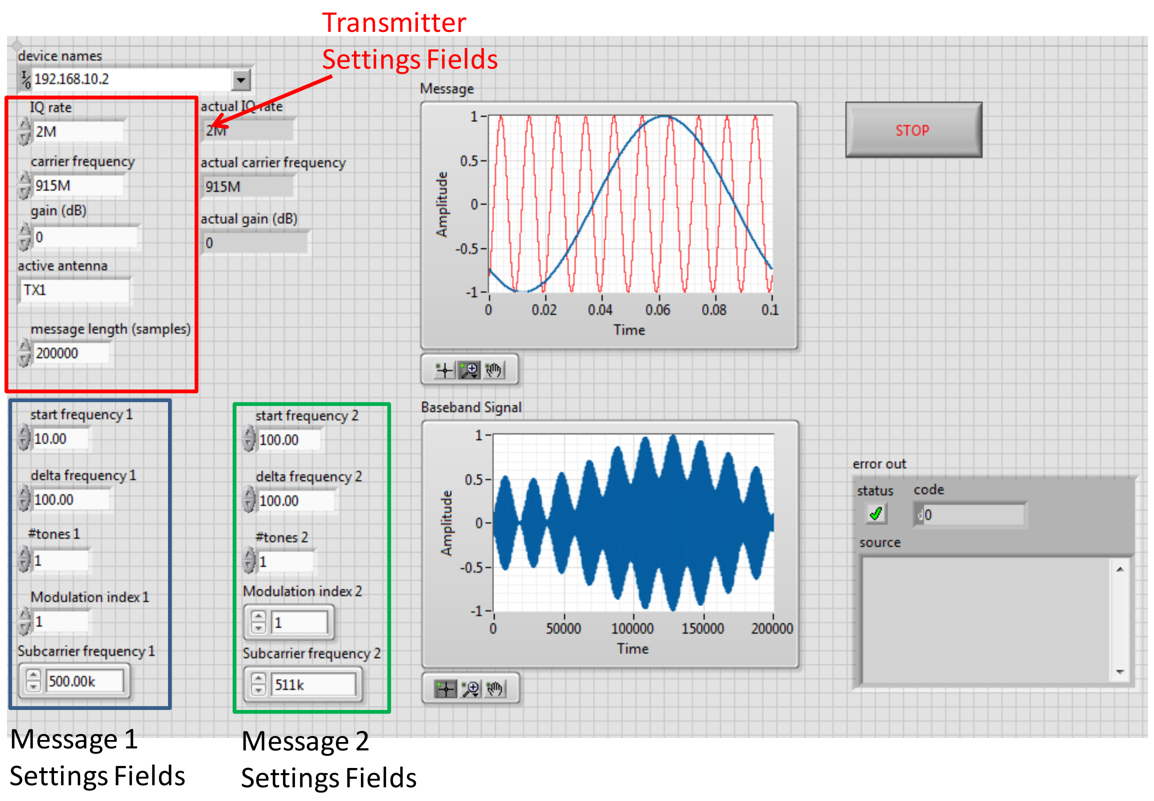
5. Ensure that the transmitter VI is set up to use the settings in Table II. For the messages, use parameters in Table XI.

Table XXXV – Transmitter Settings

Field	Setting
Device Name:	192.168.10.2
Carrier Frequency:	915 MHz
IQ Rate:	2 MS/s
Gain:	Use default value
Active Antenna:	TX1
Message Length:	200,000

Table XXXVI – Transmitter Message Settings

Field	Message 1 Settings	Message 2 Settings
Start Frequency	10 Hz	100 Hz
Delta Frequency	100 Hz	100 Hz
Number of Tones	1	1
Modulation Index	1	1
Subcarrier frequency	500 kHz	510 kHz



device names: 192.168.10.2
IQ rate: 2M
actual IQ rate: 2M
carrier frequency: 915M
actual carrier frequency: 915M
gain (dB): 0
actual gain (dB): 0
active antenna: TX1
message length (samples): 200000

start frequency 1: 10.00
delta frequency 1: 100.00
#tones 1: 1
Modulation index 1: 1
Subcarrier frequency 1: 500.00k

start frequency 2: 100.00
delta frequency 2: 100.00
#tones 2: 1
Modulation index 2: 1
Subcarrier frequency 2: 511k

error out
status: ✓
code: 0
source

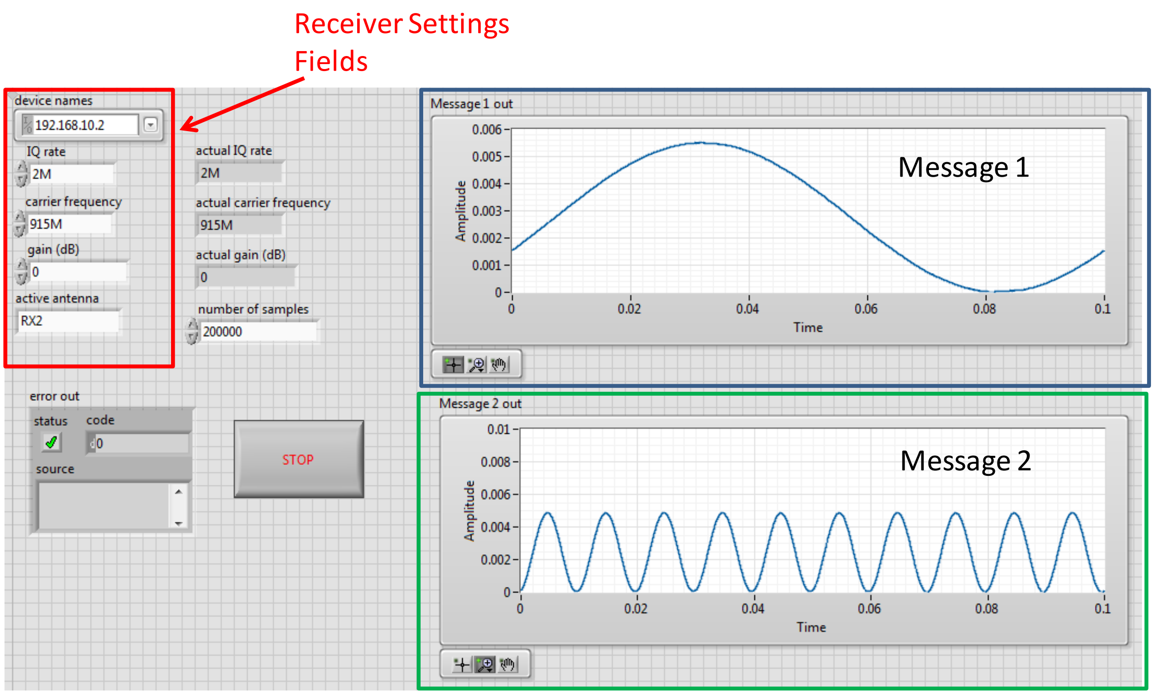
Fig. 179. Transmitter VI Front Panel

6. Ensure that the receiver VI is set up to use the settings in Table VII. (See Fig. 180)

Table XXXVII – Receiver Settings

Field	Setting
Device Name:	192.168.10.2
Carrier Frequency:	915 MHz
IQ Rate:	2 MS/s
Gain:	0 dB
Active Antenna:	RX2
Number of Samples:	200,000

Note that there is no offset between the transmitter's carrier frequency and the receiver's carrier frequency in this lab project



device names: 192.168.10.2
IQ rate: 2M
carrier frequency: 915M
gain (dB): 0
active antenna: RX2
actual IQ rate: 2M
actual carrier frequency: 915M
actual gain (dB): 0
number of samples: 200000
error out
status: ✓ code: 0
source:

Fig. 180. Receiver VI Front Panel

7. Run the receiver (FDM_Rx.vi) VI.
8. Run the transmitter (FDM_Tx.vi) VI.
9. After a few seconds, stop the receiver using the STOP button. Examine the Message Out graphs **(Fig. 180)** to ensure that the receiver is correctly demodulating and displaying the two message signals.
10. Stop the transmitter.

11.4.2 Worksheet: The Effects of Channel Separation and Cross-Talk

In telecommunications, crosstalk occurs when the signal from an adjacent channel causes distortion or interference. You may have experienced this when using a telephone and you hear someone else's conversation faintly in the back ground. One strategy is to convert the analog signals to a digital form, which is much less susceptible to crosstalk. In the case of wireless communications, crosstalk can be dealt with by ensuring that there is a large enough separation between channel carrier frequencies. **Fig. 181** shows the effects of cross talk where a 10 Hz signal and a 100 Hz signal are not sent on channels with enough separation between sub-channel carrier frequencies. You can clearly see ringing on the 10 Hz signal (top graph) and the superposition of the 10Hz wave as a bias in the 100 Hz signal (bottom graph).

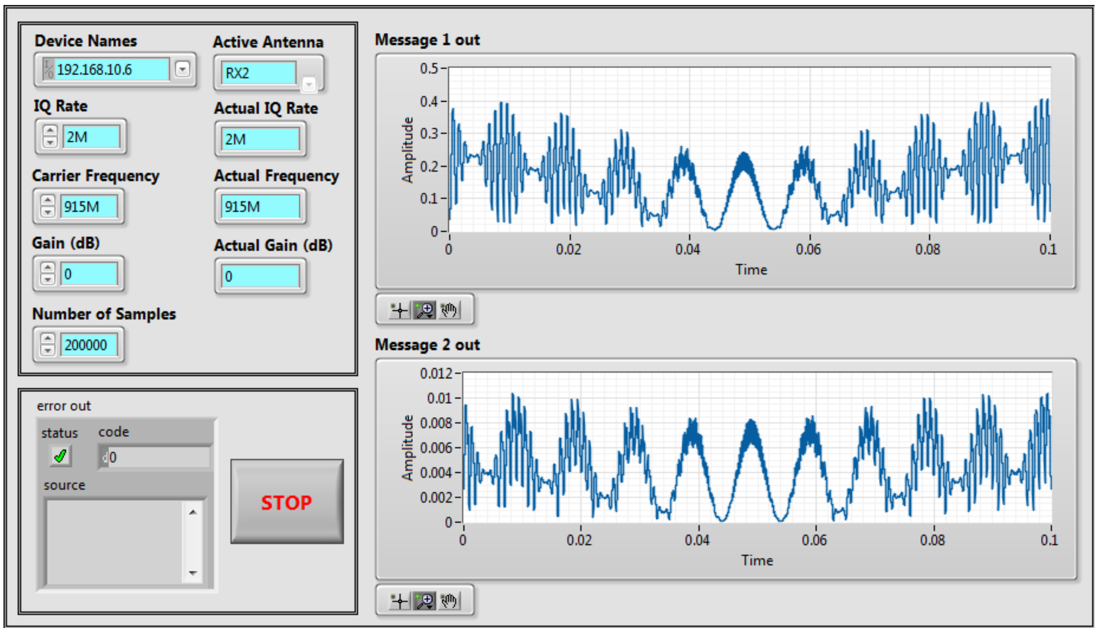

Device Names: 192.168.10.6
Active Antenna: RX2
IQ Rate: 2M
Actual IQ Rate: 2M
Carrier Frequency: 915M
Actual Frequency: 915M
Gain (dB): 0
Actual Gain (dB): 0
Number of Samples: 200000
error out
status: ✓
code: 0
source

Fig. 181. Cross Talk Example

1. Set the transmitter to use <u>one</u> of the three tones. (Please note that using more than one tone will make it very hard to make the observations.)
2. Check to ensure that the transmitter is using the settings in Table XI. (See Fig. 179).
3. Run the receiver VI.
4. Set the transmitter VI message frequencies to match the first set of frequencies given in Table VIII.
5. Start the transmitter VI.
6. Observe the wave form on the receiver VI. Record observations in Table VIII
7. Stop the transmitter VI.
8. Set carrier frequencies to the values in the next line in Table VIII and Repeat steps 5 through 8 until the table is complete

Table XXXVIII – Crosstalk Observations

Message 1 Carrier Frequency	***Message 2 Carrier Frequency***	***Observation (Are the channels Interfering with each other?)***
505 kHz	*505 kHz*	
505 kHz	*507 kHz*	
505 kHz	*509 kHz*	
505 kHz	*510 kHz*	
503 kHz	*510 kHz*	
500 kHz	*510 kHz*	
498 kHz	*512 kHz*	

Note: The table Carrier Frequency values reflect design parameters of the band pass filters used in your design. (Hint: this concept may help with answering questions in section 4.)

11.4.3 Worksheet: The Effect of Varying the Modulation Index

1. Check to ensure that the transmitter is using the settings in Table XI. (See Fig. 179).
2. Run the receiver VI.
3. Set the transmitter VI.Message1 and Message 2 modulation indices to match the first set of indices given in Table XXXIX.
4. Start the transmitter VI.
5. Observe the wave form on the receiver VI. Record observations in Table XXXIX
6. Stop the transmitter VI.
7. Set carrier frequencies to the values in the next line in Table XXXIX and Repeat steps 4 through 7 until the table is complete

Table XXXIX – Modulation Index Observations

Message 1 Modulation Index	*Message 2 Modulation Index*	*Amplitude (Peak to Peak)* (Fig. 182*)*	*Observation (Are the channels Interfering with each other?)*
0.01	*1.0*		
0.05	*1.0*		
0.1	*1.0*		
0.5	*1.0*		
1.0	*1.0*		

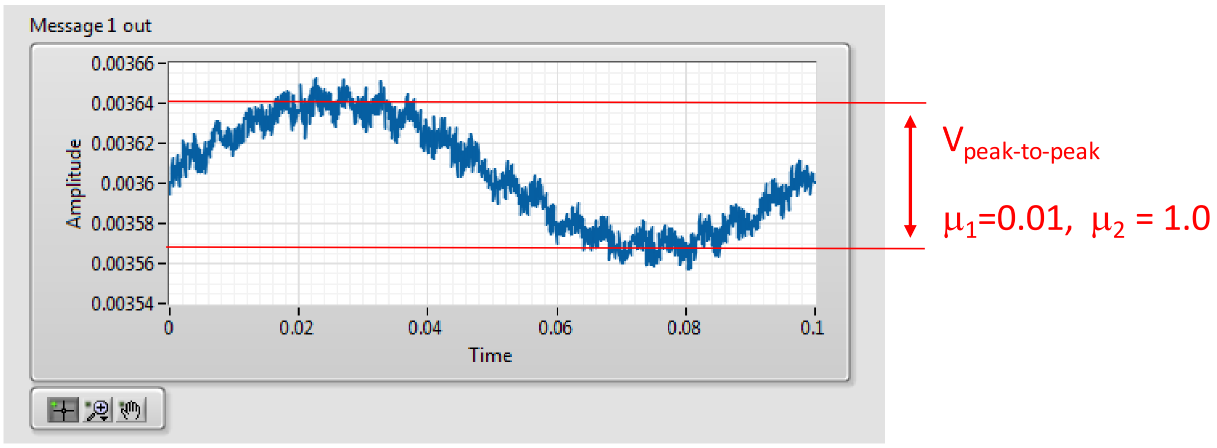


Fig. 182. Cross Talk Peak-to-Peak Example

11.5 Lab Write-up

<u>Performance checklist</u>

Frequency Modulation with noise

<u>Short Answer Questions</u>

1. As observed in the lab, the effects of one message having a much larger modulation index than the other causes distortion of the received signal for the message with the lower modulation index. Explain what the impact of transmitting on a noisy channel would have on the signals.

2. As observed in the lab moving the sub-carriers closer together creates cross talk. Explain how this would affect your spectrum allocation (number of sub-carriers used).

3. Explain whether changing the order of the band-pass filters or narrowing the pass band will or will not reduce cross talk?

4. Explain whether changing the order of the low-pass filters would or would not help with crosstalk?

<u>Performance Measures</u>

Task	Standards	Sat/Unsat
Worksheets and Short Answer Questions	Quality of answers and data collected.	
Running VIs	Successful transmission and reception of tones.	
Wiring of terminals	Clarity of Tx and Rx VI layouts.	

<u>Discussion</u>

Did all configurations perform as expected?
Did you have any difficulties completing the lab?
Did your TA provide enough guidance?
Do you have any recommendations to improve the lab?

11.6 References

[1] Lab 3: Frequency-Division Multiplexing, Bruce A. Black, Rose-Hulman Institute of Technology, July 2013.
[2] OFDM Basics Report, Yasser Ahmed Abbady. “Wireless communication course (fall 2006)” retrieved July 2, 2014, from http://www.angelfire.com/planet/wiresem/OFDM%20Basics.pdf.
[3] Wireless Communications, Andreas Goldsmith, Stanford University retrieved August 20, 2014, from http://wsl.stanford.edu/~andrea/Wireless/SampleChapters.pdf

12 Entropy and Coding Efficiency

12.1 Summary

The objective of this laboratory exercise is to give an experience of entropy encoding. In this lab, you will investigate procedure and efficiency of Huffman coding, a special form of entropy encoding, to compress the source. *You will perform Huffman coding to compress source data, form a codebook and derive the coded bits, and finally calculate entropy, average length and efficiency of the designed Huffman code.* Design and implementation of the Huffman coding algorithm would be carried out on National Instruments' LabVIEW.

Overall, this lab will answer the following questions:

- How does Entropy Encoding, particularly Huffman Coding work?
- How efficient is Huffman Coding?
- How can algorithms be implemented in NI LabVIEW?

12.2 Background

12.2.1 Digital Communications Systems

In digital communication systems, the source output is converted into a binary sequence by the source encoder (Fig. 183). This binary sequence is then processed by the channel encoder or modulator, and then transmitted over the channel. The incoming binary sequence is recreated by the channel decoder or demodulator, and the source output is recreated by the source decoder.

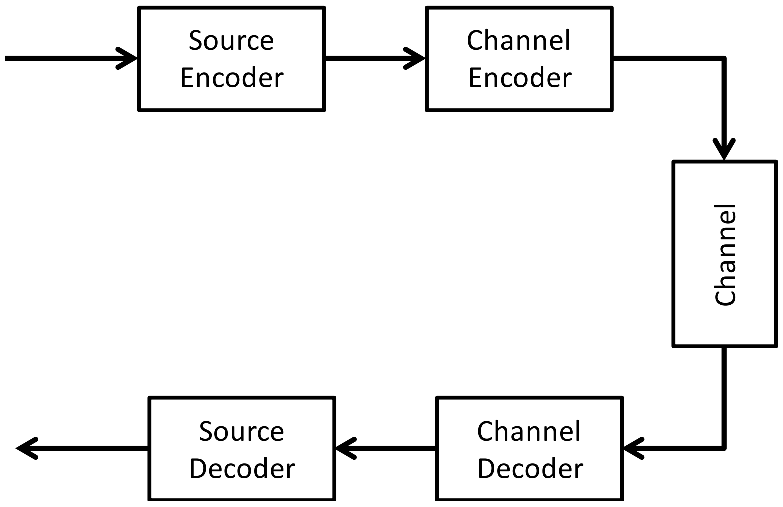


Fig. 183: Digital Communications Model

There are a number of reasons why communication systems now usually feature a binary interface between source and channel (i.e., why digital communication systems are now standard). Briefly the reasons are as follows:

- Digital hardware has become so cheap, reliable, and miniaturized, that digital interfaces are eminently practical.
- A standardized binary interface between source and channel simplifies implementation and understanding, since source coding/decoding can be done independently of the channel, and, similarly, channel coding/decoding can be done independently of the source.
- A standardized binary interface between source and channel simplifies networking, which now reduces to sending binary sequences through the network.

- Shannon's source/channel separation theorem states that if a source can be transmitted over a channel in any way at all, it can be transmitted using a binary interface between source and channel.

12.2.2 Source Coding, Entropy and Entropy Encoding

The objective of source coding is to compress the source data by exploiting the redundancies in the data, thereby enabling representation of the source with fewer bits that carry more information [1], [2]. Information from a source that produced different symbols according to some probability is described by entropy, $H(X)$. It is essentially the measure of uncertainity of a random variable with an associated probability set, $\mathrm{p}(x_i)$.

$$H(X) = -\sum_{i=1}^{n} (\mathrm{p}(x_i) \log\ \mathrm{p}(x_i)) \tag{75}$$

Entropy Encoding compresses the source data to minimize the average length of the source symbols. The aim is to create a coding scheme that achieves the limit of entropy of the source such that $C(x) \geq H(x)$, where, $H(x)$ is entropy of source, and $C(x)$ is the bit rate after compression or the average length of the coded source symbols. In particular, no source coding scheme can have a shorter average length than the entropy of the source.

12.2.3 Huffman Coding

Huffman coding is a special type of entropy encoding that results in an optimum code in the sense that the Huffman code has the minimum possible average word length for a source of given entropy [1]. Thus it is the code that has the highest efficiency.

The source output has samples, some of which are repeated more often than others. We exploit this redundancy and compress the data by making use of Huffman coding as discussed in class. It is assumed that you have general understanding of the Huffman coding procedure.

12.2.4 Source Coding Efficiency

One of the primary uses of Huffman coding is reducing the average number of bits that need to be sent for a given symbol. To illustrate this, let us suppose that the symbol set we are interested in is comprised of the letters in the English alphabet.

Table XL shows a typical example of the number of times (relative frequency) we would expect to see the letters (symbols) appear in a random piece of English text consisting of 40,000 letters. Table XLI shows a typical Huffman code generated for this sample of text. The average number of bits used to transmit the symbols in the text is approximately 4.25 bits/symbol.

Table XL -Relative Frequency of Letters in the English Language

Letter	Relative Frequency	Letter	Relative Frequency	Letter	Relative Frequency
a	3256	**j**	60	**s**	2524
b	596	**k**	308	**t**	3612
c	1108	**l**	1604	**u**	1100
d	1696	**m**	960	**v**	392
e	5184	**n**	2692	**w**	940
f	888	**o**	2992	**x**	60
g	804	**p**	768	**y**	788

Table XL -Relative Frequency of Letters in the English Language

Letter	Relative Frequency	Letter	Relative Frequency	Letter	Relative Frequency
h	2432	**q**	36	**z**	28
i	2780	**r**	2388	--	--

Following table presents the Huffman Code letters for the English alphabets.

Table XLI –Huffman Code Letters in the English Language

Letter	Huffman Code	Letter	Huffman Code	Letter	Huffman Code
e	100	**d**	11111	**p**	110001
t	000	**l**	11110	**b**	110000
a	1110	**c**	01001	**v**	001000
o	1101	**u**	01000	**k**	0010011
i	1011	**m**	00111	**j**	001001011
n	1010	**w**	00110	**x**	001001010
s	0111	**f**	00101	**q**	001001001
h	0110	**g**	110011	**z**	001001000
r	0101	**y**	110010	--	--

In the following sections of the lab, you will be asked to determine the average word length (76) and efficiency of the code (77) given by

$$Average\ length = \bar{L} = E\{\ell\} = \sum_{i=1}^{n} \mathrm{p}(x_i)\ \ell_i \tag{76}$$

and,

$$Efficency = H(x)/\bar{L} \tag{77}$$

where $\mathrm{p}(x_i)$ is the probability set of the random variable, ℓ_i is the length of i^th word, and $H(x)$ is the entropy of the source.

Using the statistics (Table XL) and the code (Table XLI), we can compile the data in Table XLII. This data can then be used with equations (76) and (77) to show the average word length is 4.2015 average bits and the entropy is 4.1722 average bits. So the code's efficiency is 0.9930.

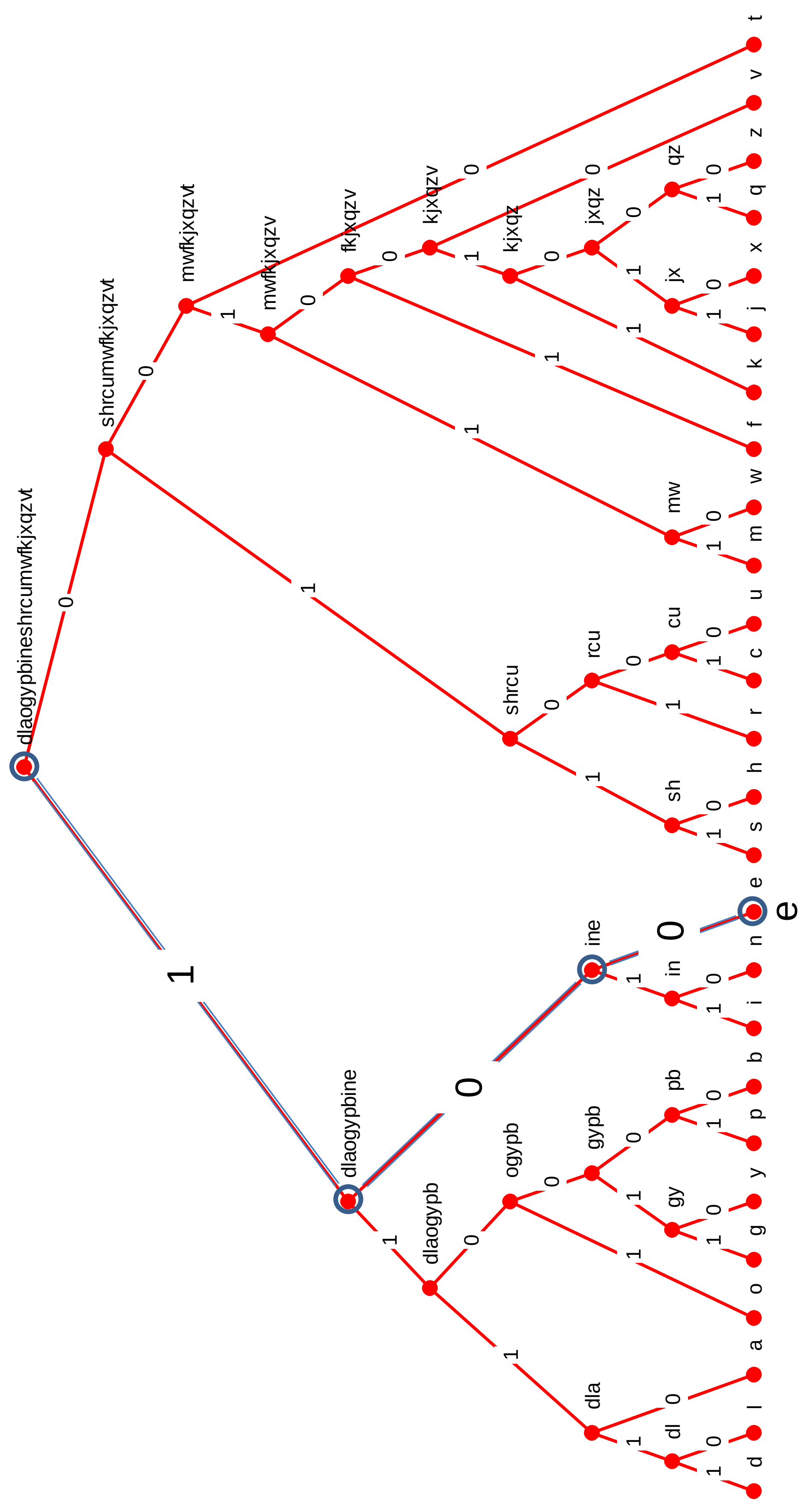


Fig. 184: Code Tree for Alphabet Code

Table XLII -Relative Frequency of Letters in the English Language

Letter	Length (ℓ_i)	Relative Frequency ($p(x_i)$)	Letter	Length (ℓ_i)	Relative Frequency(($p(x_i)$))	Letter	Length (ℓ_i)	Relative Frequency(($p(x_i)$)
a	4	0.0814	**j**	9	0.0401	**s**	4	0.0275
b	6	0.0149	**k**	7	0.0240	**t**	3	0.0098
c	5	0.0277	**l**	5	0.0673	**u**	5	0.0235
d	5	0.0424	**m**	5	0.0748	**v**	6	0.0015
e	3	0.1296	**n**	4	0.0192	**w**	5	0.0197
f	5	0.0222	**o**	4	0.0009	**x**	9	0.0007
g	6	0.0201	**p**	6	0.0597	**y**	6	0.02750
h	4	0.0608	**q**	9	0.0401	**z**	9	0.0098
i	4	0.0695	**r**	4	0.0240	--	--	--

Now suppose the symbols were encoded using a fixed length code of 5 bits per letter or symbol. One such code is shown in **Table** XLIII. Each letter would take 5 bits no matter how frequently it occurs in the text. This fixed length implies that on the average each symbol will take 5 bits. So the code's efficiency is 0.8344.

A difference of 0.80 average bits between the two codes does not seem like much but imagine sending a large volume of data, such as the rough draft of a novel for review, over the internet to 10 or 12 people. In that scenario, the average number of bits sent really begins to add up.

Table XLIII –Fixed Length Code Letters in the English Language

Letter	Relative Frequency	Letter	Relative Frequency	Letter	Relative Frequency
A	00000	**J**	01001	**S**	10010
B	00001	**K**	01010	**T**	10011
C	00010	**L**	01011	**U**	10100
D	00011	**M**	01100	**V**	10101
E	00100	**N**	01101	**W**	10110
F	00101	**O**	01110	**X**	10111
G	00110	**P**	01111	**Y**	11000
H	00111	**Q**	10000	**Z**	11001
I	01000	**R**	10001	--	--

Clearly from Table XLIII, we can make out that 11010, 11011, 11100, 11101, 11110, and 11111 are unused codes.

This lab illustrates the Huffman coding procedure using a source output of a few number of message symbols with given frequency of occurrence and calculate the efficiency of the designed Huffman code for the source. You will be responsible for understanding the Huffman encoding VIs, and then design VIs to compute code efficiency. Finally, you will integrate all VIs into a single VI that outputs the source entropy, Huffman code and average length, and efficiency of the designed Huffman code.

12.3 Pre-Lab

The following sub-VIs- Huffman_Tree.vi, Huffman_Codetree.vi and Efficiency_Code.vi will be used to design Lab_EE.vi, the final VI in this lab.
Prior to this lab it is expected that students-

- Review the Summary section.
- Understand the programming flow [1][4][4] in the Huffman_Tree and Huffman_Coding VIs.
- Understand how to build a connector pane [5] (generating terminals on a sub-vi).

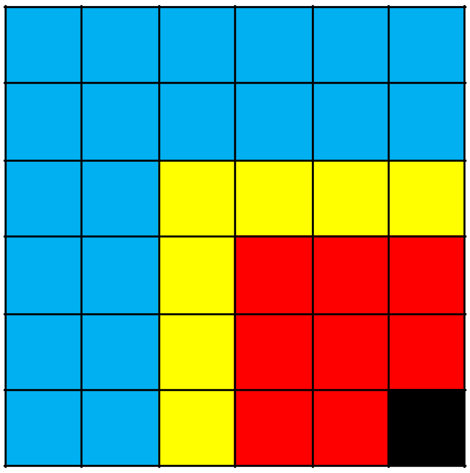
Note: Understanding the programming flow in Huffman_Tree and Huffman_Coding VIs is important since you will be asked to explain the working of these VIs in the lab write-up [d)]. You should make notes on the working of these VIs before coming to the lab so that you can simply attach your descriptions to the lab write-up. This will also help you to design the Efficiency_Code and Lab3 VIs within the timeframe of the lab.

12.4 Lab Procedure

The following is a description of the VIs to be used in the lab and the associated tasks to be accomplished for each one:

12.4.1 Code Tree for an Image

In this lab we will explore the use of Huffman codes to store the image shown in Fig. 185a. Fig. 185b provides a numeric color code for each pixel or color square in the image in Fig. 185a.



(a)

3	3	3	3	3	3
3	3	3	3	3	3
3	3	1	1	1	1
3	3	1	2	2	2
3	3	1	2	2	2
3	3	1	2	2	0

(b)

Fig. 185: Exercise Image

1. Complete Table XLIV by counting the number of squares with the color code. This is the data you will need to perform the experimental procedure. Note there are 4 color codes so N equals 4.

Table XLIV –Image Frequency Counts

Color (Node Number)	Relative Frequency (Count)
0	
1	
2	
3	

2. When you have finished with Table XLIV, start Huffman_Tree.vi.
3. Enter the Colors (Node Numbers) in the node entry fields (see Fig. 186).
4. Enter the relative frequencies or count in the Count entry field (see Fig. 186). Remember to enter the count that matches the color.

   Note: Count of symbols in 'Count' array should be inputted in increasing order (lowest probability to highest probability) for proper functioning of the designed VIs.

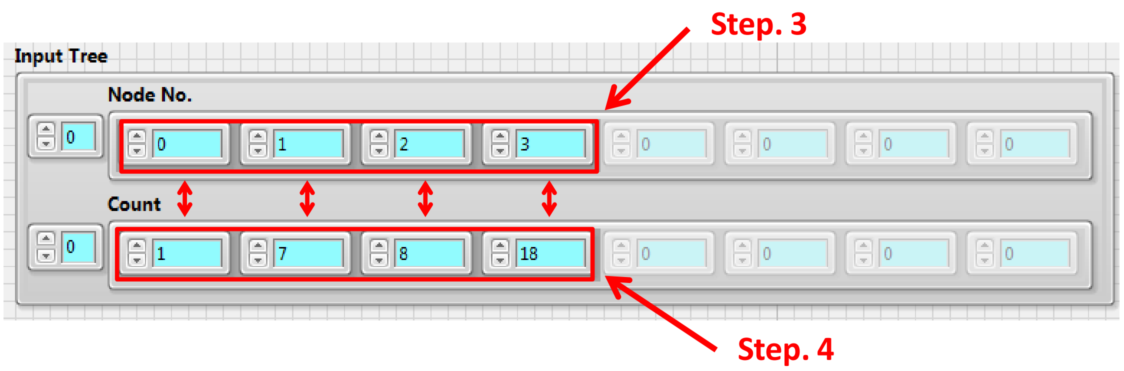
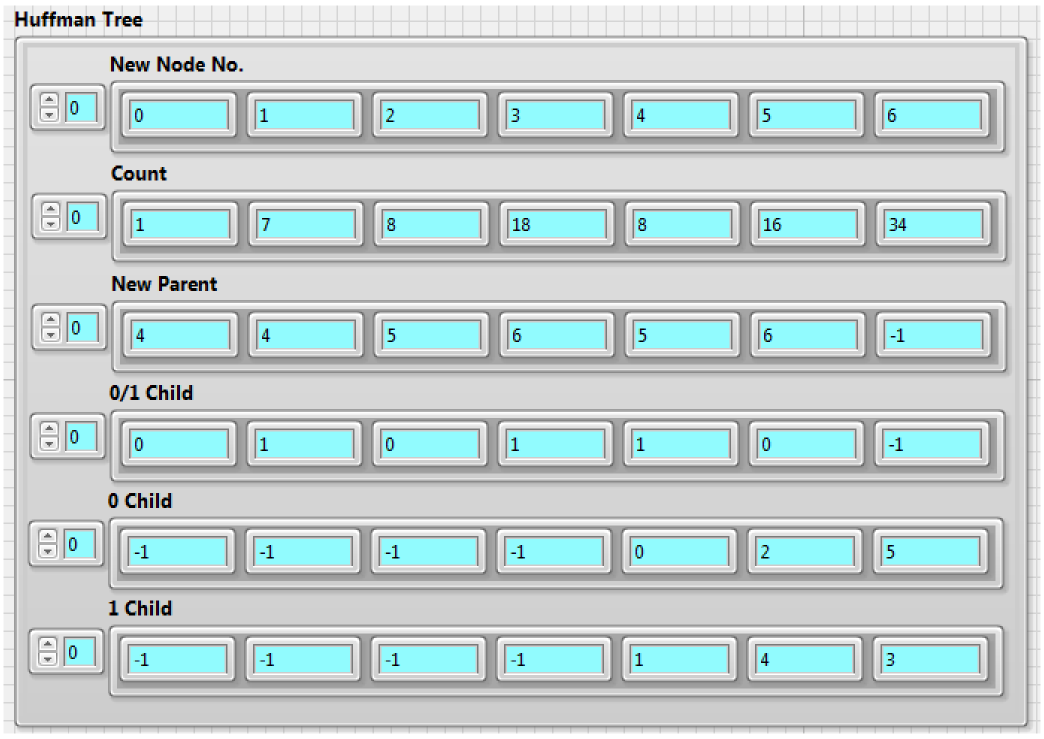


Fig. 186: Input Tree Front Panel Interface

   Note: Make sure you use the correct count values from Table XLIV. The example in the figures for this lab is an illustration to give an understanding of the problem.

5. Observe the resulting code tree output screen (Fig. 187).



Huffman Tree

	Index							
New Node No.	0	0	1	2	3	4	5	6
Count	0	1	7	8	18	8	16	34
New Parent	0	4	4	5	6	5	6	-1
0/1 Child	0	0	1	0	1	1	0	-1
0 Child	0	-1	-1	-1	-1	0	2	5
1 Child	0	-1	-1	-1	-1	1	4	3

Fig. 187: Output Tree Front Panel Interface

From these output fields, generate the Huffman tree as shown in **Fig. 188**. The Huffman tree has new internal nodes associated with a count that is the sum of the counts of its children nodes (nodes 4, 5, and 6). We also associate the symbol nodes and these newly generated nodes with a count (Fig. 189), parent node (Fig. 190), 0/1 Child (Fig. 191), 0 Child and 1 Child. All these arrays are the output of the Huffman tree.

Note: The number of message symbols is N. Indexing in LabVIEW starts at zero; so we assign the node numbers from 0 to N-1. Similarly, the total number of nodes (symbol nodes + newly created internal nodes) in the Huffman tree (Output) is 2*N-1, and so the last node number is 2*N-2.

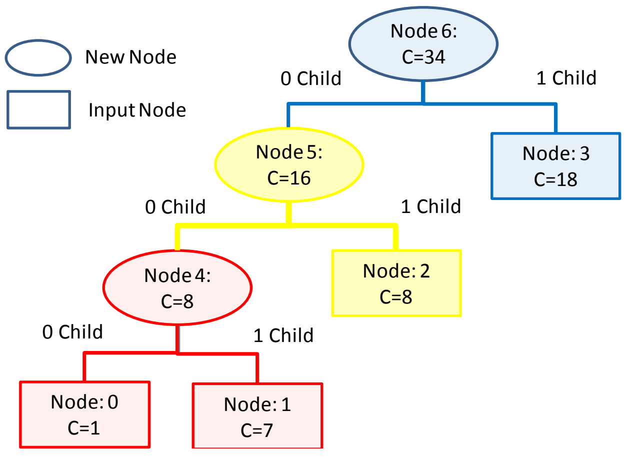
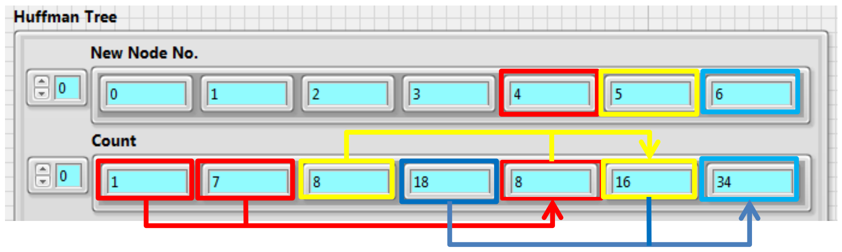
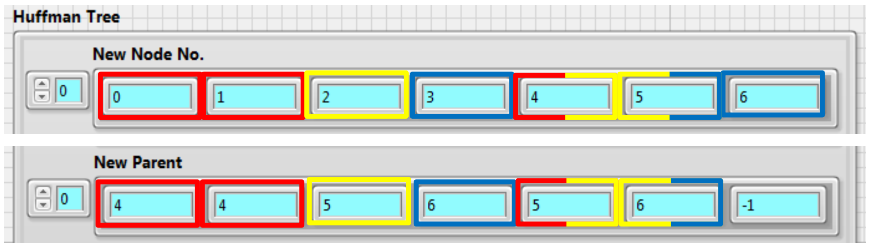


Fig. 188: Huffman Tree


Huffman Tree

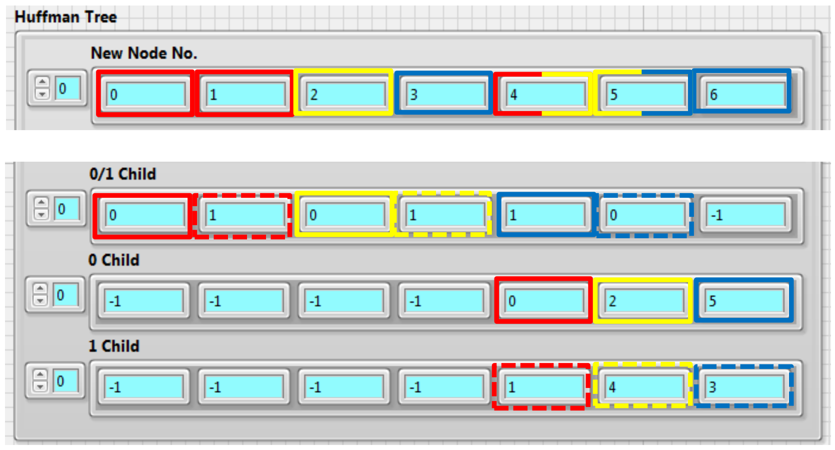
New Node No.	0	1	2	3	4	5	6
Count	1	7	8	18	8	16	34

Fig. 189: Node Counts for New Nodes


Huffman Tree

New Node No.	0	1	2	3	4	5	6
New Parent	4	4	5	6	5	6	-1

Fig. 190: Parent Node Associations


Huffman Tree

New Node No.	0	1	2	3	4	5	6
0/1 Child	0	1	0	1	1	0	-1
0 Child	-1	-1	-1	-1	0	2	5
1 Child	-1	-1	-1	-1	1	4	3

Fig. 191: Parent Node Associations

Taking into account the complexity and time requirements of programming this VI, *Huffman_Tree.vi* has already been designed for you. However, you need to understand the VI and write a description of its working. (See Lab Write-up Section, Question 1)

12.4.2 Huffman Codebook

The Huffman codebook (Huffman_Codebook.vi) is a dictionary that allows you to look up the coded bits for a symbol. The input would be the output of the Huffman Tree (Fig. 187).

The code for each symbol (Fig. 192) is formed by generating a string by traversing the Huffman tree upwards (from the specific symbol node to the top (root) node) and then reversing the order of the string.

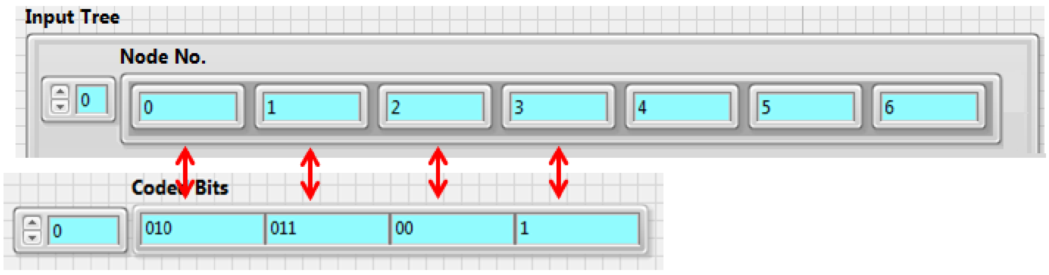


Fig. 192: Huffman Codebook Sequences Front Panel Interface

Taking into account the time requirements of programming this VI, *Huffman_Codebook.vi* has already been designed for you. However, you need to understand the VI and write a description of its working. (See Lab Write-up Section, Question 2)

12.4.3 Code Efficiency Computations

In this part of the lab procedure, you will write a VI (Code_Efficiency.vi) that automates the computation of the average length (76), entropy (5) and code efficiency (77) of the given source. The input to this VI (Fig. 193) will be the node number and count of the symbols (input for Huffman_Tree.vi) and the coded bits (output of Huffman_Codebook.vi).

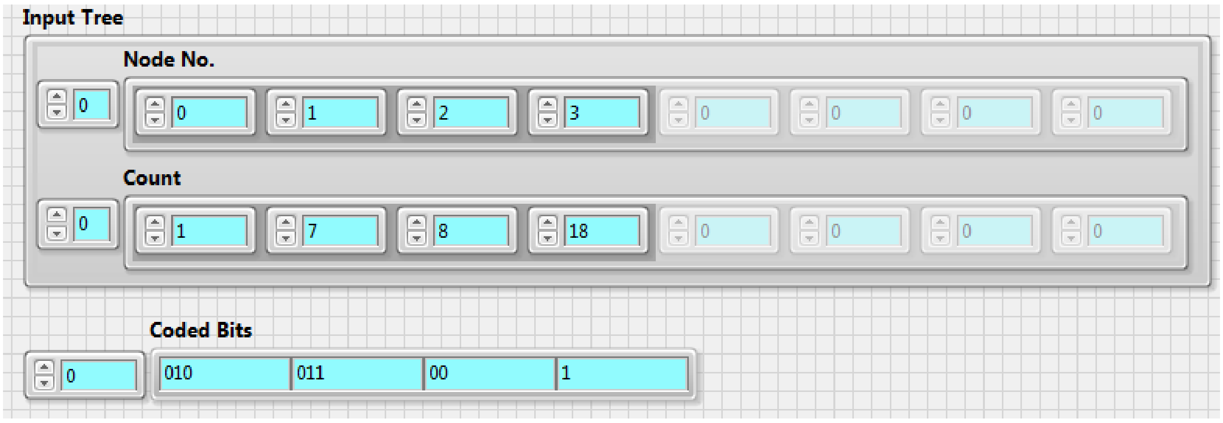


Fig. 193: Code Efficiency Input Front Panel Interface

The output is shown below (Fig. 194).

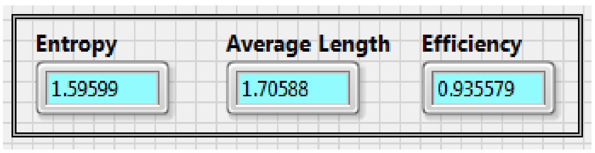
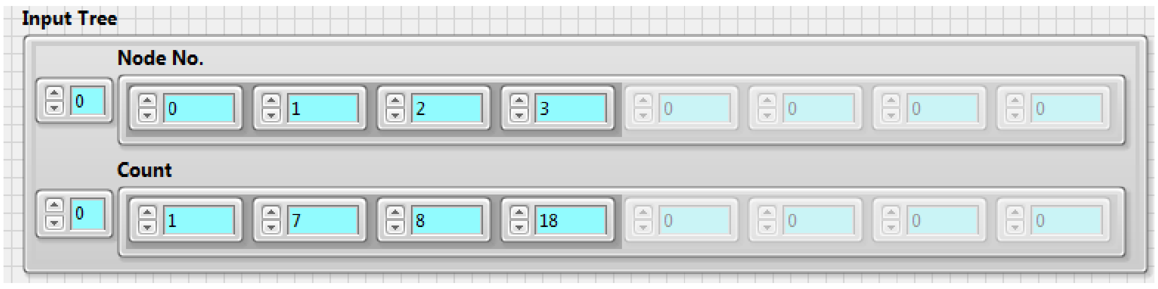


Fig. 194: Huffman Code Statistics Front Panel Interface

12.4.4 Lab_EE.vi

After designing Code_Efficiency.vi, the final task is to put all VIs together in Lab_EE.vi.
The input for this VI (Fig. 195) is a source having symbols with a given probability distribution. The output (Fig. 196) would be the entropy and Huffman code for this source, as well as the average length and efficiency of the generated Huffman code.



Input Tree

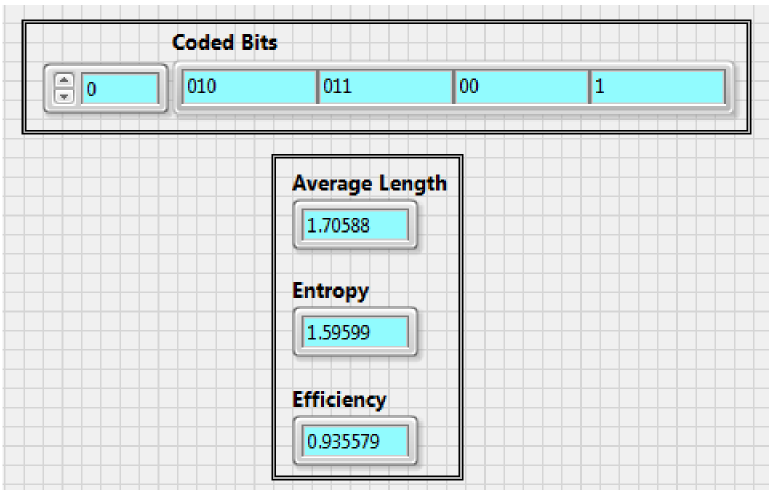
Node No.

0	0	1	2	3	0	0	0	0

Count

0	1	7	8	18	0	0	0	0

Fig. 195: Huffman Code Input for Complete VI Front Panel Interface



Coded Bits

0	010	011	00	1

Average Length
1.70588

Entropy
1.59599

Efficiency
0.935579

Fig. 196: Huffman Code Output for Complete VI Front Panel Interface

12.5 Lab Write-up

Performance Checklist
Source Coding

Analytical Questions

1) Describe the working (programming flow) of Huffman_Tree.vi (Please attach your answer).

2) Describe the working (programming flow) of Huffman_Codebook.vi (Please attach your answer).

Performance Measures

Task	Standards	Satisfactory/ Unsatisfactory
Description of 1. Huffman_Tree.vi 2. Huffman_Codebook.vi	Quality of description of: 1. Huffman_Tree.vi 2. Huffman_Codebook.vi	
Design of Code_Efficiency.vi	Functionality achieved: 1. Entropy (Yes/No) 2. Average Length (Yes/No) 3. Efficiency (Yes/No)	
Design of Lab_EE.vi	Functionality achieved (Yes/No)	

Discussion

Did all VIs perform as expected?
Did you have any difficulties completing the lab?
Did your TA provide enough guidance?
Do you have any recommendations to improve the lab?

12.6 References

[1] R. E. Ziemer and W. H. Tranter, Chapter 10: Information theory and Coding, In "Principles of Communications: Systems, Modulation and Noise", Fifth Edition.
[2] Dr. Sachin Katti, Jeff Mehlman and Aditya Gudipati, Lab 1: Source Coding, Stanford University.
[3] National Instruments, Tutorial: Data Structures in NI LabVIEW, retrieved March 20, 2014, from http://www.ni.com/gettingstarted/labviewbasics/datastructures.htm.
[4] National Instruments, Tutorial: Execution Structures in NI LabVIEW, retrieved March 22, 2014, from http://www.ni.com/gettingstarted/labviewbasics/exestructures.htm.
[5] National Instruments, Tutorial: Passing Data Between Loop Iterations in NI LabVIEW, retrieved March 22, 2014, from http://www.ni.com/gettingstarted/labviewbasics/shiftregisters.htm.
[6] National Instruments, Tutorial: SubVIs, retrieved March 20, 2014, from http://www.ni.com/white-paper/7593/en/.
[7] Data Compression: Statistical Distributions of English Text, retrieved June 14, 2014, from http://www.data-compression.com/english.html.
[8] Essays about Computer Security, Lee, E. Stewart. "University of Cambridge Computer Laboratory", retrieved June 14, 2014, from http://www.cl.cam.ac.uk/~mgk25/lee-essays.pdf.

13 Asynchronous Serial Communication

13.1 Summary

This lab is designed to help you develop a better understanding of one widely used digital protocol: Universal Asynchronous Receive/Transmit (UART) [1]. This communication protocol is used in nearly every embedded device, from mp3 players to digital watches, and is the basis for many older communication standards like RS-232 and Infrared TV remotes.

In digital communication systems, the source output is converted into a binary sequence by the source encoder (**Fig. 197**). This binary sequence is then processed by the channel encoder or modulator, and then transmitted over the channel. The incoming binary sequence is recreated by the channel decoder or demodulator, and the source output is recreated by the source decoder.

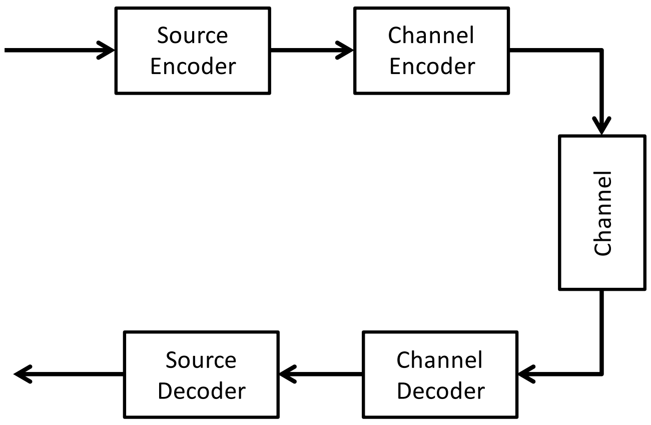


Fig. 197: Digital Communications Model

In this lab, you will be given a text string encoded using the American Standard Code for Information Interchange (ASCII). The lab addresses the link between source coding/decoding and channel encoding/decoding in Fig. 197. The link is a serial interface that uses an UART to convert the encoded text into a sequence or stream. The UART bit stream is used by the channel coding, channel modulation, channel decoding. To simplify the lab, the transmitted bit stream is passed directly to a UART receiver that reconverts the stream into the ASCII codes. You will be responsible for building the receiver portion of the UART for this lab.

13.2 Background

13.2.1 UART Basics

The UART packet structure consists of 10 bits- 1 START bit, 8 DATA bits (one byte), and 1 STOP bit. This structure is shown in Fig. 9 below, and is commonly referred to as N 8 1 (8 data bits, no parity, 1 stop bit). Sometimes an additional bit is included for parity checking (a simple form of error detection), but that is not covered in this lab. The data byte is sent Least Significant Bit (LSB) first, which means we effectively send the byte in reverse order. For instance, if we wanted to send the byte, 11001100, we would transmit the following 10-bit sequence: 0001100111 (one start bit (0), the 8 data bits LSB to MSB (00110011), and then one stop bit (1)).

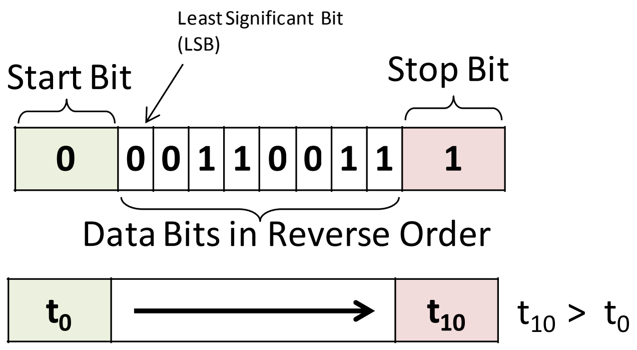


Fig. 198: UART Packet Structure

These bits are encoded as voltages: a HIGH voltage (i.e. 5 volts) to represent a 1, and a LOW voltage (i.e. 0 volts) to represent a 0. The START bit is always a LOW, and the STOP bit is always a HIGH. Each bit state is held for some period of time - the symbol period - to generate a ’square-wave’ signal. The amount of time for which we hold a LOW or HIGH bit determines the baud rate (speed) of the link. In most systems, the transmitter and receiver agree upon this baud (or bit-clock) rate in advance. The resulting packet signal is sent along a wire from the transmitter to the receiver.

UART packets can be sent from the transmitter at any point in time, and there can be any amount of time in between packets, hence the asynchronous nature of this communication link. In order to achieve synchronization with an incoming packet, the communication wire idles in the HIGH state in between packets. Since the START bit is always a LOW, we know a packet has begun when this transition occurs. After we synchronize to the start of a packet, we use the known baud rate to estimate the center of each data bit, and sample the voltage of the signal at this point. Fig. 199, shows the START bit (voltage level dropping) and the data-bits being sampled at a fixed interval. After the receiver decodes the entire data packet, the order of the received bits (00110011) is reversed to recover the original byte (11001100). The STOP bit simply returns the communications wire to the original IDLE (HIGH) state, and the receiver begins waiting for the next START bit which signals the beginning of the next packet.

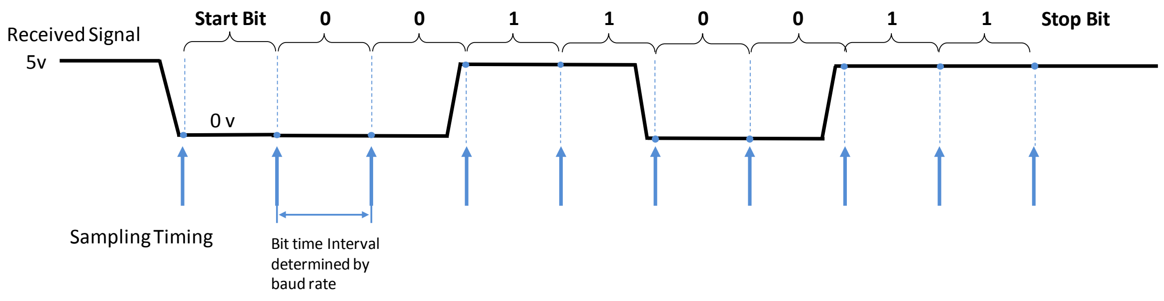


Fig. 199: Signal Timing (Baud Rate)

The signal is often oversampled (Fig. 200) to make the process of sending the data more robust. This feature is best understood by examining an example. Suppose that the signal is transmitted through a channel that causes every fourth bit to flip from 1 to 0 or vice versa. If the signal waveform is sampled only once, there is not enough information to determine if the data received is the data sent. That is why the parity bit is used in the standard. However, the parity bit can only be used to indicate is corrupted. It cannot be used to recover the original data. Now suppose more than one copy of the sampled value is sent. We can now compare the a bit received with the received copies. The number of copies depends on the error correction scheme being used. In this lab each bit is oversampled by a factor of 4. (For the purposes of this lab you will not be asked to deal with corrupted bits.)

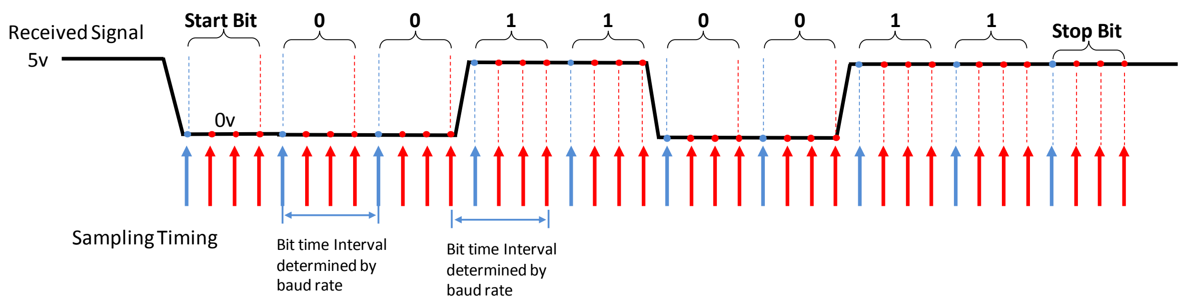

Received Signal
5v
0v
Start Bit
0
0
1
1
0
0
1
1
Stop Bit
Sampling Timing
Bit time Interval determined by baud rate
Bit time Interval determined by baud rate

Fig. 200: Signal Timing With Over Sampling (Baud Rate)

13.3 Pre-Lab

Your task in this lab is to build a functional UART receiver that decodes a message. However, to design the receiver, you need to understand the UART transmitter. The transmitter takes each character to a binary form as shown in Fig. 201.

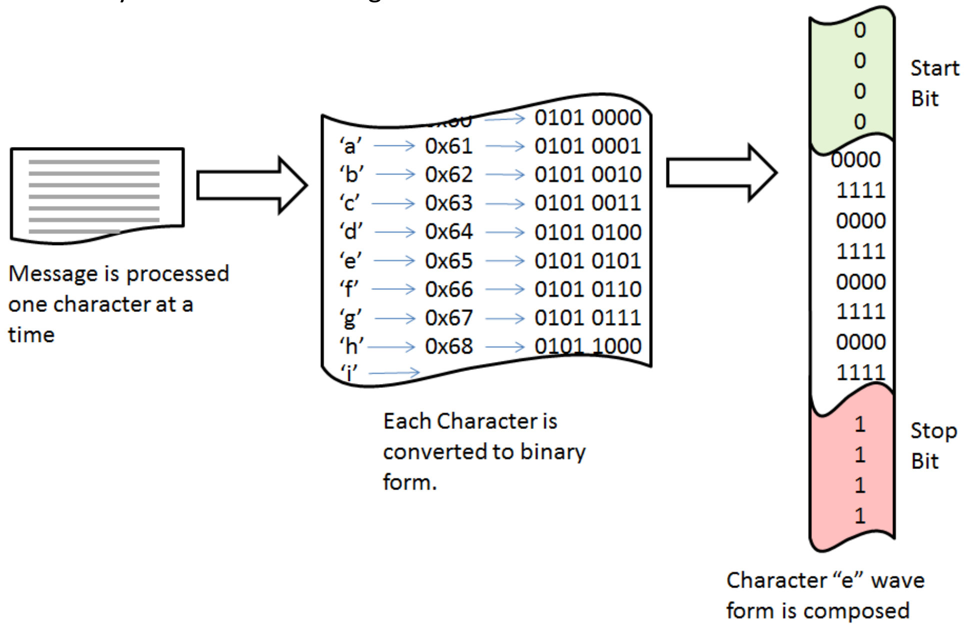


Fig. 201: UART Transmitter Schematic

The transmitter encodes 0's as 0 Volts and 1's as 5 Volts. Each character of the message is represented using ASCII encoding. ASCII is a one-byte (8 bit) representation for 128 of the most of the commonly used characters and punctuation in the English language. As a result, each UART packet you will receive consists of a single character from this message. The characters of the message are sent in order, so they should be kept in the order that they are received. After decoding the entire stream, you will have received a question (message). Let's open the ***"UART_Receiver.vi"*** sub-VI template and get started.

The receiver requires the use of a state machine (Fig. 202), with two basic states:

State 1: **IDLE**, waiting to see a transition from HIGH->LOW. On transition, go to the **READ** state.

State 2: **READ**, sample bits in the stream for each data bit. After you read 8 bits, convert these to a byte and return to **IDLE**.

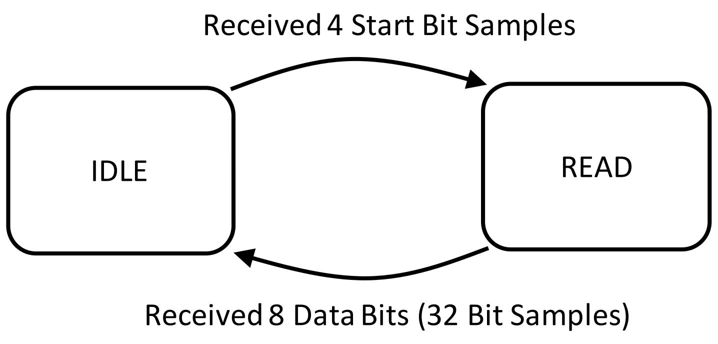

Received 4 Start Bit Samples
IDLE
READ
Received 8 Data Bits (32 Bit Samples)

Fig. 202: State Machine

In LabVIEW, you can create a basic state machine with a "for" loop, a shift register, a case statement, and some form of case selection mechanism. The basic structure of the state machine is shown in Fig. 203 and Fig. 204. The states are identified using an enumerated data constant (**IDLE** (0) and **READ** (1)). The shift register keeps track of which state (**IDLE** or **READ**) should be used in the next loop iteration. The value in the register is set to **READ** if a START bit is received and to **IDLE** after 8 bits have been received. The enumerated data constant definition is used to initialize the shift register. The case statement is set up so the 'false' case is **IDLE** (Fig. 203) and 'true' case is **READ** (Fig. 204). The **IDLE** case contains a block diagram need to detect the transition from a high to a low bit (the arrival of the start bit) and a counter to recover the 3 oversampled copies of the start bit. The **READ** case will recover each bit and the oversampled copies, reverse the bit order and convert the bits into ASCII code. The "for" loop forces the evaluation of the state selection logic (case statement) and state block diagrams every sample received. The loop will continue to execute until all the samples have been processed.

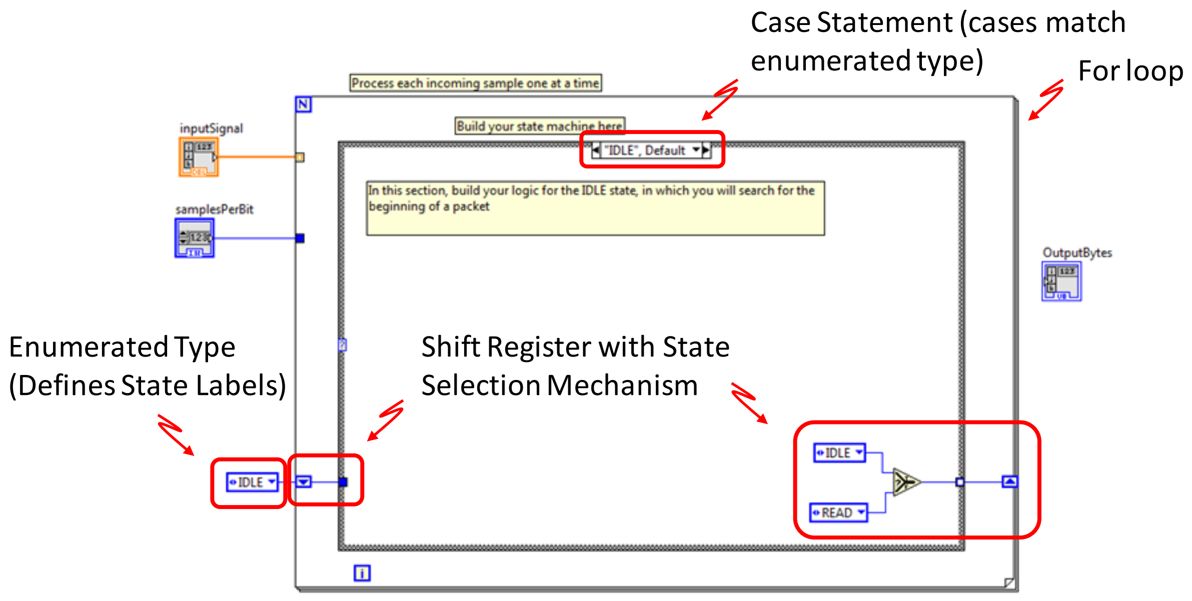

In this section, build your logic for the IDLE state, in which you will search for the beginning of a packet
Case Statement (cases match enumerated type)
For loop
Process each incoming sample one at a time
Build your state machine here
"IDLE", Default
inputSignal
samplesPerBit
OutputBytes
Enumerated Type (Defines State Labels)
Shift Register with State Selection Mechanism
IDLE
IDLE
READ
N
i

Fig. 203: Lab View Idle State Template

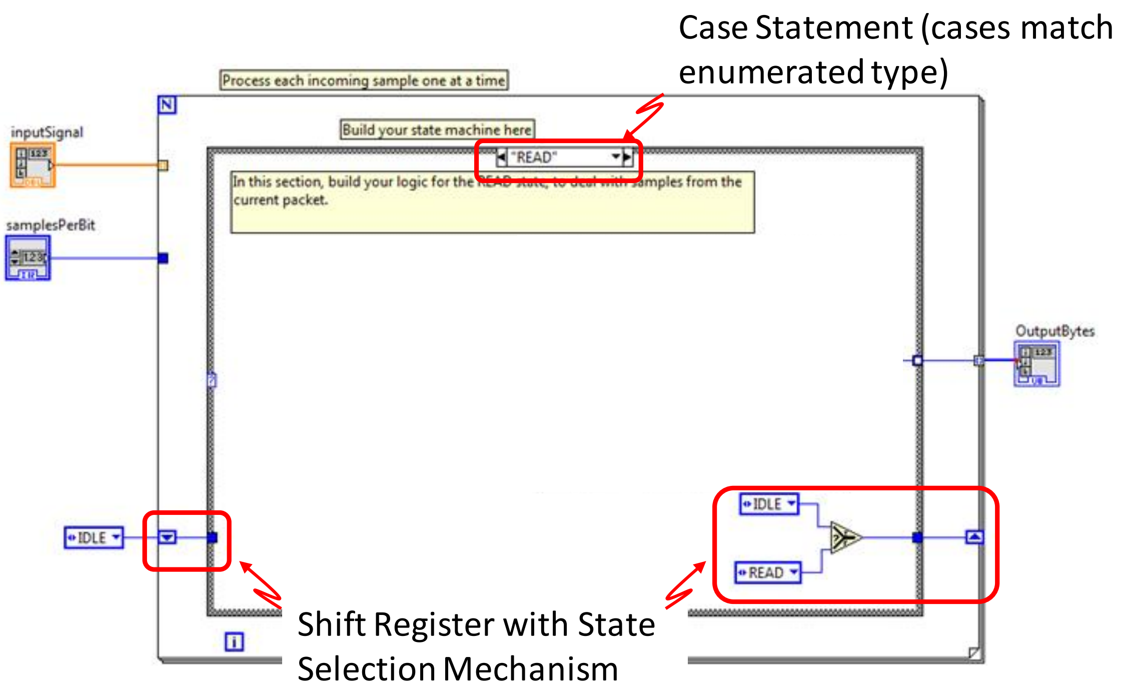

Case Statement (cases match enumerated type)
Process each incoming sample one at a time
Build your state machine here
"READ"
In this section, build your logic for the READ state, to deal with samples from the current packet.
inputSignal
samplesPerBit
OutputBytes
IDLE
IDLE
READ
Shift Register with State Selection Mechanism

Fig. 204: Lab View Read State

The UART receiver block takes two inputs: an array representing input signal (bit sample values) and the number of samples per bit or bit copies. You have to complete the missing logic in the UART_Receiver.vi by completing these four tasks: Sample Voltage to Bit Polarity Mapper; Bit Detection; Bit Counting and Storage; and Binary to Decimal conversion Conversion and Byte Storage.

13.3.1 Sample Voltage to Bit Polarity Mapper

In this task, you have to design logic inside the UART_Receiver.vi template provided to you to identify the input signal "samples" as HIGH (5Vdc) or LOW (0V). The results of this comparison are then used by the bit detection subVI. It is important that when a HIGH bit is detected the HIGH bit counter's input should be a "True" and the low bit counter should receive a "False". When a LOW bit is detected the LOW bit counter needs to receive a "True" and the HIGH bit counter needs to receive a "False".

Hint: You can do this with a comparison and a negation block.

13.3.2 Bit Detection

In this task, you have to design a resettable counter using the Resettable_Counter_Bit_Detector.vi template. Remember, a HIGH bit would be detected on reception of 4 samples with a value of 5, and a LOW bit will be detected on reception of 4 samples with value of 0.
So the resettable counter subVI for bit detection should

a) Take the Boolean input from the mapper and convert it to an integer 1 or 0 (using the type casting control highlighted in Fig. 205). Debugging Tip: One source of unexpected behavior can be the result from the mapper not outputting a boolean but an integer value.
b) The integer is then added to the previous total. You should use a feedback node (Fig. 206) to feed the count of received samples in the current loop iteration to the next loop iteration.
c) The resulting count is compared to the "samples per bit" input.

d) If the count is equal to the samples per bit, you should signal the detection of a bit using the bit detection Boolean as output and reset the counter. Resetting the counter will require the input to the feedback node change to a zero instead of the current count.

This subVI will be reused to perform bit detection for both HIGH and LOW bits so as to decode the ASCII code correctly. Save the VI move to the next step.

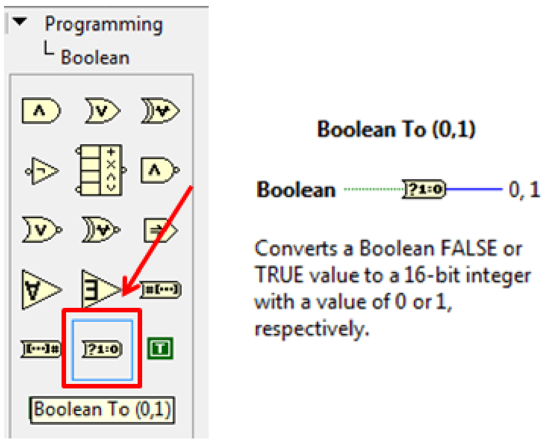

Programming
Boolean
Boolean To (0,1)
Boolean To (0,1)
Boolean
0, 1
Converts a Boolean FALSE or TRUE value to a 16-bit integer with a value of 0 or 1, respectively.

Fig. 205: Boolean To (0, 1) Convertor

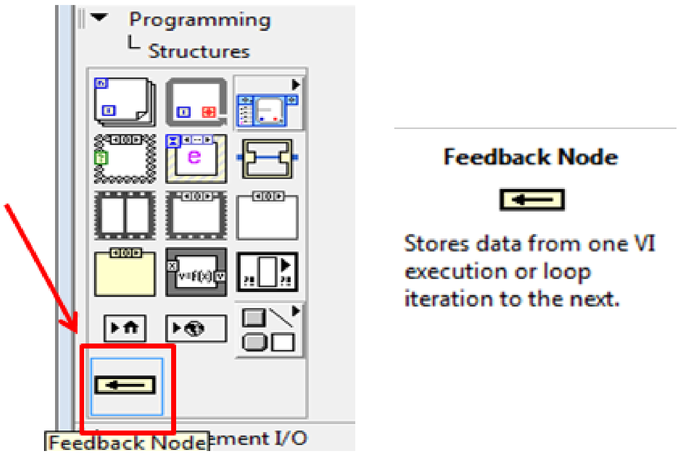

Programming
Structures
Feedback Node
Stores data from one VI execution or loop iteration to the next.
Feedback Node
ment I/O

Fig. 206: Feedback node

13.3.3 Bit Counting and Storage

In this task, you are to complete the Bit_Counter_and_Storer.vi subVI.

a) Start by copying the block diagram in Resettable_Counter_Bit_Detector.vi into the Resettable_Counter_Bit_Counter.vi and save.
b) Delete the “samples per bit” input and replace it with a constant.
c) Modify the resettable counter designed for bit detection to output a Boolean indicator after 8 bits have been counted indicating a byte has been received. The boolean activates the case structure for the next subVI (that performs binary to decimal conversion and received byte storage) when the Boolean output is “true”.
d) Unlike before, the counter must also output the count of received bits as it increments. This count is used as the index value for the bit storage array in the order they are received in.
e) Connect the count to an Replace Array Subset Function block. Using a Boolean to (0,1) conversion of the replacement value.

13.3.4 Binary to Decimal Conversion and Byte Storage

In this task, you will design a subVI that converts the received bits to an ASCII coded value (using binary_to_decimal.vi sub-VI) and stores the decimal equivalent of binary bytes in an array. This process forms the output of the UART Receiver subVi.

a) Start by using a case structure where if the “Byte Received Boolean” input is

- True - A byte has been received. First reverse the array elements in the “Stored Bits Array”. The resulting array is then converted into an ASCII code using the binary_to_decimal.vi subVI. (You do not have to build this subVI it has been provided in the template. Use the “Build Array” control in the “Array” functions palette to append the received byte onto the end of the “Previous Loop Iteration O/P Bytes Array” input and output this as the new “O/P Bytes Array”.

- False - A byte has not yet been received. In this state the “Previous Loop Iteration O/P Bytes Array” input should be passed through to the “O/P Bytes Array” without change.

b) The O/P Bytes Array out should be connected to shift register for the output of the UART Receiver. If the last bit sample value has not been processed then the shift register will pass the array onto the next loop execution as the “Previous Loop Iteration O/P Bytes Array” input [4].

13.4 Lab Procedure

With this lab, you will assemble UART Receiver using sub-VIs designed in the pre-lab section. You will be able to test UART Receiver with the ***“Lab11.vi”.*** The lab VI includes an UART transmitter which encodes a simple text message using ASCII coding and then generates a serial message stream to be received your UART receiver. If your receiver works, you should be able to read the question coded by the UART ASCII Transmitter.

In the **READ** state, you should use the two resettable counters for bit detection, a bit counter and storer, and a Binary to Decimal convertor and byte storer. In the **IDLE** state, you should use check for the reception of a LOW bit and once the LOW bit has been received, the VI should switch to the “Read” state. Once you wire all the blocks according to their functionality, run Lab11.vi. In case you do not see a sensible message (the question), you should debug each sub-VI you built in prelab to figure where you erred.

13.4.1 Debugging the Resettable Counter SubVI

1. Set up a break point on the Resettable_Counter_Bit_Detector.vi subVI (Fig. 207) in the **IDLE** state of the state machine.

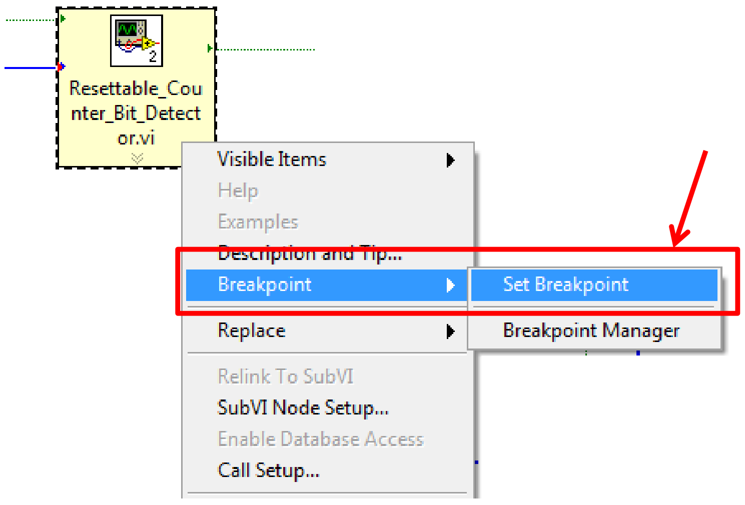
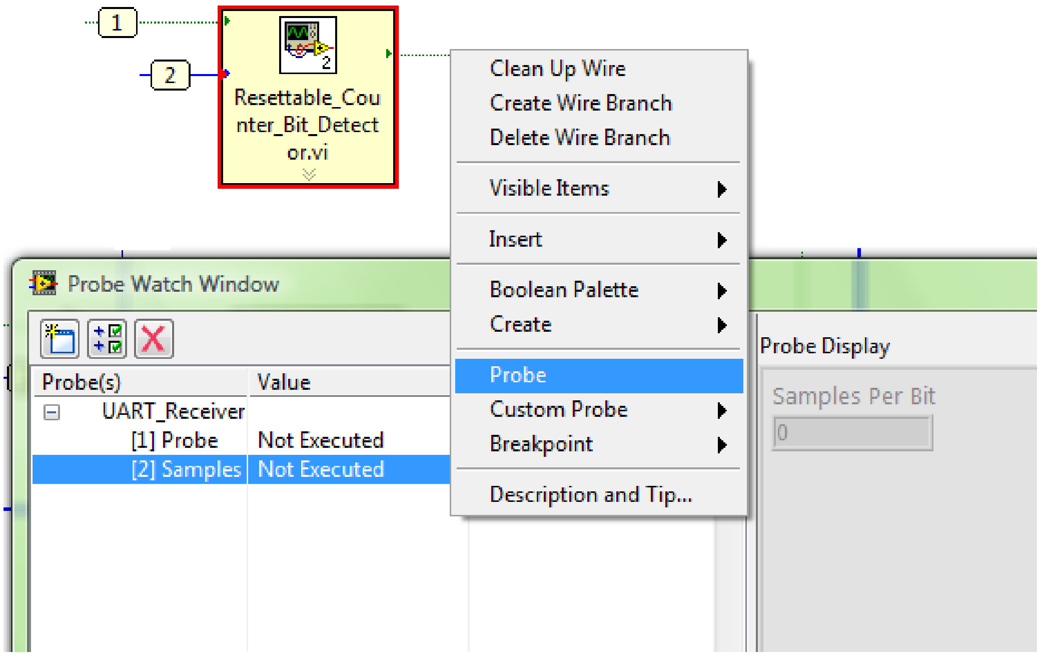


Fig. 207: Setting up a breakpoint

2. Set up probes (Fig. 208) on the outputs of the resettable counters that are inputs to this subVI.



Probe(s)	Value
UART_Receiver	
[1] Probe	Not Executed
[2] Samples	Not Executed

Fig. 208: Setting Up a Probe

3. Run the Async Serial VI. The execution of the block diagram halts at the break point. You will see that the probes on the input side of the subVI are executed and the probes on the output side have not been executed. At this point the probe labeled 1 should read low or false. The other input should have the value 4.
4. Hit “Continue” on the UART_Receiver.vi sub-VI to run the second loop iteration. You will once again get false values on both probes (only two samples read). The VI execution will halt again at the same break-point. Now the third probe should have a value that represents the output from the previous iteration. It should read false and the other two probes should read the same as before.

5. Again hitting "Continue" would result in the same result on the probes (only three samples read). The third time you hit the "Continue" button, the third probes should go true since 4 samples have been read.
6. To stop the VI once debugging is over, hit "Stop" button on the UART_Receiver.vi block diagram panel, and then hit the "Pause/Continue" button. To clear the breakpoint, right click on the sub-VI where you imposed the breakpoint, scroll over to "Breakpoint" and then click on "Clear the breakpoint" (Fig. 209).

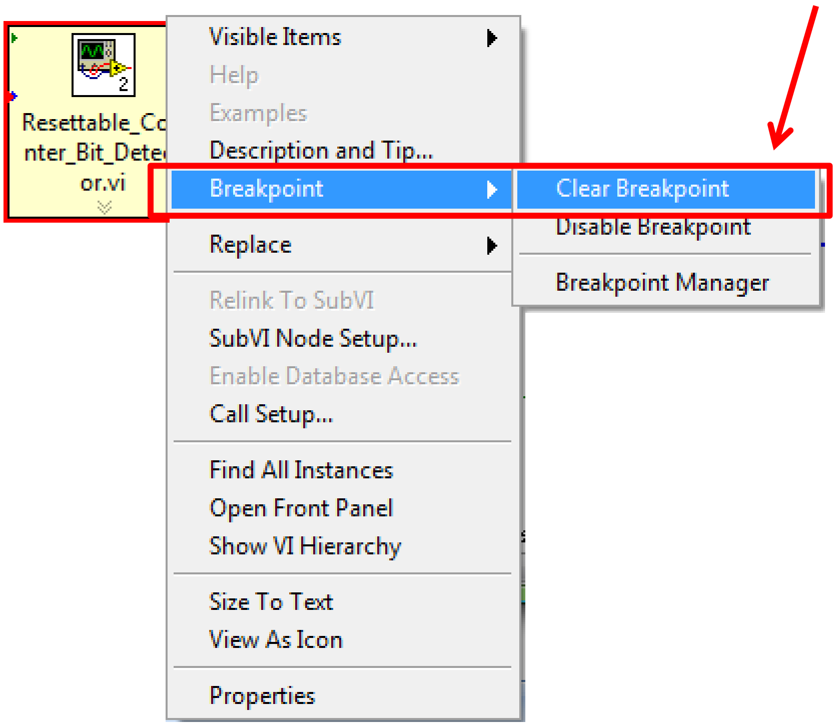


Fig. 209: Clearing a Breakpoint

Should the subVI not work as expected, you will need to perform a similar process on the block diagram for the subVI.

13.4.2 Debugging the Bit Counter and Storer

1. Set up a break point on the Bit_Counter_and_Storer.vi subVI in the **Read** state of the state machine.
2. Set up probes on the subVI outputs and inputs.
3. Run the Async Serial VI. The execution of the block diagram halts at the break point. You will see that the probes on the input side of the subVI are executed and the probes on the output side have not been executed. At this point both the Booleans should read false (only one sample has been received).
4. To run the second loop iteration such that the VI execution halts at the same break-point, hit "Continue" on the UART_Receiver.vi sub-VI. You will once again get false values on both probes (only two samples read).

   Again hitting "Continue" would result in the same result on the probes (only three samples read). The third time you hit the "Continue" button, one of the probes should go true since 4 samples have been read. On the fourth until sixth hit, both probes should show false values. Again, one of the values should go true on the seventh hit (8 samples read). If the pattern continues for the next hits, the resettable counter you designed is working fine.

So, the two probes should display:

Probe 1: (False-3x, True-1x), (False-3x, True-1x), (False-3x, True-1x), (False-4x), …
Probe 2: (False-4x), (False-4x), (False-4x), (False-3x, True-1x)…

5. To stop the VI once debugging is over, hit "Stop" button on the UART_Receiver.vi block diagram panel, and then hit the "Pause/Continue" button. To clear the breakpoint, right click on the sub-VI where you imposed the breakpoint, scroll over to "Breakpoint" and then click on "Clear the breakpoint".

   Make sure that you keep previously applied probes in place to follow the logic. The value on the newly applied probe should go true on reception of a byte (after 4*8 = 32 samples).

13.4.3 Debugging the Binary to Decimal Conversion and Byte Storage

To test the Binary to Decimal convertor and byte storer, you may set a breakpoint on the output wire of this sub-VI and follow a similar approach. You should observe the output array that stores decimal (0-255) equivalent of the received byte updated after reception of each byte. The other way you may test it is to set an arbitrary array of decimal values as the previous loop value and a new incoming value, and then test it for both values of the Boolean input. If the Boolean is true, you must see the new incoming value appended to the array of previous loop values, else the output should be the previous loop value itself.

13.5 Lab Write-up

Lab Performance Checklist
UART Communication and Sync

Questions (Please attach your answer).
Ques. 1: Explain in your words the data flow and operations of the UART receiver design you attempted.

Performance Measures

Task	Standards	Satisfactory/ Unsatisfactory
Explanation of the receiver data flow and operations	Quality of description.	
Running VIs	1. Working of sub-Vis. a. Resettable Counter (Bit Detector) b. Resettable Counter (Bit Counter) c. Bit Counter and Storer d. B2D Counter and Byte Storer 2. Overall successful demodulation.	
Connector pane set-up	Successful set-up of appropriate I/P and O/P terminals of sub-VIs	
Debugging tools	Level of understanding of debugging tools	

Discussion
Did you have any difficulty in connector pane set-up and debugging of sub-VIs?
Did all VIs perform as expected?
Did you have any difficulties completing the lab?
Did your TA provide enough guidance?
Do you have any recommendations to improve the lab?

13.6 References

[1] Dr. Sachin Katti, Jeff Mehlman and Aditya Gudipati, Lab 2: UART Communication, Sync, and Channel Correction, Stanford University.
[2] National Instruments, Tutorial: State Machines, retrieved March 20, 2014, from http://www.ni.com/white-paper/7595/en/.
[3] Robert Gallager, "Course materials for 6.450 Principles of Digital Communications I, Fall-2006" retrieved June 7, 2014, from OpenCourseWare (http://ocw.mit.edu/), Massachusetts Institute of Technology.
[4] National Instruments, Tutorial: Passing Data between Loop Iterations in NI LabVIEW, retrieved July 16, 2014, from http://www.ni.com/gettingstarted/labviewbasics/shiftregisters.htm.
[5] National Instruments, Tutorial: Sub-VIs, retrieved July 16, 2014, from http://www.ni.com/white-paper/7593/en/.
[6] National Instruments, Tutorial: Debugging in NI LabVIEW, retrieved July 16, 2014, from http://www.ni.com/gettingstarted/labviewbasics/debug.htm.

14 Binary Phase Shift Keying

14.1 Summary

In this lab, you will encode/decode a text string (the source) using the American Standard Code for Information Interchange (ASCII)[2]. The encoded message is passed through the transmitter part of the Universal Asynchronous Receive/Transmit (UART). The resulting bit stream is transmitted and received using Binary Phase Key Shifting (BPSK) as the modulation technique. The detected bits at the receiver are passed through a UART receiver to reconvert them into the ASCII codes. The learning goal for this lab is to have you build the BPSK modulator and demodulator portion of this communication system.

14.2 Background

14.2.1 Digital Communications Systems

In digital communication systems, the source output is converted into a binary sequence by the source encoder (Fig. 210). This binary sequence is then processed by the channel encoder or modulator, and then transmitted over the channel. The incoming binary sequence is recreated by the channel decoder or demodulator, and the source output is recreated by the source decoder.

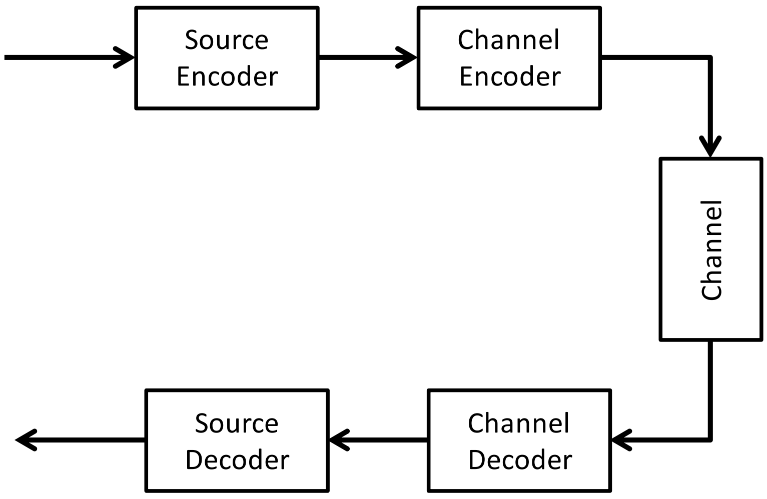


Fig. 210: Digital Communications Model

14.2.2 Binary Phase Shift Keying Modulation Basics

In PSK (Phase Shift Keying), the phase of a carrier is changed between two values according to the binary signal level[3]. The information about the bit stream is contained in the phase changes of the transmitted signal. For instance to transmit the signals “0” and ”1”, PSK signals can be chosen as follows:

$$\phi_0(t) = \begin{cases} A\sin(\omega t + \theta_0), & 0 < t < T \\ 0, & \text{Otherwise} \end{cases} \qquad \phi_1(t) = \begin{cases} A\sin(\omega t + \theta_1), & 0 < t < T \\ 0, & \text{Otherwise} \end{cases} \tag{78}$$

Here, $\theta_0$and θ_1 are constant phase shifts. The values of $\theta_0$and θ_1 can be chosen as desired as long as $\theta_1 - \theta_0 = \pi$. This phase difference of π radians in the PSK signals is a good choice to improve the error-performance the implemented system. Usually the values of 0 and π or $\pi/2$ and $3\pi/2$ are chosen for $\theta_0$and θ_1 to further simplify the modulator design. Therefore, $\phi_0(t)$ and $\phi_1(t)$ waves can be written as follows:

$$\phi_0(t) = \begin{cases} A\cos(\omega t), & 0 < t < T \\ 0, & \text{Otherwise} \end{cases} \qquad \phi_1(t) = -\phi_0(t) \tag{79}$$

$$= \begin{cases} -A\cos(\omega t), & 0 < t < T \\ 0, & \text{Otherwise} \end{cases}$$
$$= \begin{cases} A\cos(\omega t - \pi), & 0 < t < T \\ 0, & \text{Otherwise} \end{cases}$$

As illustrated in Fig. 211, a sinusoidal waveform is modulated (multiplied) by the input bit stream. Each time the bit stream changes sign, the phase of the PSK signal also changes.

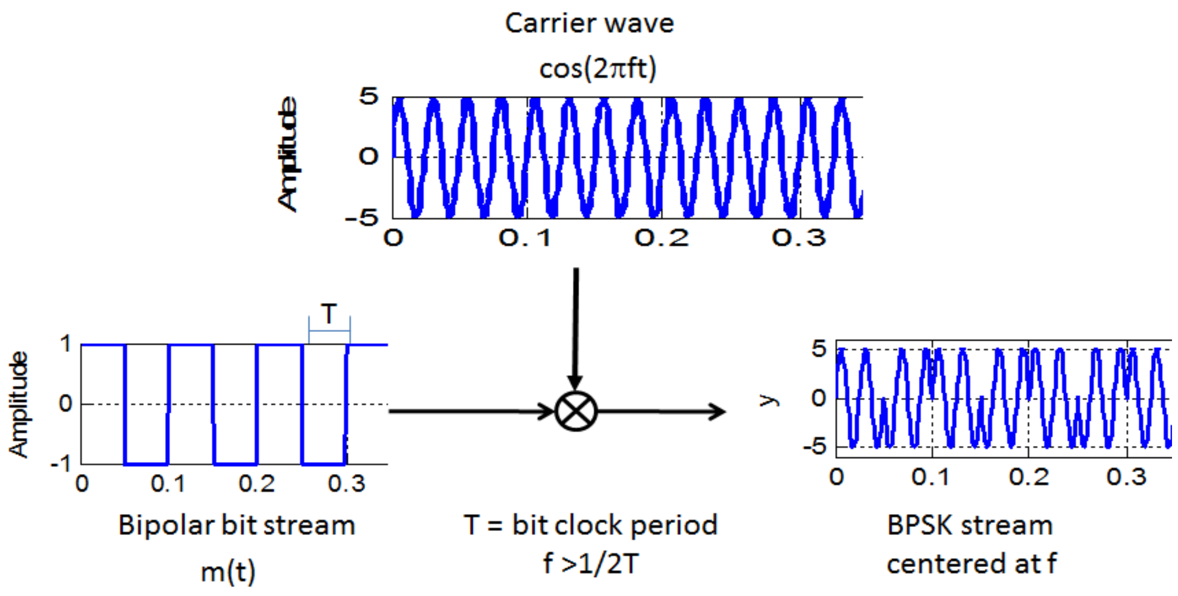


Fig. 211: Generation of BPSK Stream

14.2.3 Binary Phase Key Shifting Demodulation Basics

For BPSK signal demodulation **Error! Reference source not found.**, [4], [5], we will use edge detection to recover the binary signal from the BPSK modulated signal as shown in Fig. 212. For this approach to work, we need to be able to divide the BPSK modulated signal into equal length segments of "N_c" samples based on the length of a complete cycle of the carrier wave (T_c), called a frame. Each frame can be examined to see if the signal is rising or falling at the start of the frame. If the signal is rising, we can conclude that the frame represents a zero or logic "low". Similarly, a signal that is decreasing (falling) at the start of the frame would indicate a one or logic "high". Once the binary signal has been recovered in this manner, it can be passed through the UART receiver you developed in the UART lab to obtain the original text message.

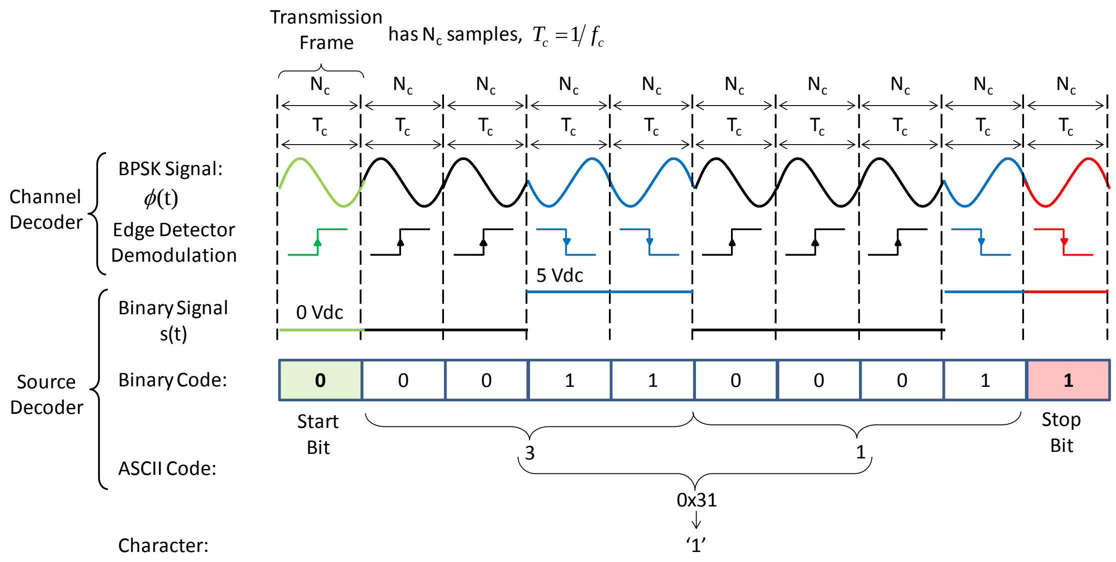


Fig. 212: BPSK Signal to Binary Output Signal

The overall data flow and operations in this lab are summarized in Fig. 213.

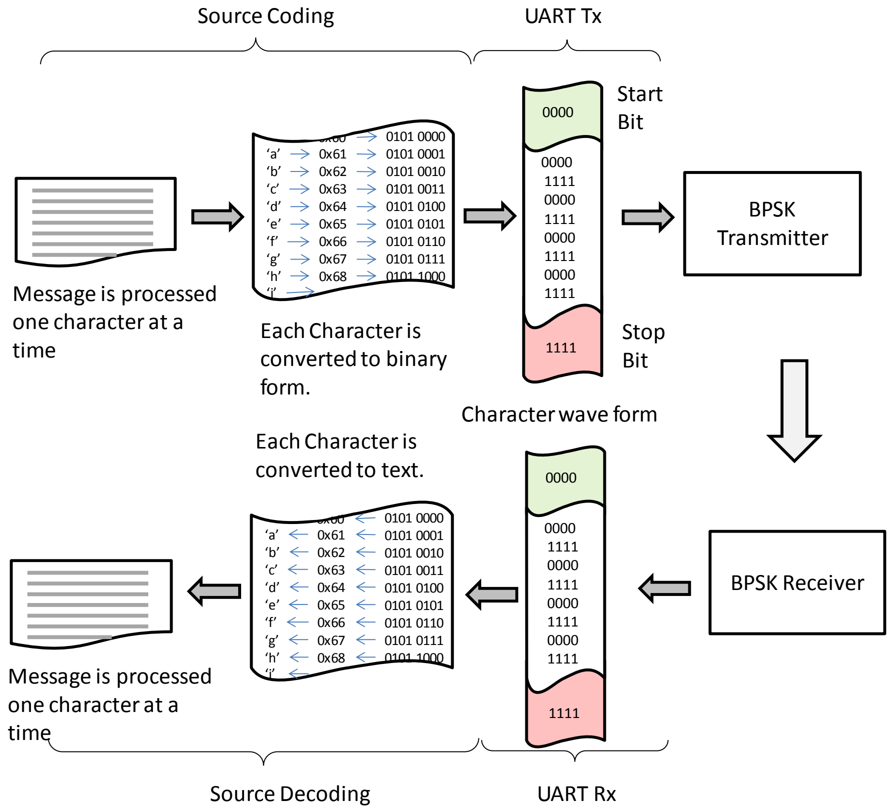


Fig. 213: Data flow diagram of BPSK modulation/demodulation of UART pulses

14.3 Pre-Lab

Your pre-lab task is to build a functional BPSK modulator using the template (*BPSK_Modulator_Template.vi)* provided.

14.3.1 BPSK Modulator

The BPSK modulator first performs antipodal voltage encoding on the UART transmitter encoded bit string of high and low voltages. The UART transmitter output is modified to map a HIGH bit as the having the value of 1, and a LOW bit as a -1 value. This will result in mapping the bit values onto the phase shifts ($\theta_0 = 0$ and $\theta_1 = \pi$, respecively).

Mapping Symbols -Design the "PSK map symbols" VI to establish this mapping by following the undermentioned procedure.

a) Use a "for" loop to read an element from the message signal per loop iteration, and test the incoming element to see if it is a voltage HIGH bit (5V DC) or not.
b) Initialize an array with dimension size equal to the size of the incoming bit-stream array, and set all its values to be "-1".
c) Make a "Case Structure" inside the "for" loop. Associate the "True" case to reception of a voltage HIGH. To establish this, use the "Replace Array Subset" block initialized by the array you built in previous step such that the default value of -1 is changed to 1 at the corresponding index. Use shift registers to update this array with each loop iteration and output the updated bit-stream.

   The resulting mapped symbols bit-stream is modulated with a carrier sine wave so as to get the desired BPSK signal. For this you have to make sure each carrier wave cycle is multiplied by the appropriate bit value (1 or -1) of the mapped symbols bit stream which means you will have to add samples to each mapped symbol bits to make the number of samples in both pulses equal. For example, if you want to use a sine carrier with frequency of 1000 Hz, you should have 1000 samples for each bit value. So you need an additional 249 copies (1000/4 − 1) of each mapped sample. This resampling allows you to use a simple multiplication of the carrier by the bit value (1 or -1) for all carrier samples in a given cycle.

BPSK Modulator - Build a functional "BPSK Modulator" by following the undermentioned steps.

a) Use "Resampler.vi**"** to insert the requisite number of copies of each symbol in the mapped symbols bit-stream.
b) Use "Bundle by name" control block to bundle the resampled bit stream with the sampling time interval.
c) Use "Carrier Sine Wave Generator.vi" to generate the carrier sine wave.
d) Modulate the carrier with resampled bit stream to get the BPSK modulated waveform.
e) For your understanding of the modulation, use "Modified_Signal_Merge.vi" to compare the rescaled incoming UART bit-stream with the BPSK modulated UART waveform.

The working of the BPSK modulator is summarized in Fig. 214.

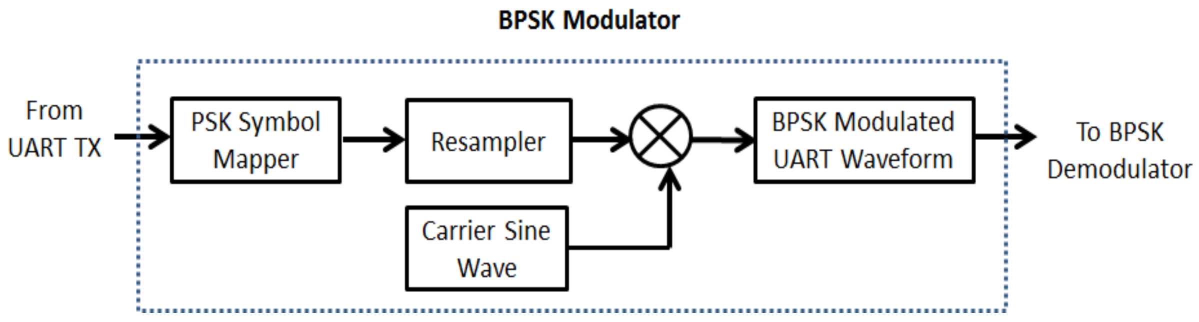


Fig. 214: BPSK Modulator Block Diagram

14.4 Lab Procedure

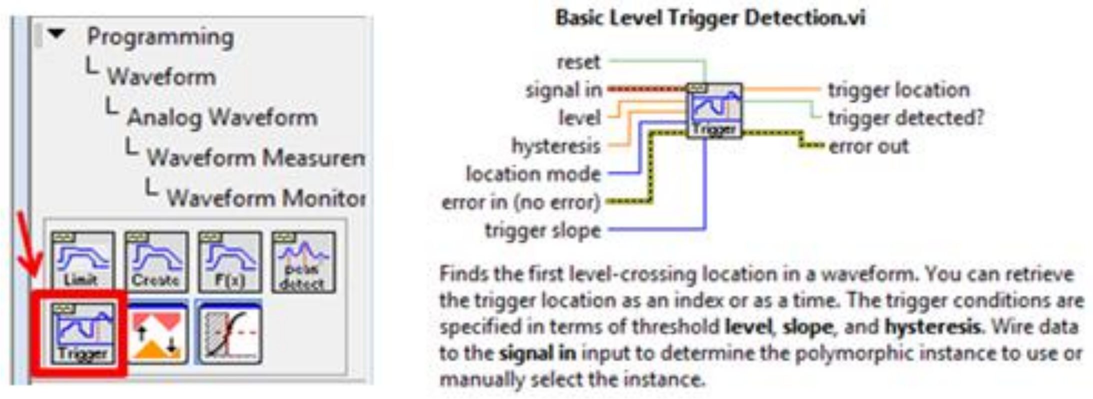
In the lab, you have to build a functional BPSK demodulator using the template provided (*BPSK_Demodulator_Template.vi)*, and then test for the working of the overall process outlined in Fig. 213.

14.4.1 BPSK Demodulator

The BPSK Demodulator uses a simple decision rule to find the value of each received bit. If the bit samples have an increasing (+) slope it is a “zero”, otherwise it is a “one”. The slope is tested using the “Basic Level Trigger Detection” VI with a “falling edge” or “negative slope” trigger detection criteria. The demodulated bits are then stored in an array and resampled so as to get the UART pulses.

BPSK Demodulator Build a functional “BPSK Demodulator” by following the undermentioned steps**.**

1. Use “Get waveform components” VI to get the data values for samples of the BPSK modulated UART waveform.
2. Set up a “for” loop to process the number of bits in the transmission, i.e. the total number of samples divided by the sampling frequency.
3. Use an array subset VI by setting the “index” input to the product of the current loop iteration with the sampling frequency of the carrier, and the subset’s length equal to the sampling frequency of the carrier.
4. Input the array subset to the “Basic Level Trigger Detection” VI to detect the index at which the trigger is detected. Set the location mode as “Index” and trigger slope as “Falling Edge”. This VI setting would output the first level crossing trigger location i.e. the array index at which the trigger is detected. Since you are using the “Falling Edge” as the trigger slope, you will get the trigger location as the mid-point of the array (Index of Sampling Frequency/2 = Index 500) for a HIGH bit, and the start of the array (Index 0) for a LOW bit (Fig. 212).



Finds the first level-crossing location in a waveform. You can retrieve the trigger location as an index or as a time. The trigger conditions are specified in terms of threshold **level**, **slope**, and **hysteresis**. Wire data to the **signal in** input to determine the polymorphic instance to use or manually select the instance.

Fig. 215: Basic Level Trigger Detection VI

5. Compare the trigger location with the index of mid-point of the array (Sampling Frequency/2 = 500). Scale the received Boolean by 5 to get the voltage of the UART waveform (High bits: 5 volts, Low bits: 0 V).
6. Using a “Build array” block initiated to a zero constant, store the scaled voltages in an array. Use shift registers to update this array with each loop iteration.
7. Once all bits are demodulated, resample the bit array by a factor of number of samples per bit to get the BPSK demodulated UART pulses.

14.4.2 Testing it all together

Open and run “Lab13.vi”. Make sure you name all the sub-VIs same as that of those put up in the Lab13.vi, or replace the sub-Vis in Lab13.vi with the ones you designed. If you built all the sub-VIs in line with the instructions in the pre-lab and lab procedure, you should be able to see the message encoded by the UART ASCII transmitter. Use debugging tools to track the data flow and operations in case you don’t get the desired output.

14.5 Lab Write-up

Performance Checklist

Binary Phase Shift Keying

Questions

1. Read out the byte in the BPSK modulated waveform in Fig. 216.

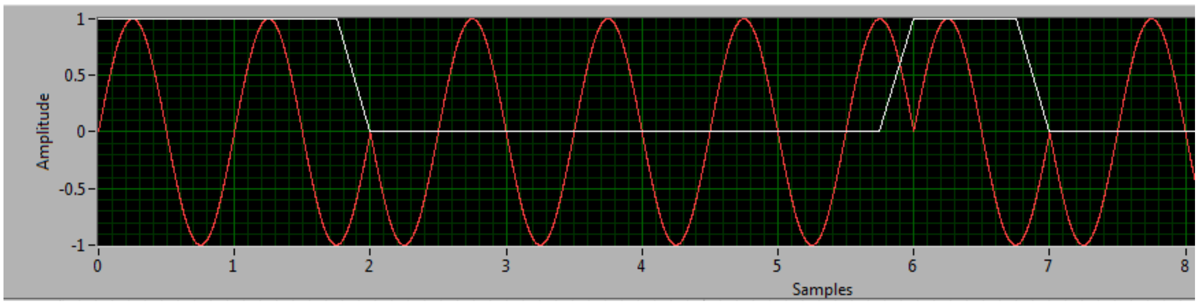


Fig. 216: Sample BPSK modulated waveform

2. Suppose you are asked to implement the "Basic Level Trigger Detection" VI using the "Rising Edge" trigger slope. What changes you would make in the BPSK Demodulator VI and why?

3. Explain your approach in case you didn't want to use the resampling VI for this lab?

4. Summarize what you learnt about BPSK modulation and demodulation in your words.

Performance Measures

Task	Standards	Satisfactory/ Unsatisfactory
Short answer questions	Quality of description. 1. 2. 3. 4.	
Design of VIs	Functionality achieved: 1. PSK Mapper (Yes/No) 2. BPSK Modulator (Yes/No) 3. BPSK Demodulator (Yes/No)	

Discussion

Did all VIs perform as expected?
Did you have any difficulties completing the lab?
Did your TA provide enough guidance?
Do you have any recommendations to improve the lab?

14.6 References

[1] Robert Gallager, "Course materials for 6.450 Principles of Digital Communications I, Fall-2006" retrieved June 7, 2014, from OpenCourseWare (http://ocw.mit.edu/), Massachusetts Institute of Technology.
[2] Dr. Sachin Katti, Jeff Mehlman and Aditya Gudipati, Lab 2: UART Communication, Sync, and Channel Correction, Stanford University.
[3] Dr. Sachin Katti, Jeff Mehlman and Aditya Gudipati, Lab 3: Modulation, Stanford University.
[4] Dr. Sachin Katti, Jeff Mehlman and Aditya Gudipati, Lab 4: Demodulation, Stanford University.
[5] Dogus University, Faculty of Engineering, Experiment 5: Binary Phase Shift Keying (BPSK), retrieved June 7, 2014, from http://www3.dogus.edu.tr/ehmb/en/Labs/AssignmentsPDF/DigiCom/CEE312_Exp5.pdf.

15 Appendix A: LabVIEW Tutorial

The basic structure of a LabVIEW block diagram is shown in the following figure. The simplest way to think of a block diagram is as a network of components (sub-VIs) and their connections. Each node can have up to three kinds of connections. The Configuration Data connection passes the data used to configure the system in the source node. The second connection, Status/Error, passes data about the success of the sub-VIs execution. The final type of connection allows for the passing of data between VIs. The components or sub-Vis are either created by you, or are provided by the LabVIEW environment or the lab instructor.

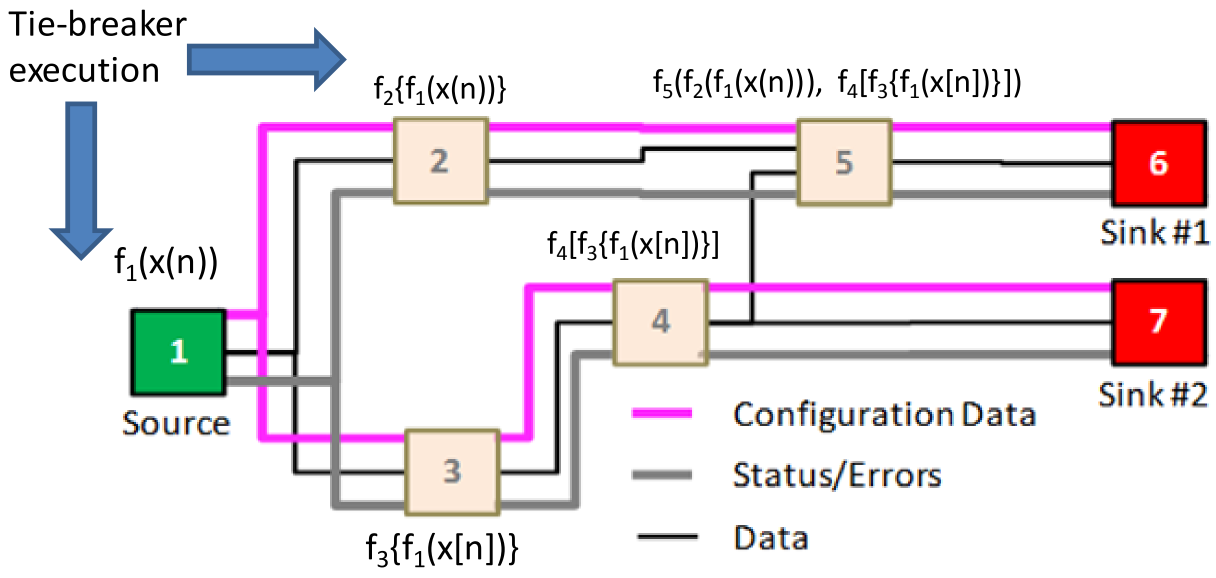


Fig. 217: Block Diagram Structure and Execution

It is important to note when debugging a block diagram, that the diagram's execution begins at the upper left corner and proceeds left to right. If two blocks are in parallel (like blocks 2 and 3), the diagram executes from top to bottom and then continues left to right. The blocks in the figure are labeled or numbered in the sequence they would execute. Another helpful thing to know when debugging a diagram is that you can attach a probe to any of the connections. This probe is like using an oscilloscope to observe the signal on the line. The main difference is that instead of a trace you will be able to see the actual data.

By now you might be wondering, *why bother with a front panel*. The front panel allows you to control the execution of the block diagram by providing switches, knobs, and data entry sources that when combined with a looping or conditional structure can reconfigure the block diagram for different tasks. The front panel switches and buttons are sub-VIs in the diagram shown above. Similarly, there are display sub-VIs that allow for the data to be displayed in numerical format or as traces on a graph. These graphs are similar to the traces you see on an oscilloscope.

The use of a network composed of sub-VIs supports two important features of LabVIEW: code reuse and modularity. The graphical nature of LabVIEW provides another virtue: it allows developers to easily visualize the flow of data in their designs. NI calls this Graphical System Design. Also, since LabVIEW is a mature data flow programming language, it has a wealth of existing documentation, toolkits, and examples which can be leveraged in development.

In this course you will use National Instruments USRP hardware. LabVIEW provides a simple interface for configuring and operating various external I/O, including the NI USRP hardware used in this lab. This is the main reason why you will use LabVIEW as the programming language to build an SDR in this course. The choice of hardware and software in this lab is mostly a matter of convenience. In future labs you will need to be familiar with LabVIEW and the documentation/help available to you. This is the only lab in this course which will give you the opportunity to learn and practice LabVIEW programming; so it is important that you take this opportunity to ask the instructor any questions you might have. The following tutorials and reference materials will help guide you through the process of learning LabVIEW:

- *LabVIEW 101* [5]
- *LabVIEW Fundamentals* from National Instruments [6];
- Online LabVIEW tutorials from NI [3], [4].

New LabVIEW programmers should carefully review all of the material in [5] and [4]. Please remember to refer to [6] and [4] often as they are excellent references for all basic LabVIEW questions. In order to test your familiarity with LabVIEW, you will be asked at the end of this lab to build some simple VIs in LabVIEW and integrate them into a simulator. Bring any questions or concerns regarding LabVIEW or these tutorials to your instructor’s attention. For the remainder of this lab you should be familiar with the basics of LabVIEW programming and where to look for help.

16 Appendix B: Communications Palettes

Many of the algorithms implemented in this lab (and in digital communications in general) use linear algebra. LabVIEW provides support for matrix and vector manipulation, and linear algebra, with VIs for functions like matrix inversion (*Inverse Matrix.vi*), matrix multiplication (*A x B.vi*), and reshaping arrays (*Reshape Array.vi*). LabVIEW also has many built-in signal processing functions, such as the fast Fourier transform (*FFT.vi*), inverse fast Fourier transform (*Inverse FFT.vi*), and convolution (*Convolution.vi*). Additionally, the Modulation Toolkit is a toolset of common digital communication algorithms which will also be leveraged in the lab. Note that many of the functions you will implement in this lab are already available in the Modulation Toolkit in some form. The objective of this course is to understand the principles of wireless digital communication by implementing the physical layer in as much detail as possible. Once familiar with these concepts, you will be able to decide when to use existing VIs and when to write your own. It is recommended that you explore tool palettes such as the (1) Signal Processing palette, (2) Digital palette (part of the Modulation Toolkit palette), (3) Structures palette, (4) Complex palette (part of the Numeric palette), and (5) Array palette in order to acquaint yourself the VIs likely to be used in this course.

Appendix B enumerates some common VIs and highlights how to access them.

Transmitter

LabVIEW interacts with USRP transmitter by means of six VIs located in Instrument I/O tool palette (Fig. 8.).

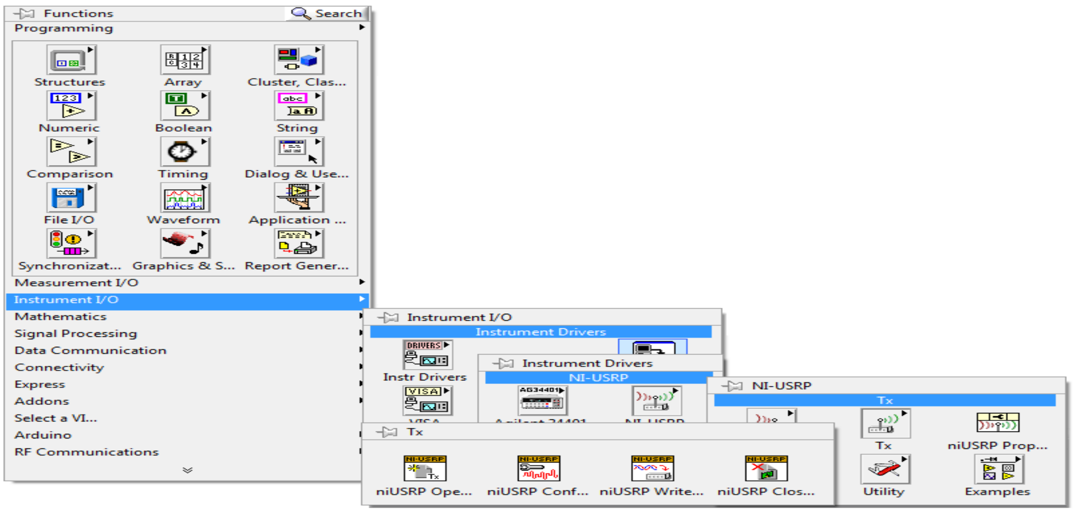


Fig. 218: Transmitter sub-VI Selection Tool Palette Error! Reference source not found.

Fig. 219: Basic Transmitter structure

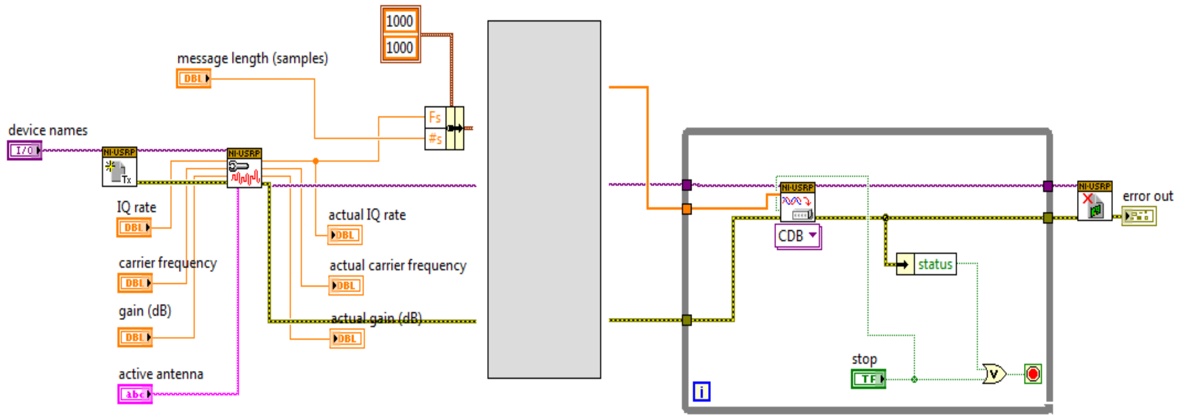


Fig. 220: Transmitter Template

Open Tx Session.vi – This sub-VI initiates the transmitter session and generates a session handle and an error cluster that are propagated through all VIs. When you use this sub-VI, you must add a control called "device names" that you will use to inform LabVIEW of the IP address of the USRP (192.168.10.2).

Configure Signal.vi – This sub-VI is used to set parameter values in the radio. Attach four controls and three indicators to this sub-VI as shown in Fig. 11. To get started, set the IQ rate to 200 kS/s (the lowest possible rate), the carrier frequency to 915.1 MHz, the gain to 20 dB, and the active antenna to TX1. When the VI runs, the radio will return the actual values of these parameters. These values will be displayed by the indicators you connected. Normally the actual parameter values will match the desired values, but if one or more of the desired values is outside the capability of the radio, the radio will choose the nearest acceptable parameter value, rather than return an error.

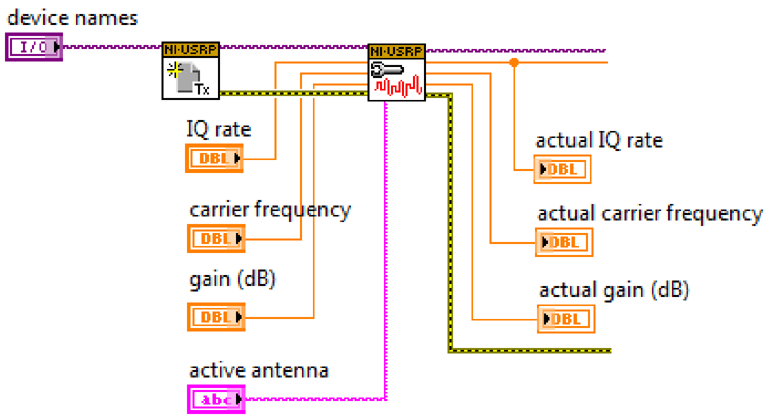


Fig. 221: Transmitter Setup

Write Tx Data.vi - This sub-VI is writes the baseband signal to the USRP for transmission. Placing this sub-VI in a *while* loop (Fig. 12.) allows a block of baseband signal samples to be sent over and over until the “stop” button is pressed. Note that the *while* loop will also terminate if an error is detected. Baseband signal samples can be provided to the **Write Tx Data** sub-VI as either an array of complex numbers or as a complex waveform data type. A pull-down tab allows you to choose the data type. We will explore how the baseband signal is generated in subsequent lab projects that use both components.

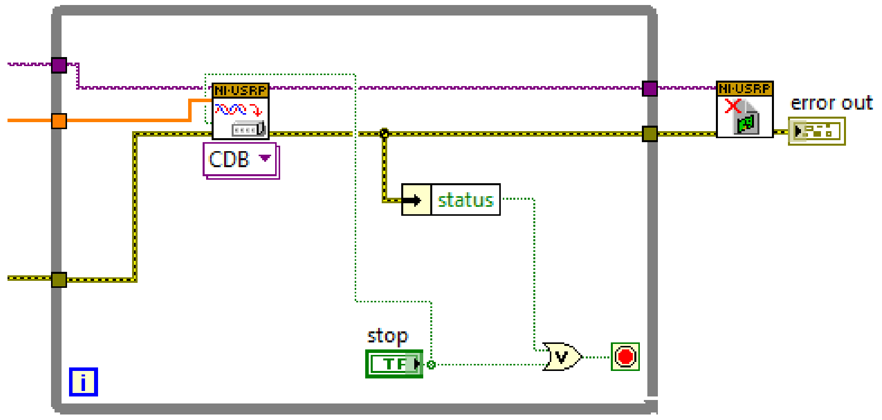


Fig. 222: Data Transmission

Close Session.vi - This sub-VI terminates transmitter operation once the *while* loop ends. Note that the sub-VI should be terminated using the ***STOP*** *button* rather than with “Abort Execution” on the toolbar. This is so that the Close Session VI will be sure to run and will correctly close out the data structures that the VI uses.

Receiver
LabVIEW interacts with the USRP receiver by means of six VIs located in Instrument I/O tool palette as shown in Fig. 13.

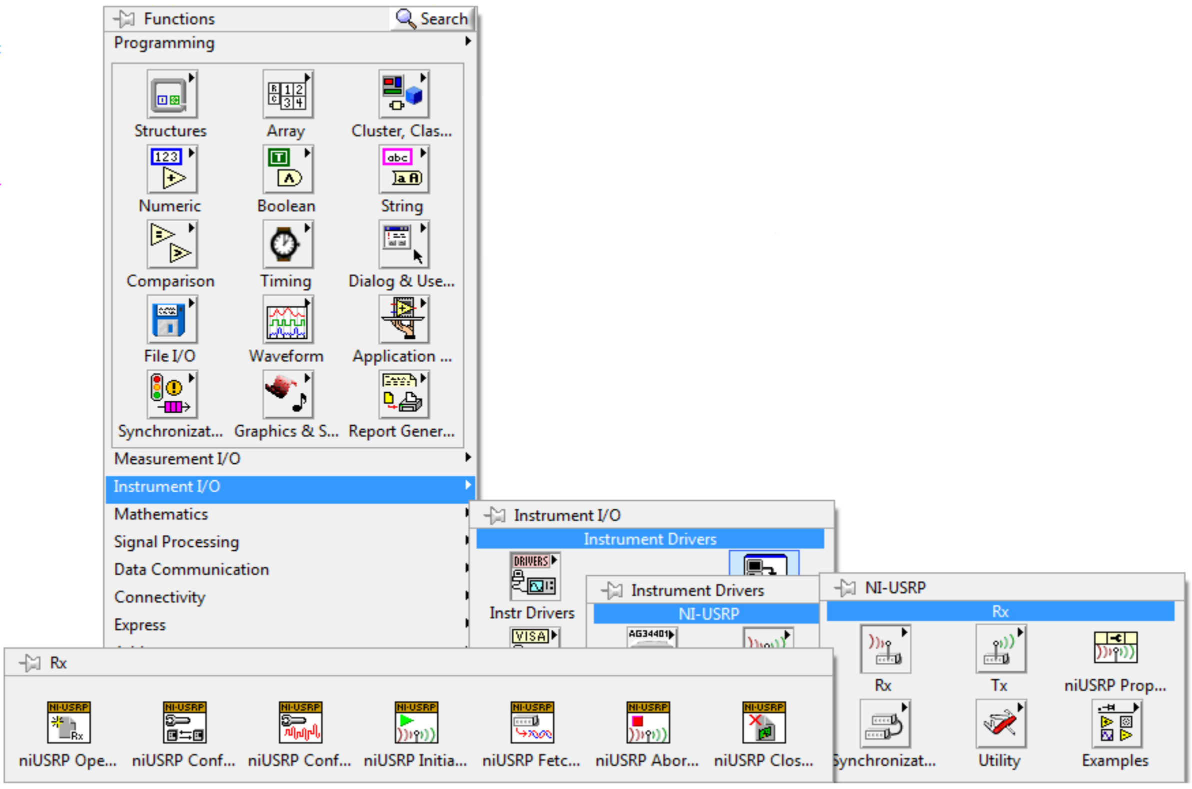


Fig. 223: Reciever sub-VI Selection Tool Palette

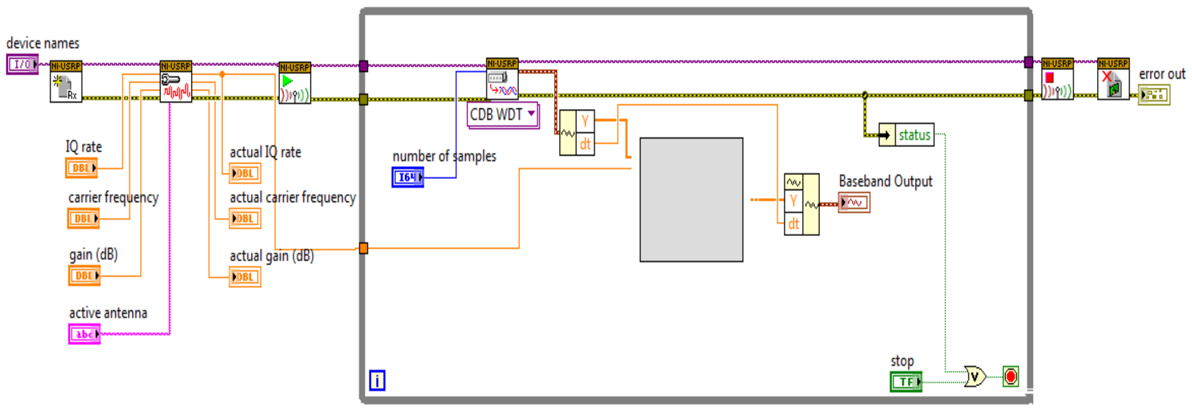


Fig. 224: Receiver Template

Open Rx Session.vi - This sub-VI initiates the receiver session and generates a session handle and an error cluster that are propagated through all six VIs. You must add a control called "device names" that you will use to inform LabVIEW of the IP address of the USRP (192.168.10.2).

Configure Signal.vi - This sub-VI has the same function as the corresponding VI in the transmitter. Attach four controls and three indicators to this VI as shown in the figure. This time, set the IQ rate to 1 MS/s, the carrier frequency to 915.0 MHz, the gain to 0 dB, and the active antenna to RX2. When the VI runs, the radio will return the actual values of these parameters.

Initiate.vi - This sub-VI sends the parameter values you selected to the receiver and starts it running.

Fetch Rx Data.vi - This sub-VI retrieves the message samples received by the USRP. Placing this VI in a *while* loop allows message samples to be retrieved one block at a time until the "STOP" button is pressed. Note that the *while* loop will also terminate if an error is detected. A "number of samples" control allows you to set the number of samples that will be retrieved with each pass through the *while* loop. Message samples can be provided to the user as either an array of complex numbers or as a complex waveform data type. A pull-down tab allows you to choose the data type for the message samples.

Abort.vi - This sub-VI stops the acquisition of data once the *while* loop ends.

Close Session.vi: This sub-VI terminates receiver operation. As noted above, use the STOP button to terminate execution so that Close Session will be sure to run.

17 Appendix C: Cantenna Specifications

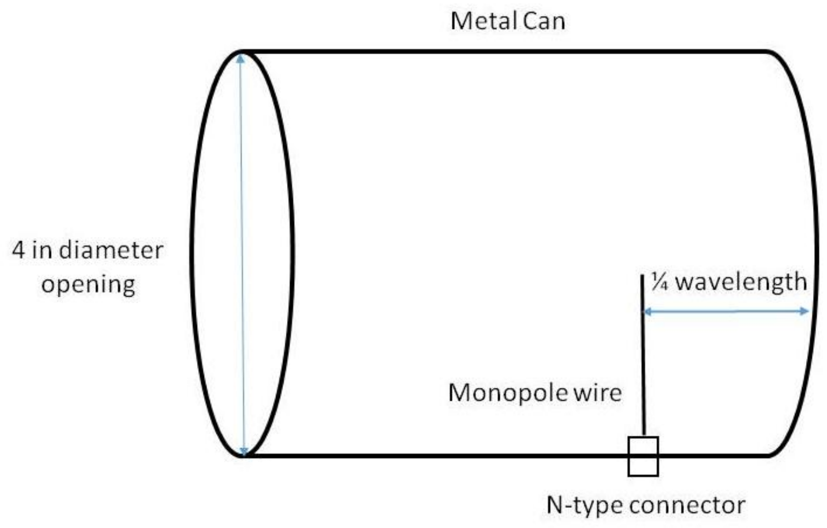
Cantenna Design - 1400 Mhz

Parts

N-type chassis mount connector
BNC cable (at least one foot)
BNC (female) to SMA (male) adapter
BNC (female) to N (male) adapter
18 gauge AWG copper wire (at least 3 in.)
Can (4in diameter, at least 4.5 in long)
Bolts (4 ea.), if N-type chassis mount requires them

Instructions

1. Cut copper wire to 1/4 wavelength (2.1 in.) to monopole antenna.
2. Drill a hole in can 1/4 wavelength (2.1 in.) from the back of can, and large enough to fit N-type connector.
3. Solder copper wire to N-type connector.
4. Insert N-type connector through hole and attach and tighten nut or bolts.
5. Attach N and RPM-SMA adapters to BNC cable.
6. Attach BNC cable to N-type connector.



$$\frac{1}{4}\, wavelength\ (meas.\, in\ feet) = \frac{246}{1400\ Mhz\ (center\ freq.\ in\ MHz)} = .176\ ft = 2.1\ in.$$

Finished Product

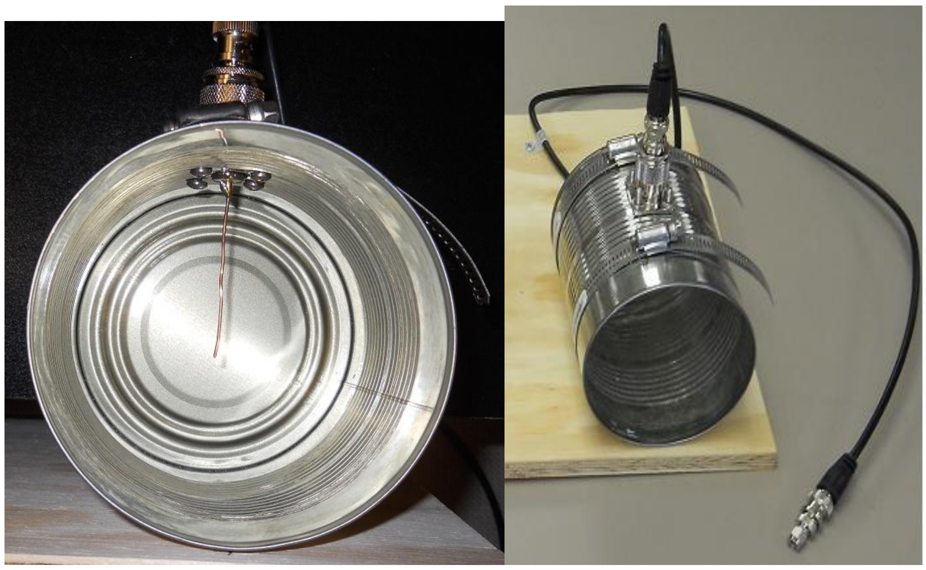


Fig. 225: Assembled Cantenna