Une Introduction à la Radio Définie par Logiciel

avec NI LabVIEW et NI USRP

Version 1.1 – Q4 2013

NATIONAL INSTRUMENTS™

Worldwide Technical Support and Product Information
National Instruments Corporate Headquarters
Worldwide Offices

Important Information

Copyright

End-User License Agreements and Third-Party Legal Notices

You can find end-user license agreements (EULAs) and third-party legal notices in the following locations:

- Notices are located in the `<National Instruments>\_Legal Information` and `<National Instruments>` directories.
- EULAs are located in the `<National Instruments>\Shared\MDF\Legal\License` directory.
- Review `<National Instruments>\_Legal Information.txt` for more information on including legal information in installers built with NI products.

Trademarks

Refer to the *NI Trademarks and Logo Guidelines* at `ni.com/trademarks` for more information on National Instruments trademarks.

ARM, Keil, and µVision are trademarks or registered of ARM Ltd or its subsidiaries.

LEGO, the LEGO logo, WEDO, and MINDSTORMS are trademarks of the LEGO Group. ©2013 The LEGO Group.

TETRIX by Pitsco is a trademark of Pitsco, Inc.©2013

FIELDBUS FOUNDATION™ and FOUNDATION™ are trademarks of the Fieldbus Foundation.

EtherCAT® is a registered trademark of and licensed by Beckhoff Automation GmbH.

CANopen® is a registered Community Trademark of CAN in Automation e.V.

DeviceNet™ and EtherNet/IP™ are trademarks of ODVA.

Go!, SensorDAQ, and Vernier are registered trademarks of Vernier Software & Technology. Vernier Software & Technology and vernier.com are trademarks or trade dress.

Xilinx is the registered trademark of Xilinx, Inc.

Taptite and Trilobular are registered trademarks of Research Engineering & Manufacturing Inc.

FireWire® is the registered trademark of Apple Inc.

Linux® is the registered trademark of Linus Torvalds in the U.S. and other countries.

Handle Graphics®, MATLAB®, Real-Time Workshop®, Simulink®, Stateflow®, and xPC TargetBox® are registered trademarks, and TargetBox™ and Target Language Compiler™ are trademarks of The MathWorks, Inc.

Tektronix®, Tek, and Tektronix, Enabling Technology are registered trademarks of Tektronix, Inc.

The Bluetooth® word mark is a registered trademark owned by the Bluetooth SIG, Inc.

The ExpressCard™ word mark and logos are owned by PCMCIA and any use of such marks by National Instruments is under license.

The mark LabWindows is used under a license from Microsoft Corporation. Windows is a registered trademark of Microsoft Corporation in the United States and other countries.

Other product and company names mentioned herein are trademarks or trade names of their respective companies.

Members of the National Instruments Alliance Partner Program are business entities independent from National Instruments and have no agency, partnership, or joint-venture relationship with National Instruments.

Patents

For patents covering National Instruments products/technology, refer to the appropriate location: **Help»Patents** in your software, the `patents.txt` file on your media, or the *National Instruments Patent Notice* at `ni.com/patents`.

Contents

Overview

Purpose of the Hands-on Seminar

This hands-on seminar seeks to educate researchers and graduate students in the fields of science and engineering on the fundamental concepts of wireless signal acquisition. You will physically setup and plug in a USRP, write a program in LabVIEW, and then visualize the results.

The exercises are examples of how to acquire, analyze, and present data for applications involving wireless signals. While the exercise is generic, we hope that you can apply what you learn to your own applications. For future reference, this manual is available both electronically and in print.

What You Will Do

Through three main exercises that contain sub-sections, you will use LabVIEW and the NI USRP 2920 to create a FM radio receiver. There are also supplementary exercises that teach you how to create a custom LabVIEW user interface, how to configure the USRP, and a glossary for the common terms used in this document.

The presenter will start with a quick presentation that defines some common terms and describes how National Instruments hardware and software approach the task.

Why You Should Take This Course

You should take this course if you:

- are researching wireless communications
- need to quickly and easily prototype and validate algorithms
- looking to evaluate the usefulness of software defined radio in your research
- need exposure to setting up software defined radio

Time Required to Complete Course

The course should take approximately three hours, but this time can vary depending on your background.

Required Background

It is recommended that you have some exposure to LabVIEW, but it is not required. The instructions for the exercises will cover all necessary steps to complete the task. It should be noted that you will be expected to learn basic tasks as you progress, and the instructions will become less detailed and require that you retain some of the knowledge. Visit ni.com/academic/students/learn-labview to learn more.

You will be expected to be familiar with using a computer, a mouse, a keyboard, and basic wireless communication concepts.

Required Equipment

Hardware

- NI USRP-2920 device
- NI USRP universal power adapter
- Gigabit (Gb) Ethernet cable
- Antenna
- Computer with a spare Gigabit (Gb) Ethernet port

Software

- NI LabVIEW 2012 or later
- NI USRP 1.1 or later
- NI Modulation Toolkit
- LabVIEW MathScriptRT Add On

Configuring Hardware

The following steps are an overview of the *NI USRP-292x Getting Started Guide*. Follow these steps after you install LabVIEW on your computer:

1. Install the NI USRP Software Suite DVD. The software suite adds the following items to your LabVIEW installation: the NI-USRP driver, LabVIEW Modulation Toolkit, LabVIEW MathScriptRT module, and LabVIEW Digital Filter Design Toolkit. For more information, refer to the Installing the Software section of this document.
2. Referring to Figure 1, attach the antenna to the front of the NI USRP-292x.
3. Connect the device directly to your computer with the enclosed Ethernet cable and plug in the power. For more information, refer to the Installing and Configuring the Hardware section of this document.
4. Change the IP address of your computer's Ethernet port to a static IP. NI recommends a static IP address of 192.168.10.1 because NI USRP-2920 has a default IP address of 192.168.10.2. For more information, refer to the Installing and Configuring the Hardware section (page 8).
5. Examples are located in LabVIEW. Navigate to the start menu and select **Start » All Programs » National Instruments » NI-USRP » Examples**. For more information, refer to the Programming the NI 292x section of this document on page 11.
6. If you have trouble with any of these steps, refer to the Appendix of this document or watch the video at ni.com/usrp/gettingstarted

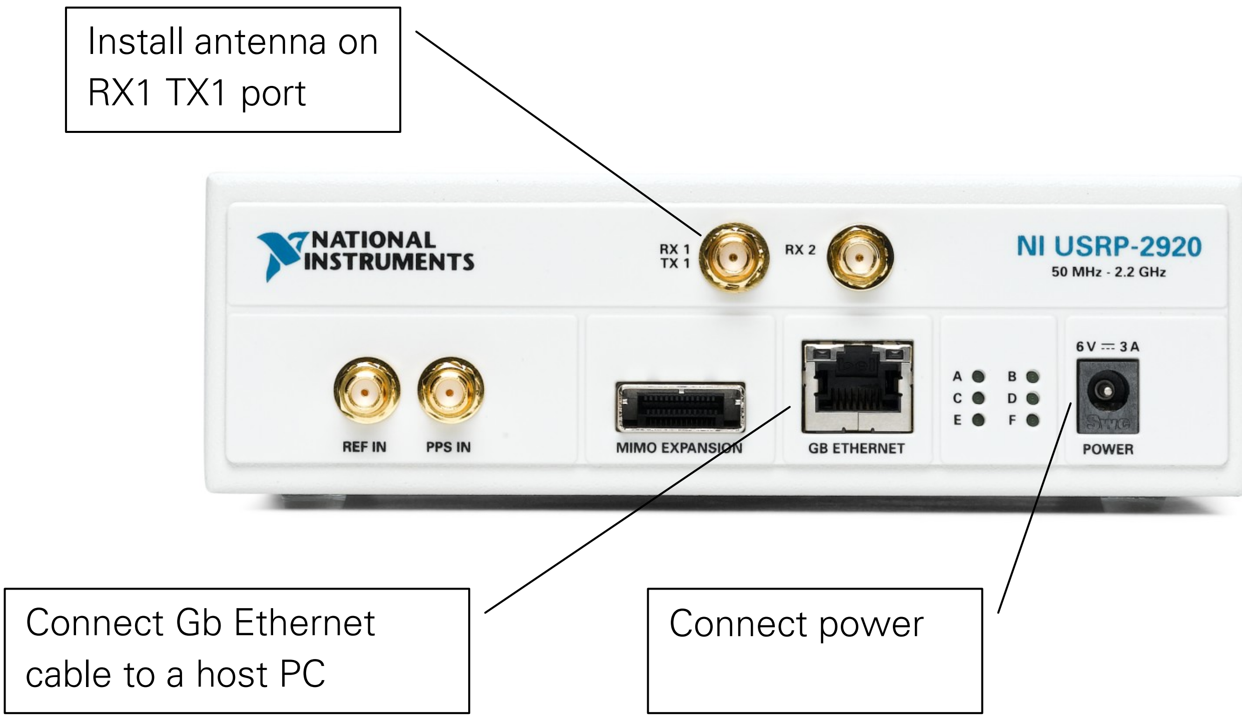


Figure 1. USRP Front Panel

Software Defined Radio Fundamentals

This section reviews the fundamental concepts of software defined radio.

What is Software Defined Radio?

The Wireless Innovation Forum defines Software Defined Radio (SDR) as:

"Radio in which some or all of the physical layer functions are software defined." [1]

SDR refers to the technology wherein software modules running on a generic hardware platform are used to implement radio functions. Combine the NI USRP hardware with LabVIEW software for the flexibility and functionality to deliver a platform for rapid prototyping involving physical layer design, wireless signal record & playback, signal intelligence, algorithm validation, and more.

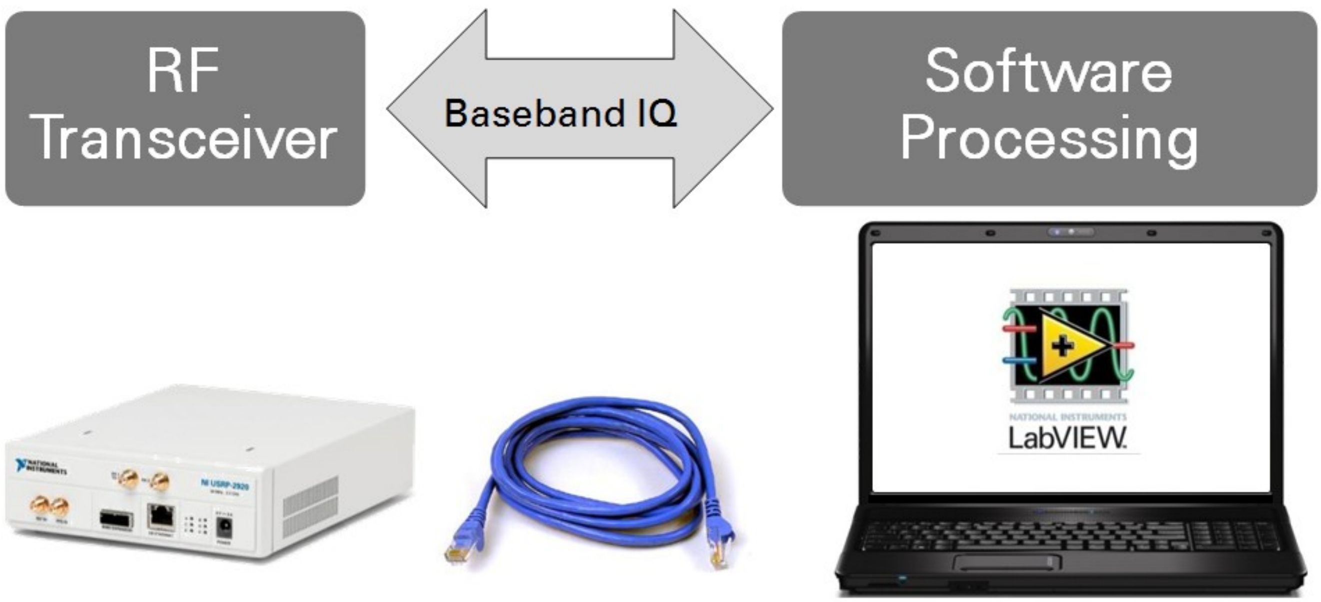


Figure 2. Simplified Overview of a SDR Setup Built Around an NI USRP

[1] http://www.sdrforum.org/pages/documentLibrary/documents/SDRF-06-R-0011-V1_0_0.pdf

Digital Communication System Fundamentals

A typical digital communication system includes a transmitter, a receiver, and a communication channel. Figure 3 illustrates the general components of a digital communication system. The transmitter, shown as blocks on the top row, contains blocks for source and channel coding, modulation, simulating real-world signal impairments, and up conversion. The receiver, which includes the blocks in the bottom row, has blocks for down conversion, matched filtering, equalization, demodulation, channel decoding and source decoding. Refer to the NI LabVIEW Modulation Toolkit online Help for more information about measurement and visualization tools for digital communication systems.

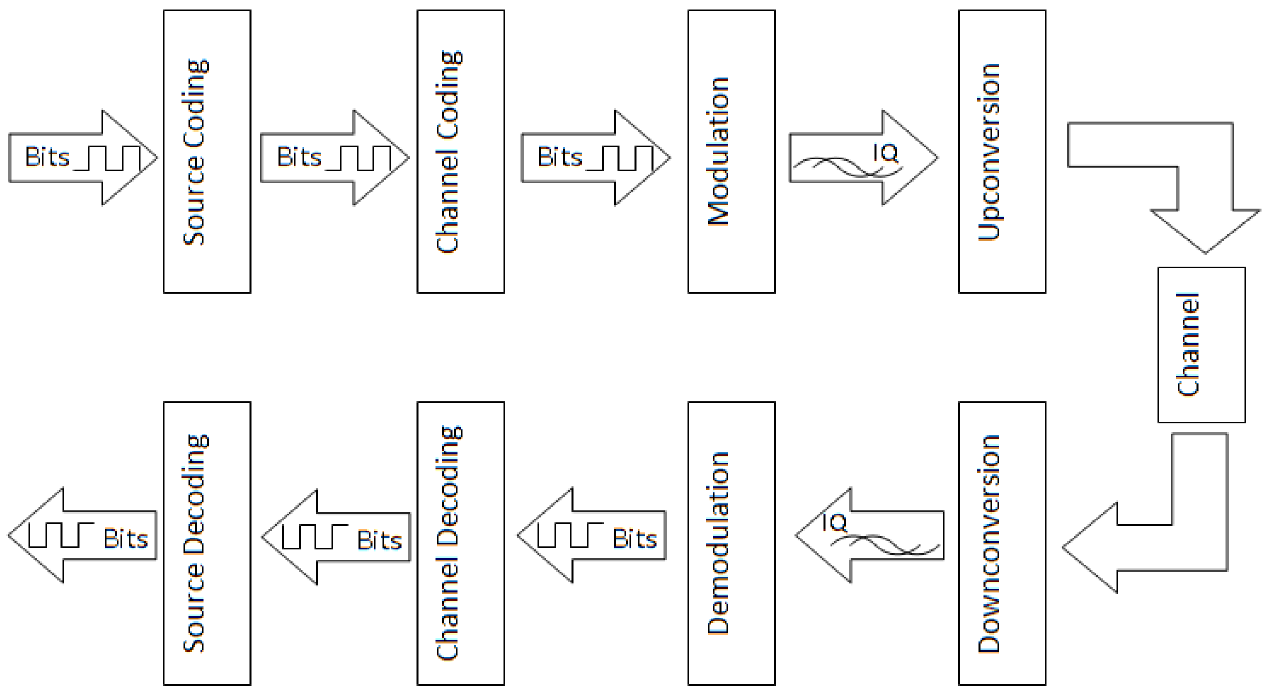


Figure 3. Digital Communication System Block Diagram

USRP Hardware

The NI USRP connects to a host PC to act as a software-defined radio. Incoming signals attached to the standard SMA connector are mixed down using a direct-conversion receiver (DCR) to baseband I/Q components, which are sampled by a 2-channel, 100 MS/s, 14-bit analog-to-digital converter (ADC). The digitized I/Q data follows parallel paths through a digital downconversion (DDC) process that mixes, filters, and decimates the input 100 MS/s signal to a user-specified rate. The downconverted samples, when represented as 32-bit numbers (16 bits each for I and Q), are passed to the host computer at up to 20 MS/s over a standard Gigabit Ethernet connection.

For transmission, baseband I/Q signal samples are synthesized by the host computer and fed to the USRP-2920 at up to 20 MS/s over Gigabit Ethernet when represented with 32-bits (16-bits each for the I and Q components). The USRP hardware interpolates the incoming signal to 400 MS/s using a digital upconversion (DUC) process and then converts the signal to analog with a dual-channel, 16-bit digital-to-analog converter (DAC). The resulting analog signal is then mixed up to the specified carrier frequency.

An available 8-bit mode, in which 16-bits total are used to represent the I and Q values of a downconverted sample or sample to be upconverted, can enable a transfer rate of up to 40 MS/s over the Gigabit Ethernet connection between the host PC and the USRP.

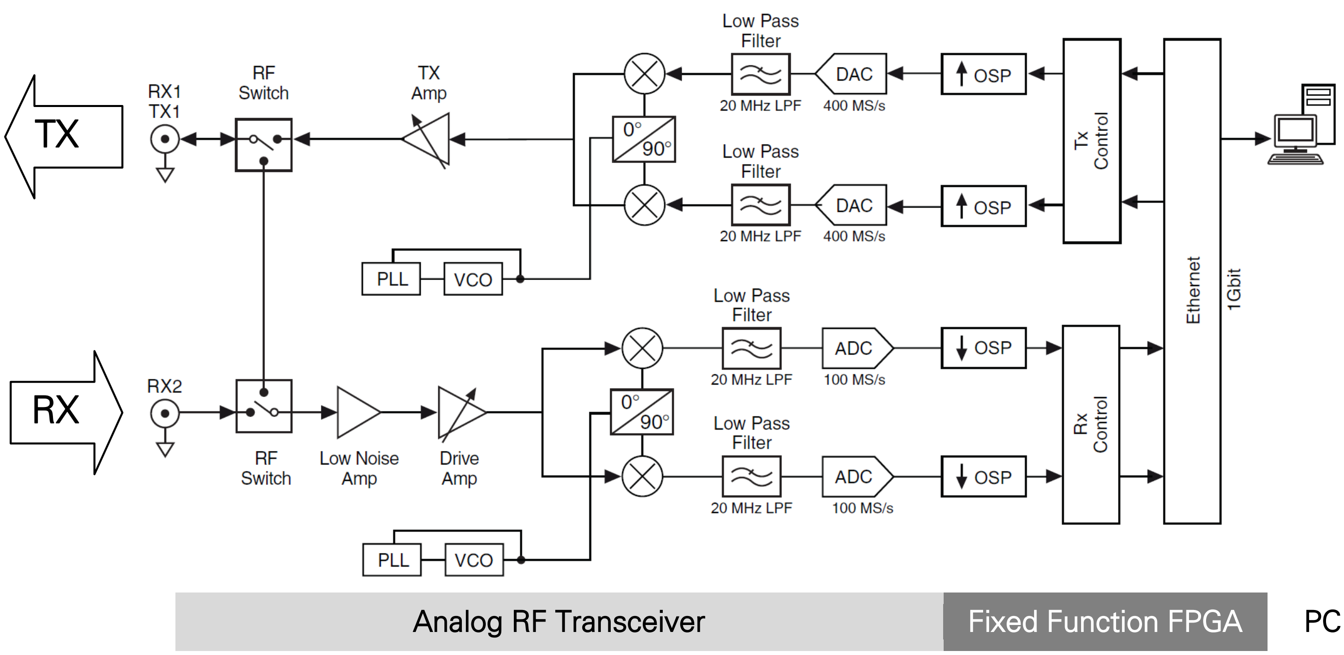


Figure 4. USRP Block Diagram

What is a Computer's Role in SDR?

A software-defined radio system is a radio communication system in which certain hardware components are implemented in software. These hardware components include filters, amplifiers, modulators, and demodulators. Because these components are defined in software, you can adjust a software-defined radio system as needed without making significant hardware changes. Since computers today may contain very fast processors and high-speed interfaces, we can leverage these abilities for our software defined radio by implementing them on a computer quickly, using LabVIEW.

Driver Software

Driver software provides application software the ability to interact with a device. It simplifies communication with the device by abstracting low-level hardware commands and register-level programming. Typically, driver software exposes an application programming interface (API) that is used within a programming environment to build application software.

For the USRP, NI-USRP is the hardware driver. As mentioned below, NI offers development environments that can make driver calls into NI-USRP, but other text-based environments can also access the hardware driver.

Application Software

Application software facilitates the interaction between the computer and user for acquiring, analyzing, processing, and presenting measurement data. It is either a prebuilt application with predefined functionality, or a programming environment for building applications with custom functionality. Custom applications are often used to automate multiple functions of a device, perform signal-processing algorithms, and display custom user interfaces.

The NI-USRP driver currently supports the National Instruments' LabVIEW graphical development environment software for rapidly developing custom applications.

For more information, navigate to ni.com and search for "Out-of-the-Box Video with NI USRP." If you are viewing this digitally, click here to access this Developer Zone tutorial directly.

Finding Hardware & Installing Software

It is first important to understand the complexities and requirements of your application to ensure that your wireless signal system is the right fit.

Identifying the Right Hardware

To help you identify the right hardware for your system, first navigate to ni.com/usrp. At this portal you will find everything that you need to know about the USRP. You can use the left-hand navigation facets to begin your search and narrow in on the right selection.

Finding the Specifications Online

The most important document for any device is the specifications document. This document is housed on the product page for each device. Once you know the device of interest, it is easiest to find the device using the search box in the upper right hand corner of the ni.com website. Simply enter the product model number (i.e., 2920) and click search. An example search query is shown on the next page, with the product page result outlined in red.

NI USRP Online Community

This ni.com community is the home for example VIs associated with the NI USRP, "Universal Software Radio Peripheral." Browse to find a starting point, or post your own NI-USRP-based examples.

Navigate to ni.com/usrp and click the Code Sharing Community image.

Glossary A: RF & Communication Reference for help understanding any terms in the Specifications Documents

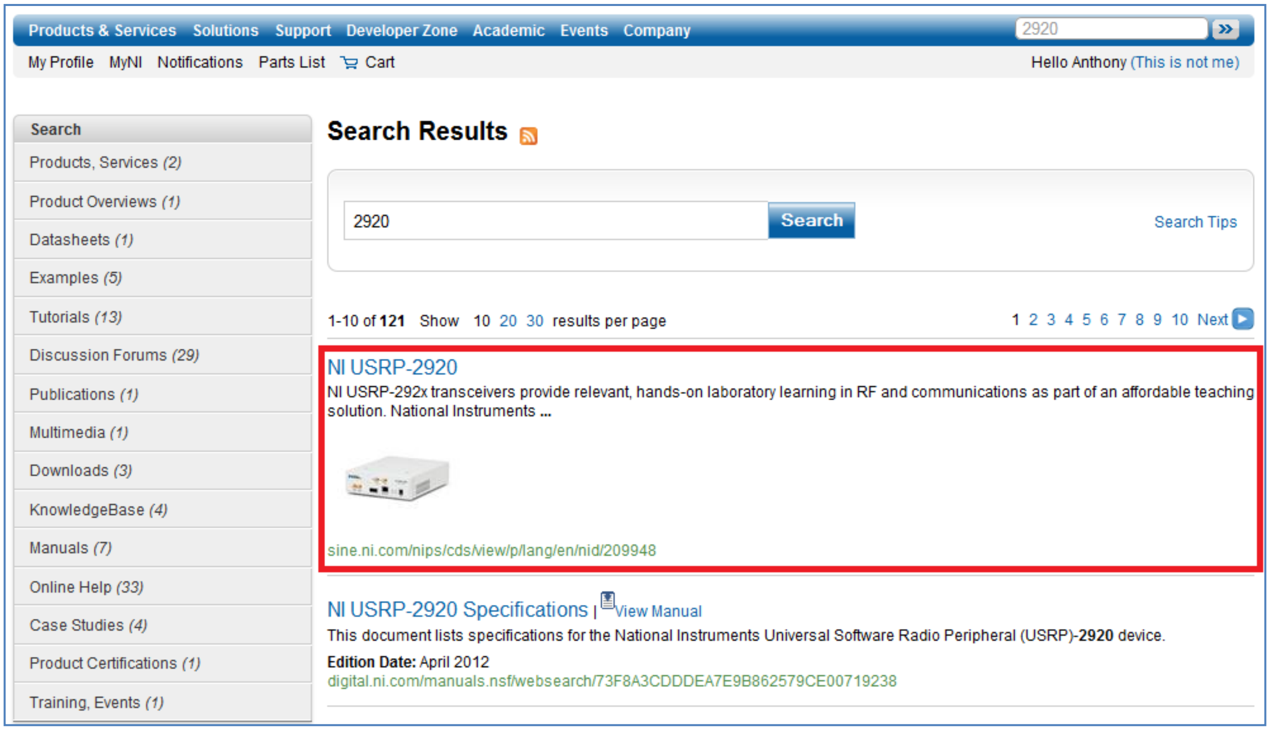


Figure 5. Search Results for “2920” at ni.com

Once on the product page, click on the **Resources** tab to surface documents about the device; you can also access the complete **NI USRP Specifications** document for the device by clicking on the **Manuals** link.

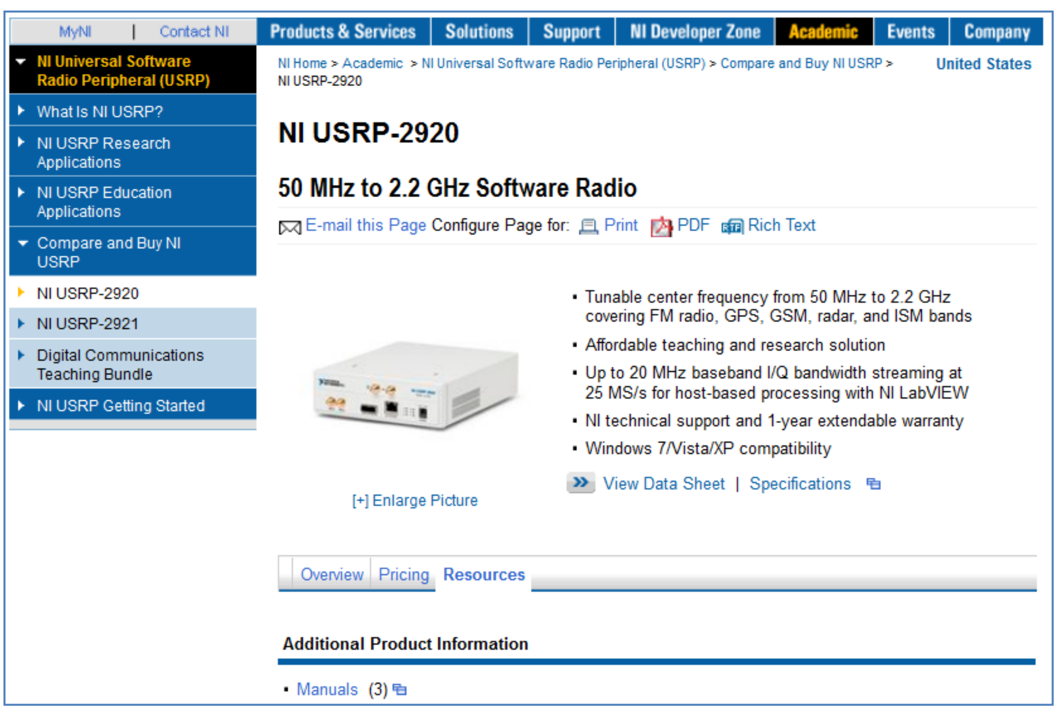


Figure 6. Product Page for NI USRP-2920

Purchasing Hardware

To receive a quote for NI USRP Hardware, you can add the hardware to your cart at ni.com from the product page for the product. You can also call your NI Sales Engineer to help design the best system for your application.

To speak with a technical representative, navigate to ni.com and click on the **Contact Us** link on the homepage.

Installing Software

To acquire data from the USRP, you will need to first install a software development environment and then the hardware driver.

Development Environment: NI LabVIEW

The development environment facilitates the interaction between the computer and user for acquiring, processing, analyzing, and presenting measurement data. It is either a prebuilt application with predefined functionality, or a programming environment for building applications with custom functionality. Custom applications are often used to automate multiple functions of a device, perform signal-processing algorithms, and display custom user interfaces.

The primary development environment for NI-USRP is NI LabVIEW. LabVIEW is a graphical programming language that abstracts the low-level complexities of text-based programming into a visual language that scientists and engineers use worldwide to acquire, analyze, process, and present data in the same environment.

Visit ni.com/trylabview for more information and to download an evaluation.

Hardware Driver: NI-USRP

Driver software provides application software the ability to interact with a device. It simplifies communication with the device by abstracting low-level hardware commands and register-level programming. Typically, driver software exposes an application-programming interface (API) that is used within a programming environment to build application software.

The hardware driver for the USRP is NI-USRP. The USRP driver comes with example LabVIEW programs and help files to get you started.

Visit ni.com/drivers and search for 'USRP' to download the latest version of the driver.

Academic Site License

If you are on a university with access to an Academic Site License (ASL), then you might have free access to almost all NI software, including LabVIEW. Contact your university software administrator for more information.

NI-USRP LabVIEW Driver

When installed the NI USRP hardware driver automatically installs example programs, help files, and functions.

NI-USRP Example Programs

To locate example files for using the USRP driver in LabVIEW, navigate to the **Start Menu » All Programs » National Instruments » NI-USRP » Examples.**

NI-USRP Help Files

There are two forms of help offered within the LabVIEW development environment: Context Help and Detailed Help.

Context Help

The context help window displays basic information about LabVIEW objects when you move the cursor over each object. To toggle the display of the context help window, select **Help » Show Context Help** or press <Ctrl-H>.

When you move the cursor over front panel and block diagram objects, the context help window displays the icon for subVIs, functions, constants, controls, and indicators, with wires attached to each terminal. When you move the cursor over dialog box options, the context help window displays descriptions of those options.

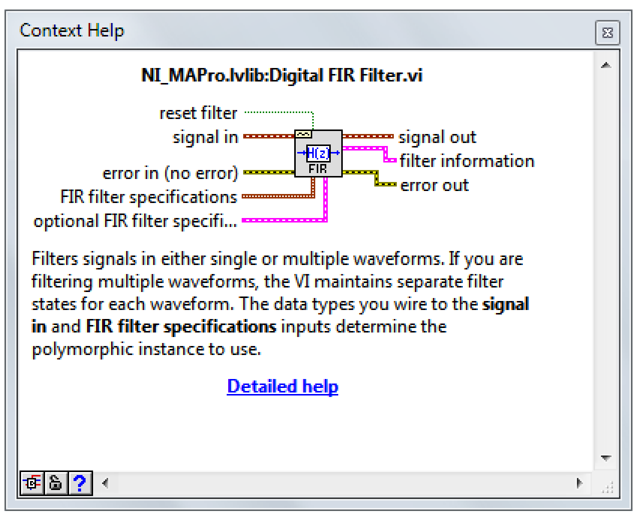


Figure 7. LabVIEW Context Help

If a corresponding detailed help topic exists for an object that the context help window describes, a blue **detailed help** link appears in the context help window (Figure 7). Click the link or the button to display the LabVIEW help for more information about the object.

Detailed Help (LabVIEW Help)

The LabVIEW help is the best source of detailed information about specific features and functions in LabVIEW. Detailed help entries break down topics into a concepts section with detailed descriptions and a how-to section with step-by-step instructions for using LabVIEW functions.

You can access the LabVIEW Help by selecting **Help » Search the LabVIEW Help**, or clicking the blue **Detailed help** link in the context help window. You also can right-click an object and select **Help** from the shortcut menu.

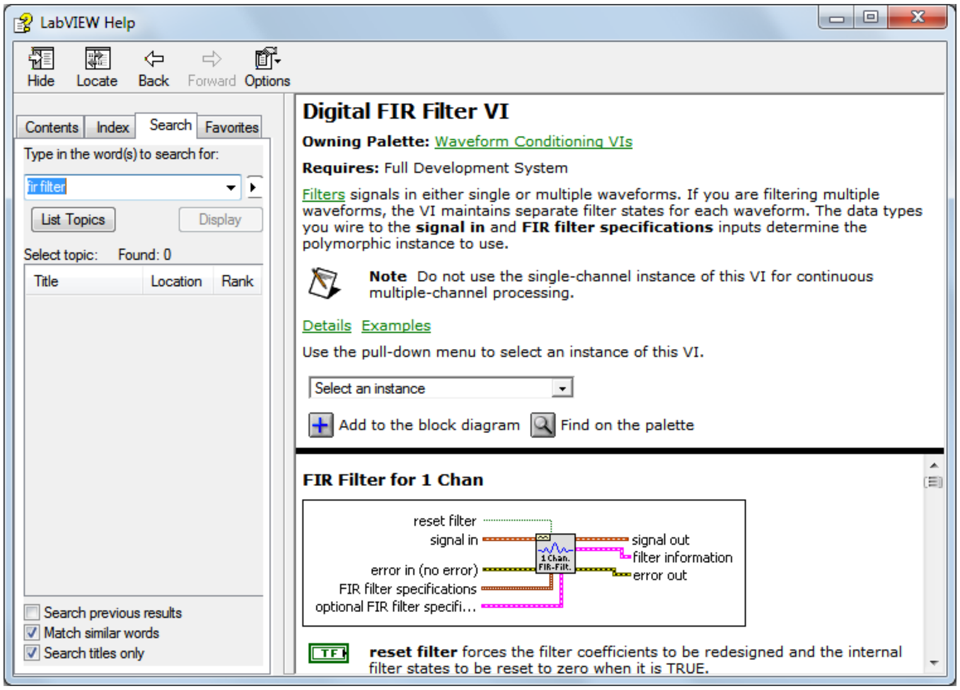


Figure 8. Detailed LabVIEW Help

NI-USRP Functions Palette

To access the NI-USRP functions in LabVIEW, navigate to the block diagram and right-click empty white space to bring up the functions palette. Then navigate to **Instrument Drivers » NI-USRP**. The functions will appear similar to the palette below. Drag and drop a function onto the block diagram to begin programming.

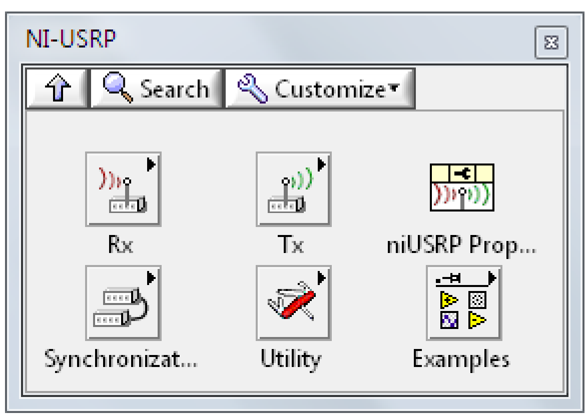


Figure 9. NI-USRP Palette in LabVIEW

niUSRP Property Node

Use the niUSRP properties to access advanced configuration options for NI-USRP driver applications.

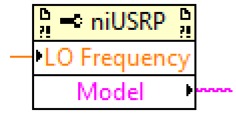


Figure 10. niUSRP Property Node

The Eight Most-Used NI-USRP Functions

The following section outlines the eight most-used USRP functions to help get you started with your experiments. They have been grouped in categories by their functionality. These categories are: Configure, Read/Write, and Close. These categories are included in most data acquisition programs and are important programming models to consider when creating a new LabVIEW Virtual Instrument (VI).

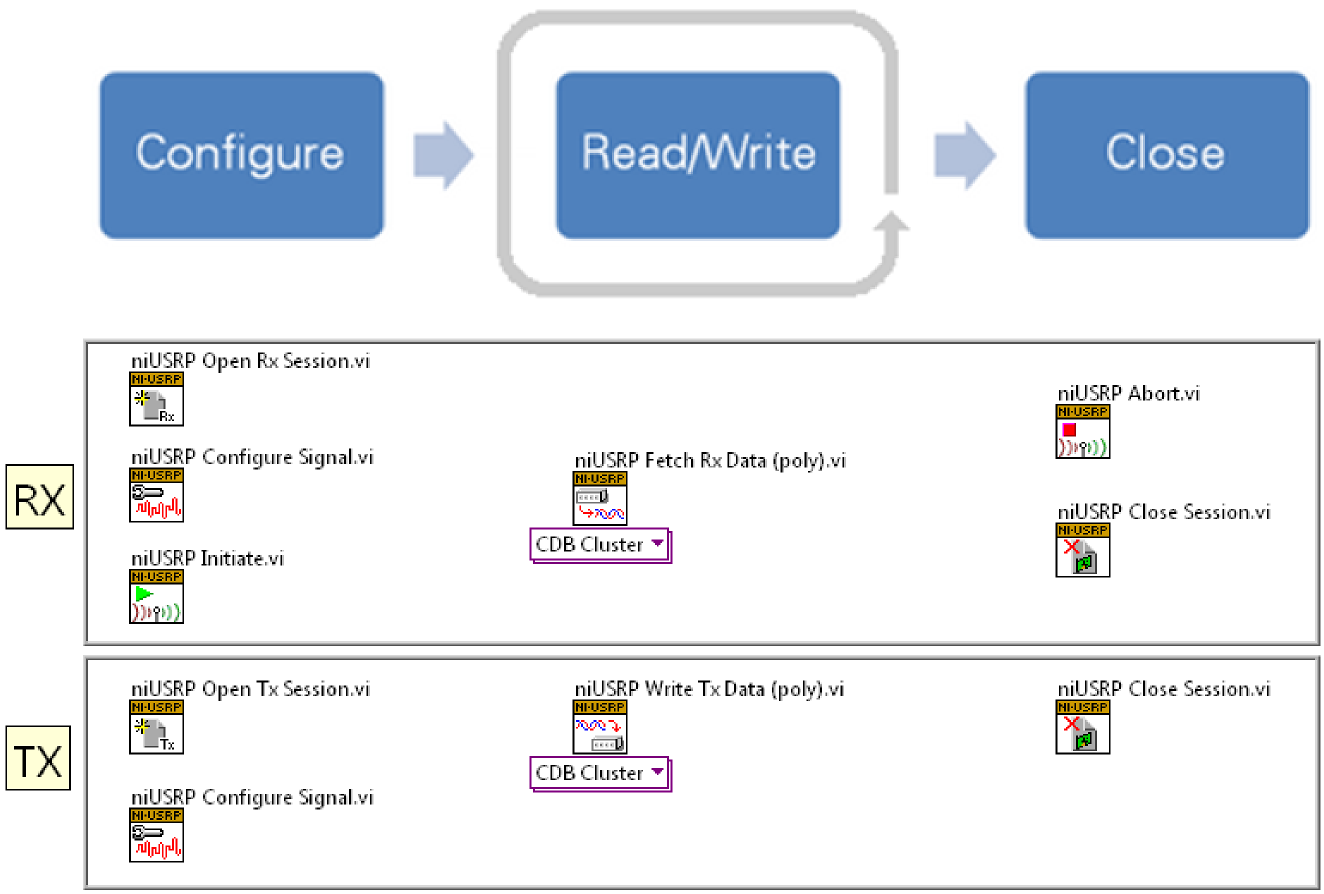


Figure 11. The Eight Most-Used NI-USRP Functions

Configure Functions

niUSRP Open Rx Session

The *niUSRP Open Rx Session VI* is the first VI that is used to create a software session with the USRP for receiving an RF signal. A session is necessary to send configuration data and retrieve IQ data from the USRP.

An Rx session can only be used with Rx functions.

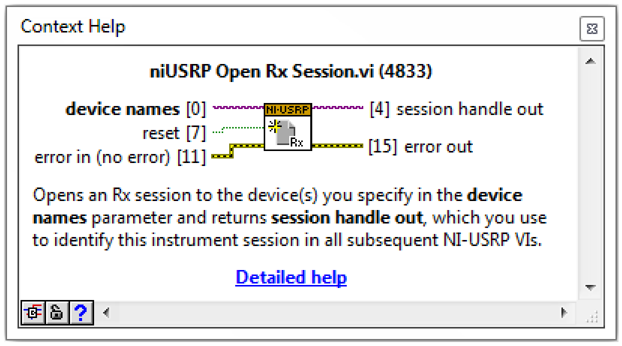
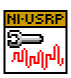


Figure 12. Context Help for niUSRP Open Rx Session VI

niUSRP Configure Signal



The *niUSRP Configure Signal VI* can be used with a receive (Rx) or a transmit (Tx) session. It sets the IQ rate, carrier frequency, gain, and active antenna. For multiple USRP configurations the channel list specifies a specific USRP. Not all IQ rates, frequencies and gains are valid. Always read the coerced values to see if the requested and actual (coerced) values are different.

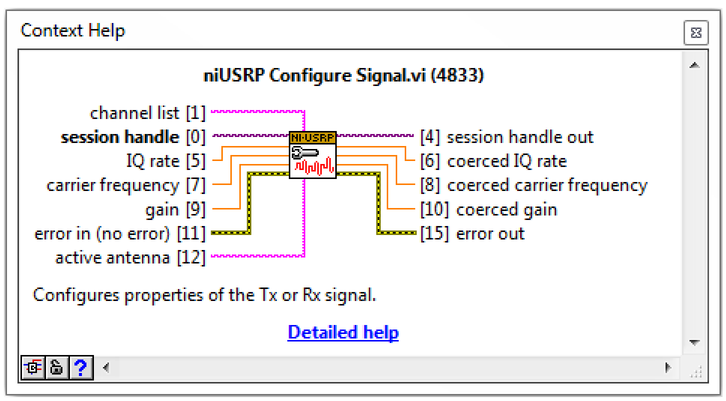


Figure 13. Context Help for *niUSRP Configure Signal VI*

niUSRP Initiate

The *niUSRP Initiate VI* starts the receive session and tells the USRP that all configuration is complete and that the USRP should begin to capture IQ data (samples).

This VI can only be used with an Rx session.

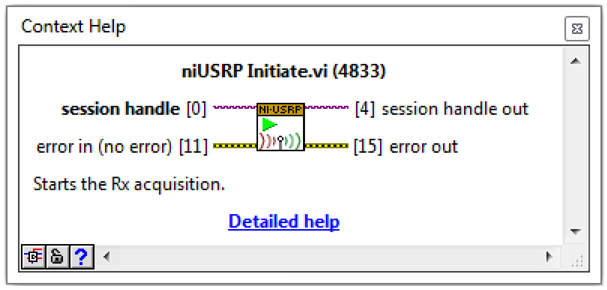
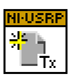


Figure 14. Context Help for *niUSRP Initiate VI*

niUSRP Open Tx Session



The *niUSRP Open Tx Session VI is* the first VI that is used to create a connection to the USRP for transmitting an RF signal. A session is necessary to send configuration data and send IQ data to the USRP.

A Tx session can only be used with Tx functions.

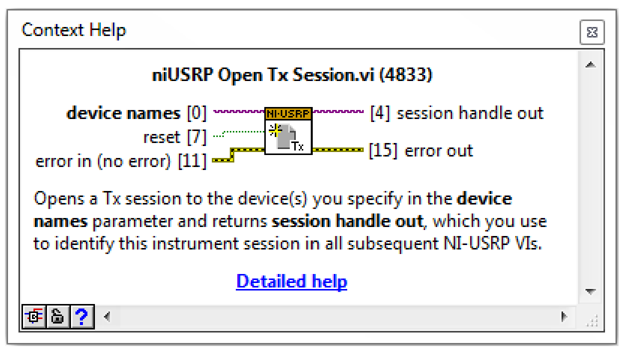
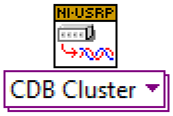


Figure 15. Context Help for *niUSRP Open Tx Session VI*

Read/Write Functions

niUSRP Fetch Rx Data (Polymorphic)



The *niUSRP Fetch Rx Data VI* allows you to retrieve IQ data from a USRP that has an Rx session created with the *niUSRP Open Rx Session VI*. This data can then be graphed in time domain, or digitally processed for analysis.

This VI is polymorphic, meaning that there are several versions (instances) of the VI available to choose from depending on the **data type** you wish to work with.

This VI can only be used with an Rx session.

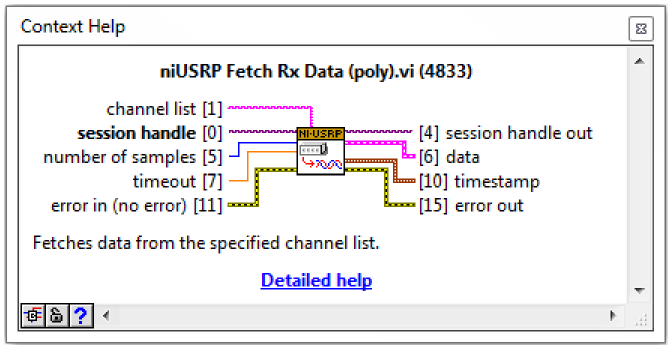
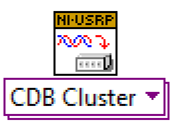


Figure 16. Context Help for *niUSRP Fetch Rx Data VI*

niUSRP Write Tx Data (Polymorphic)



The *niUSRP Write Tx Data VI* allows you to send IQ data to the USRP so that it may transmit that data at the carrier frequency specified by the *niUSRP Configure Signal VI*.

This VI is polymorphic, meaning that there are several versions (instances) of the VI available to choose from depending on the **data type** you wish to work with.

This VI can only be used with a Tx session.

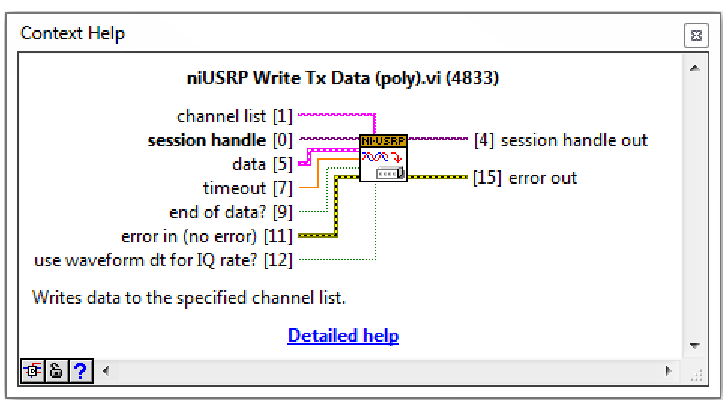


Figure 17. Context Help for niUSRP Write Tx Data VI

NI-USRP Read and Write Data Types

There are several instances of *Write Tx Data* and *Fetch Rx Data VIs* to choose from for your convenience. The table below represents the options available instances.

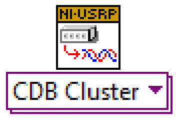
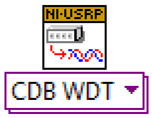
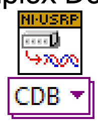
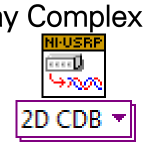
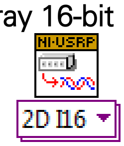
Polymorphic Type	Description
Complex Double Cluster 	Fetches a cluster of complex, double-precision floating-point data from the specified channel. Modulation Toolkit VIs use the complex, double-precision floating-point cluster data type. Use this VI in applications that use Modulation Toolkit VIs.
Complex Double Waveform Data 	Fetches complex, double-precision floating-point data in a waveform data type from the specified channel.
Complex Double 	Fetches complex, double-precision floating-point data from the specified channel.
16-bit Integer 	Fetches complex, 16-bit signed integer data from the specified channel. To use this VI, you must set the Host Data Type property to I16.
2D Array Complex Double 	Fetches complex, double-precision floating-point data from multiple channels.
2D Array 16-bit Integer 	Fetches complex, 16-bit signed integer data from multiple channels. To use this VI, you must set the Host Data Type property to I16.

Table 1. NI USRP Read and Write Data Types

Close Functions

niUSRP Abort

The *niUSRP Abort VI* tells the USRP to stop an acquisition in progress. This VI allows you to change configuration settings without completely closing the session and creating a new session.

This VI can only be used with an Rx session.

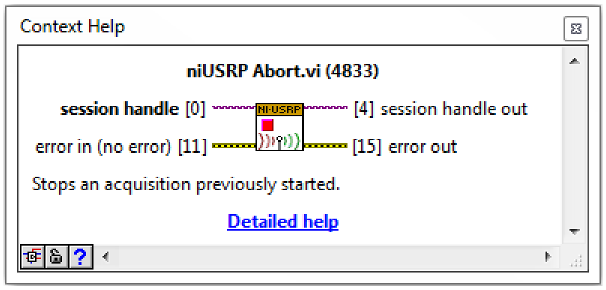


Figure 18. Context Help for *niUSRP Abort VI*

niUSRP Close Session

The *niUSRP Close Session VI* closes the current Rx or Tx session and releases the memory in use by that session. After calling this VI you can no longer transmit to or receive data from the USRP until you re-open a new session.

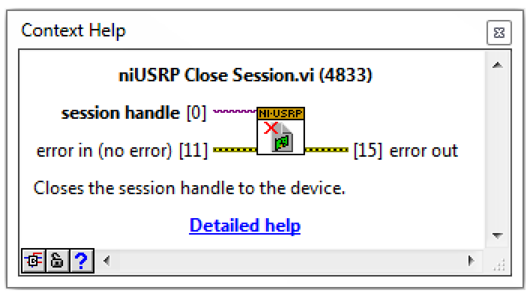


Figure 19. Context Help for *niUSRP Close Session VI*

USRP Receive and Transmit Examples

The following examples are the skeleton programs for transmitting and receiving. Each example will have a few example applications, along with example LabVIEW block diagram code to help get you started. These examples can be found at **Start » All Program » National Instruments » NI USRP » Examples**

Single Channel, Finite

Applications: Burst data transmits, frequency hopping, large frequency sweeps
Example VI Name: niUSRP EX One Shot Rx.vi

The following table lists out the VIs that you will use for this type of application. Note that some VIs are required, others can be added for more advanced functionality and some are not applicable.

NI-USRP VI	Acquire	Transmit
Open Session	✓	✓
Configure Signal	✓	✓
Initiate	✓	
Fetch/Write Data	✓	✓
While Loop		
USRP Property Node	✓	✓
Abort	✓	
Close Session	✓	✓

The single channel finite example can be used as a starting point for a simple transmitter that transmits bursts of data. A while loop is not needed if only a single burst of data needs to be transmitted. Using a while loop and the abort and initiate VIs we can change the frequency of the carrier frequency on the fly without closing and reopening Rx sessions (which takes extra time). For finite Tx sessions, a boolean indicator called **End of Data?** is set to true to let the USRP turn off the transmitter after the last sample has been transmitted.

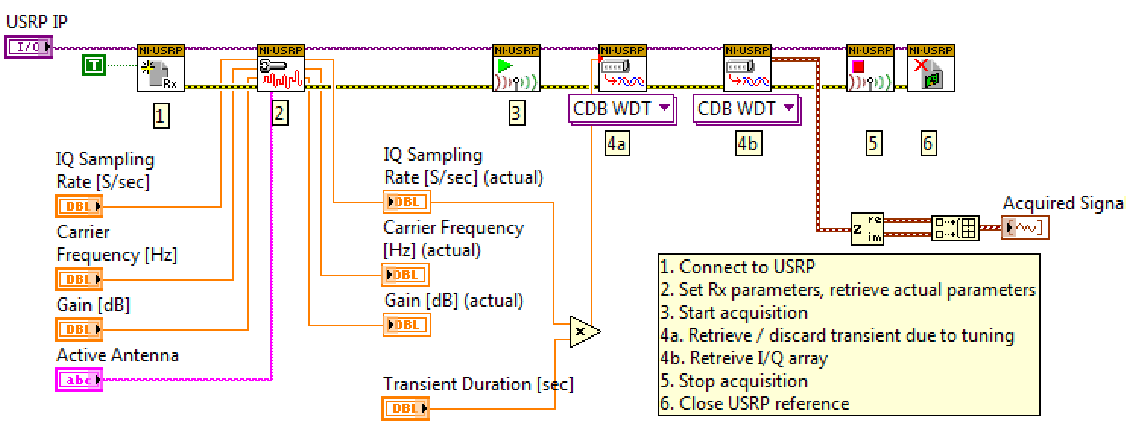


Figure 20. Example block diagram code for single USRP (channel) finite acquisition

Single Channel, Continuous

Application: Burst data reception, power lever monitoring, continuous transmission
Example VI Name: niUSRP EX Rx Continuous Async.vi

The following table lists out the VIs that you will use for this type of application. Note that some VIs are required, others can be added for more advanced functionality and some are not applicable.

NI-USRP VI	Acquire	Transmit
Open Session	✓	✓
Configure Signal	✓	✓
Initiate	✓	
Fetch/Write Data	✓	✓
While Loop	✓	✓
USRP Property Node	✓	✓
Abort	✓	
Close Session	✓	✓

The single channel continuous example is almost identical to the finite example except the added while loop. This is useful to keep from missing a transmission that comes in bursts or to receive a transmission that is continuous. For Rx sessions, the abort and initiate VIs are not used inside the while loop, which keeps the transmitter on continuously. For Tx sessions, the **End of Data?** boolean indicator is set to false to keep the transmitter on continuously. If your loop does not fetch or write fast enough then you will eventually get a buffer underflow or overflow error. You should always check for errors and exit the loop if an error occurs.

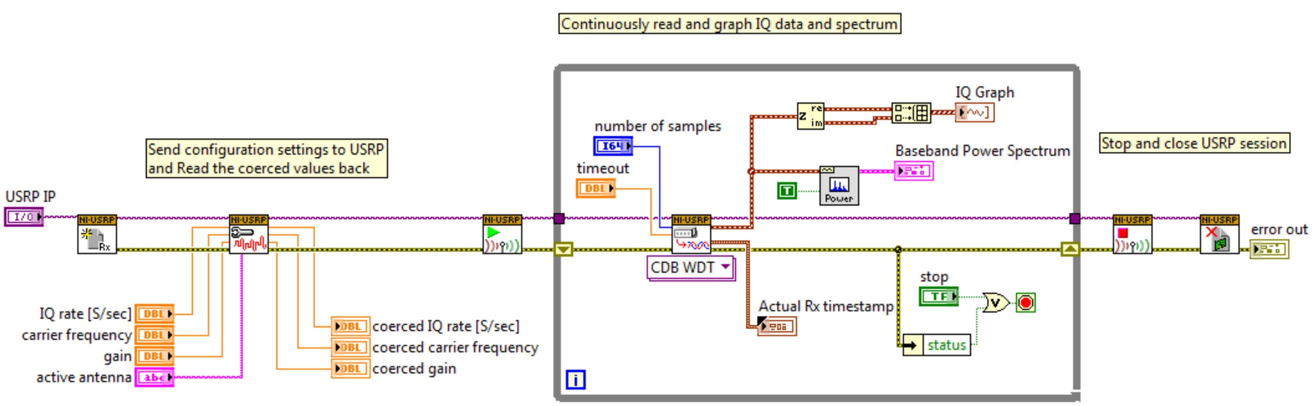


Figure 21. Example block diagram code for single USRP (channel) continuous acquisition

Multiple Channel, Continuous

Application: Broadband spectrum monitoring, MIMO transmission and reception
Example VI Name: niUSRP EX Rx Multiple Synchronized Inputs.vi

The following table lists out the VIs that you will use for this type of application. Note that some VIs are required, others can be added for more advanced functionality and some are not applicable.

NI-USRP VI	Acquire	Transmit
Open Session	✓	✓
Configure Signal	✓	✓
Initiate	✓	
Fetch/Write Data	✓	✓
While Loop	✓	✓
USRP Property Node	✓	✓
Abort	✓	
Close Session	✓	✓

The multiple channel continuous example is an extension of the continuous example that is set up for multiple USRPs (channels). Multiple USRPs allow you to obtain a larger system bandwidth by using the individual USRPs at separate frequencies. To configure a full MIMO system, you invoke a LabVIEW property node and use the MIMO cable or synchronize to an external clock source.

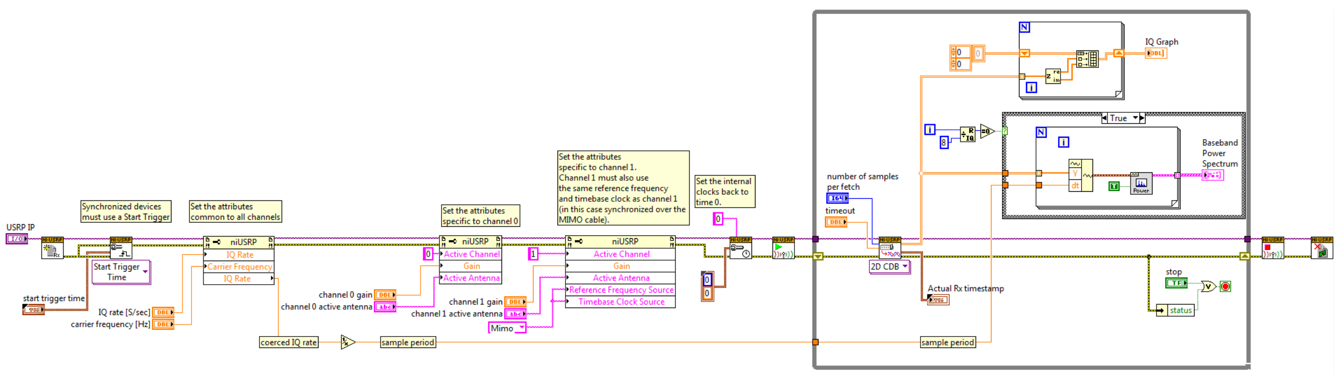


Figure 22. Example block diagram code for multiple channel continuous acquisition

Avant de commencer : l'environnement LabVIEW

Avant de commencer, il est important que vous soyez à l'aise avec l'environnement LabVIEW. Il s'agit d'un langage de programmation "G" - pour graphique - utilisé par des milliers d'ingénieurs à travers le monde pour concevoir des applications de petite et moyenne envergures, ainsi que des applications de type système.

L'interface utilisateur graphique du programme s'appelle la "Face-avant" et correspond à la fenêtre grise de l'environnement LabVIEW. Vous pouvez déposer des commandes, des boutons rotatifs, des interrupteurs, des graphes, des graphes déroulants, des LED, et bien d'autres éléments courants qui permettent à l'utilisateur/opérateur de contrôler le programme.

L'interface de programmation ou de codage s'appelle "diagramme" et correspond à la fenêtre blanche. En tant que programmeur, vous pouvez placer une multitude de fonctions et de sous-routines afin de déterminer la fonctionnalité de votre programme. LabVIEW dispose de milliers de fonctions pré-définies qui vous permettent de gagner du temps en réutilisant les fonctions les plus prisées et en accédant rapidement aux matériels.

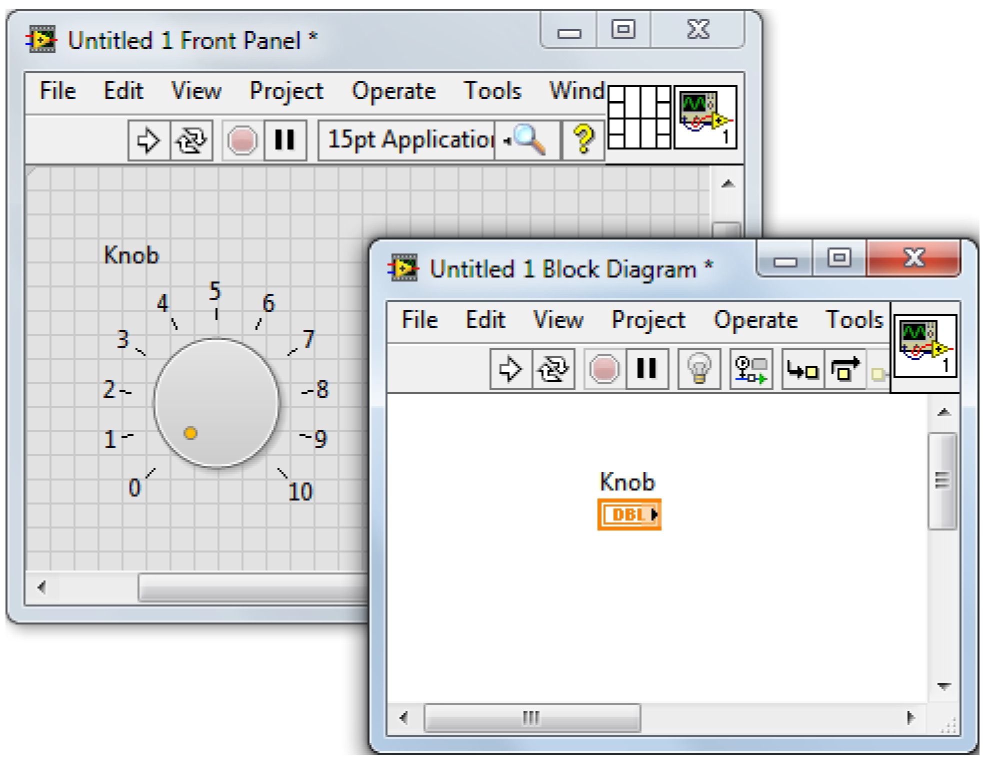


Figure 23. Un VI LabVIEW simple avec une face-avant (sur la gauche) et un diagramme (sur la droite)

Pour en savoir plus et vous familiariser avec ce logiciel, le moyen le plus rapide est un portail gratuit en ligne créé pour les étudiants et appelé Self-Paced Video Training for Students (Vidéos d'auto-formation destinées aux étudiants).

Pour de plus amples détails, reportez-vous à la section Options de formation supplémentaires à la fin de ce document.

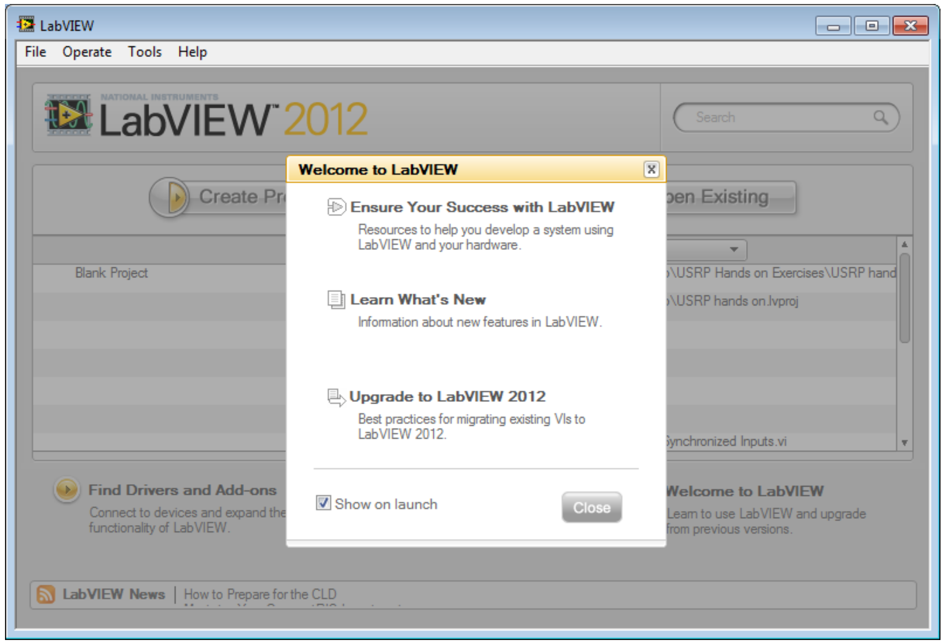
Préparer LabVIEW

Remarque : cet exercice part du principe que vous avez déjà installé tous les logiciels conformément à ce qui est décrit dans les sections précédentes. Si ce n'est pas le cas, veuillez vous référer à la section appropriée.

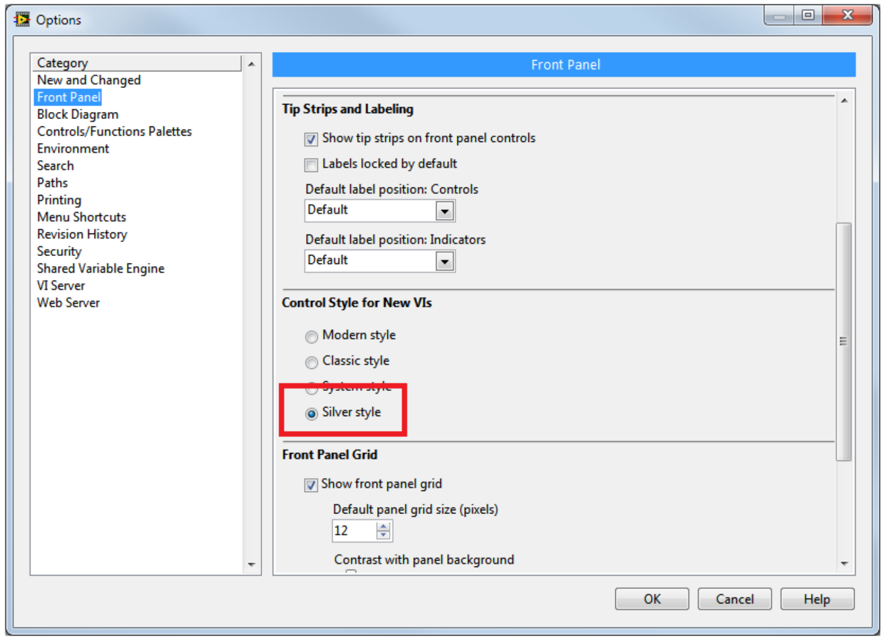
Objectifs : Configurer l'environnement LabVIEW pour une programmation facile

Instructions :

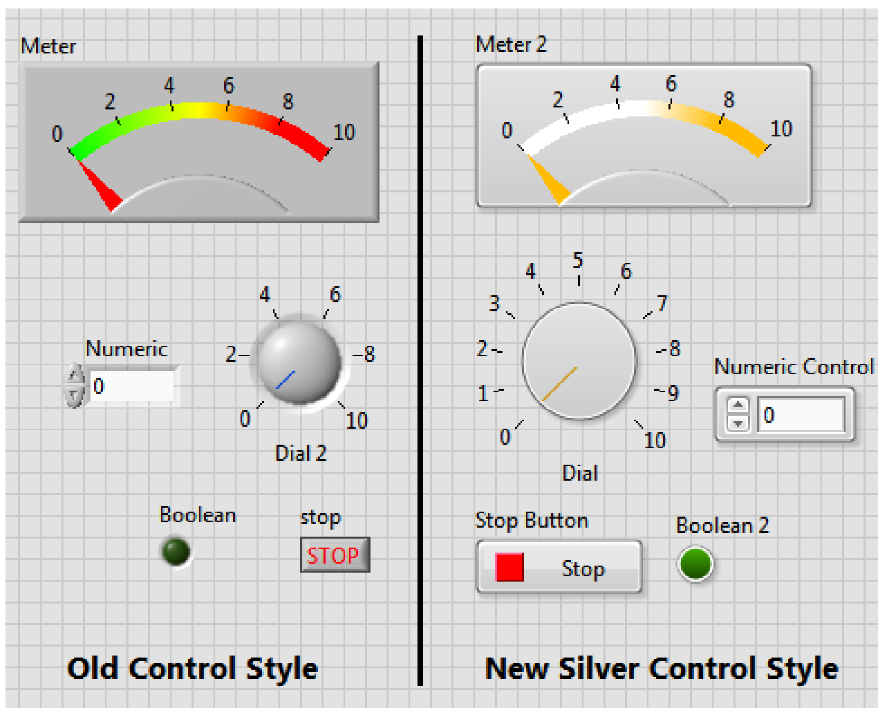
1. Lancez LabVIEW en naviguant jusqu'à **Démarrer » Tous les programmes » National Instruments » LabVIEW 2013 » LabVIEW**



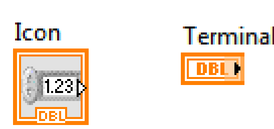
2. Sur la fenêtre de démarrage LabVIEW, naviguez jusqu'à **Outils » Options…**
    a. Dans la catégorie *Face-avant*, choisissez '**Style Argent**' pour la sous-section **Style des commandes des nouveaux VIs**. Tous les nouveaux indicateurs et commandes déposés sur la face-avant auront le look argent modernisé



Lorsque vous programmerez, votre face-avant ressemblera à ceci :

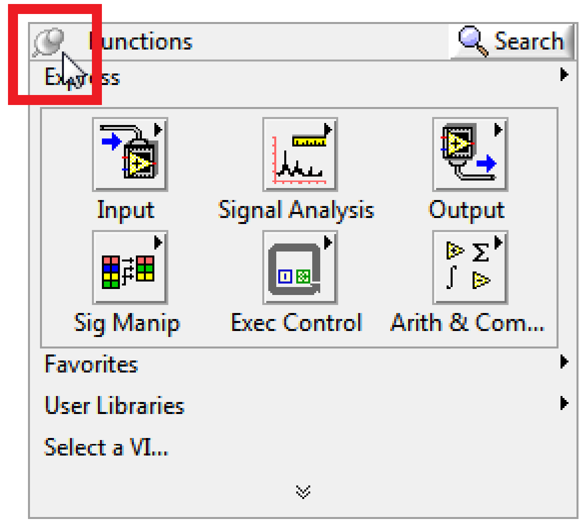


b. Dans la section correspondant à la **catégorie** *Diagramme*, désélectionnez ‘Nouveaux terminaux sous forme d'icônes’ dans la sous-section **Paramètres généraux** pour nettoyer le diagramme et gagner de la place

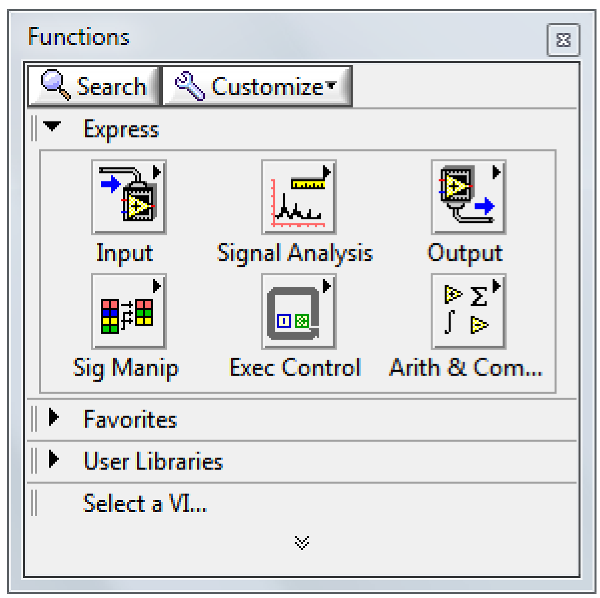


c. Cliquez sur **OK** et fermez cette fenêtre

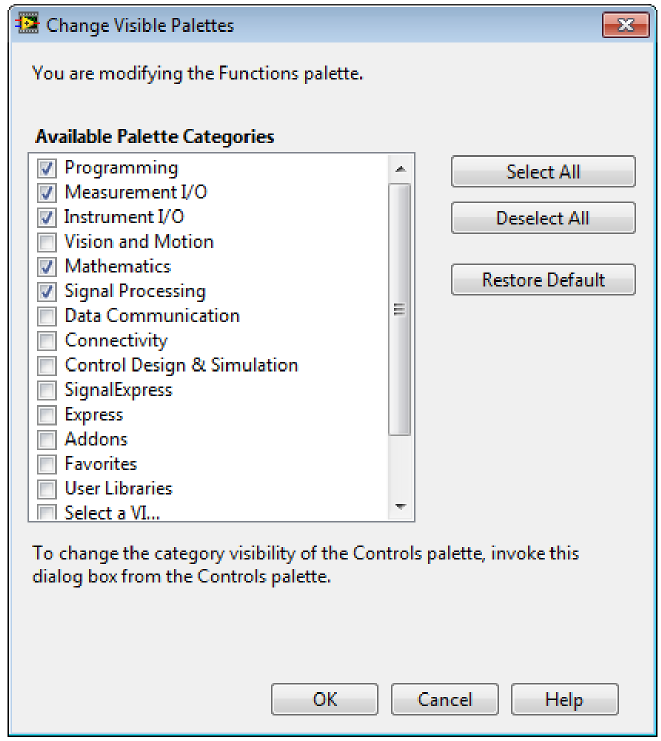
3. Sur le diagramme, effectuez un clic droit dans l'espace blanc pour afficher la palette des fonctions, puis un clic gauche sur la punaise en haut à gauche de la palette pour la fixer à l'écran



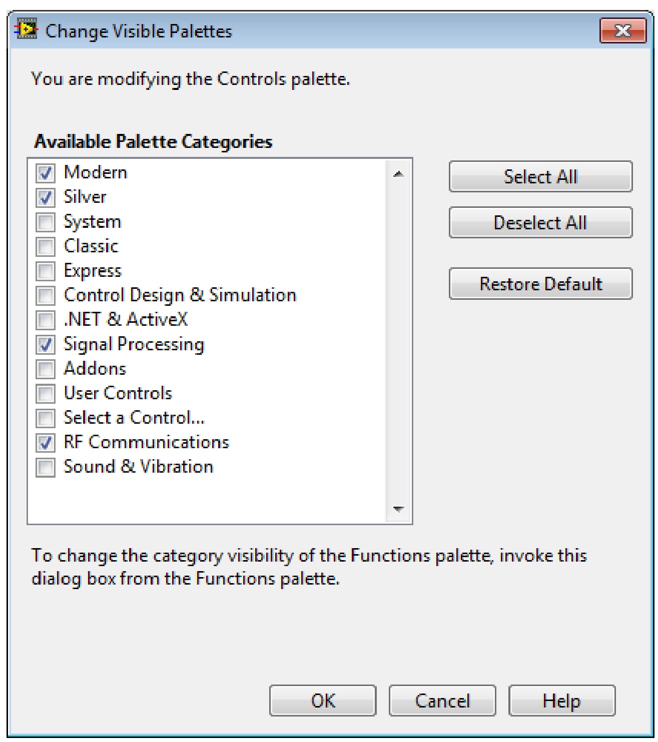
   a. Une fois celle-ci fixée, un nouveau bouton apparaît pour *Personnaliser*. Effectuez un clic gauche sur ce nouveau bouton et sélectionnez Changer les palettes visibles...



   b. Cliquez sur le bouton Tout désélectionner.
   c. Cliquez sur les cases à cocher correspondant à Programmation, E/S de mesure, E/S d'instruments, Mathématiques, Traitement du signal, et RF Communications, puis cliquez sur OK



4. Passez à la face-avant et changez les palettes visibles en Moderne, Argent, Traitement du signal, et RF Communications



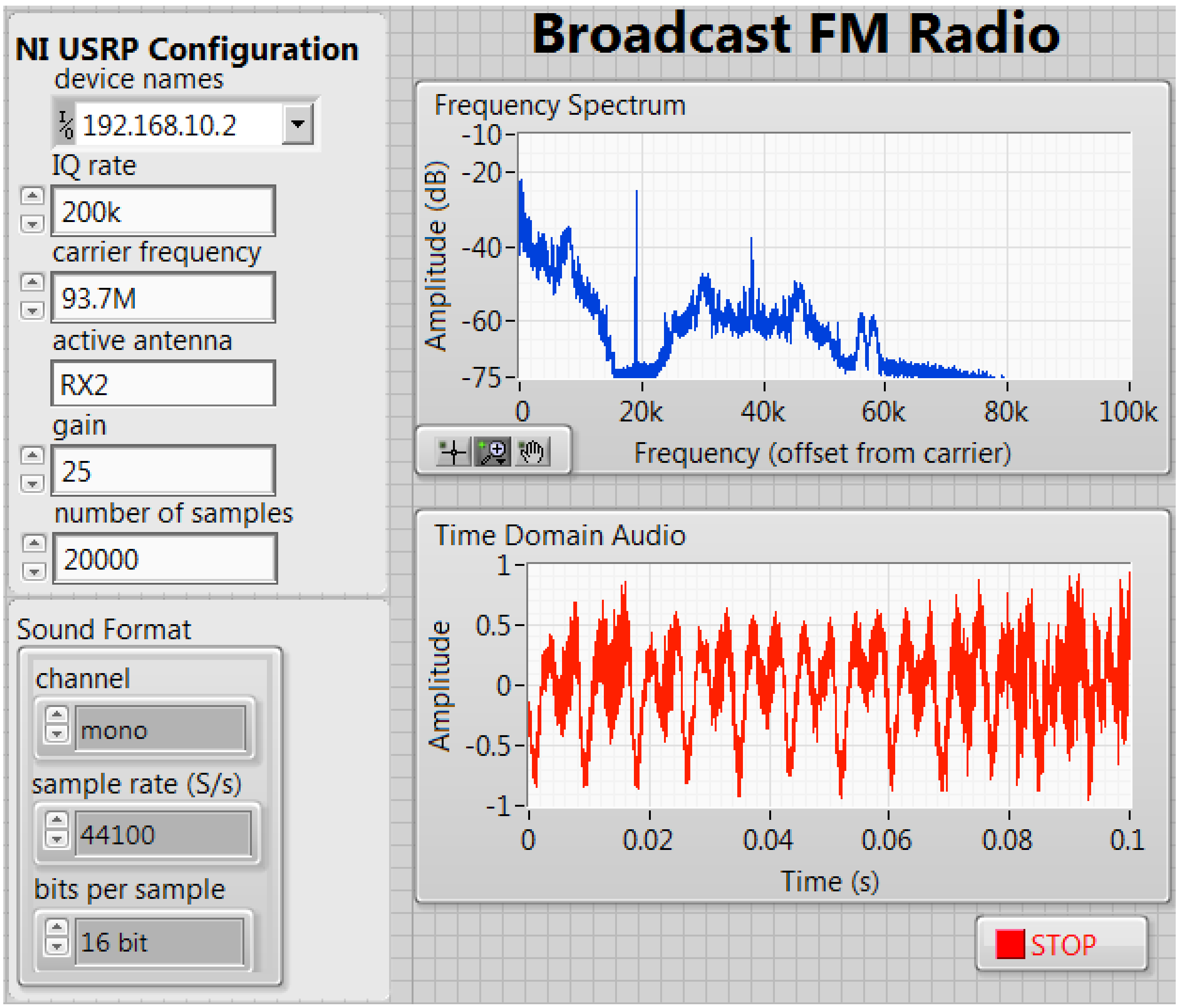
5. Cliquez sur le bouton OK pour fermer la boîte de dialogue des palettes visibles

Raccourcis LabVIEW

Le tableau ci-dessous recense les raccourcis clavier disponibles dans l'environnement LabVIEW. Reportez-vous à la Carte de référence rapide LabVIEW pour obtenir une version PDF des raccourcis clavier.

Frappe de clavier	Description
Ctrl + C	Copier l'élément ou les éléments sélectionnés
Ctrl + V	Coller l'élément ou les éléments sélectionnés
Ctrl + X	Couper l'élément ou les éléments sélectionnés
Ctrl + Z	Annuler
Ctrl + Espace	Activer le placement rapide
Ctrl + H	Activer/désactiver l'Aide contextuelle
Ctrl + B	Supprimer tous les fils de liaison brisés du diagramme
Ctrl + E	Naviguer entre un diagramme et sa face-avant
Ctrl + R	Exécuter le VI sélectionné
Ctrl + S	Enregistrer le VI sélectionné
Ctrl + T	Mosaïque verticale des fenêtres de la face-avant et du diagramme
Ctrl + U	Nettoyer le diagramme
Ctrl + cliquer et glisser	Insérer un espace vide sur le diagramme

Tableau 2. Raccourcis clavier LabVIEW



Travaux Pratiques

Création d'une radio FM logicielle

Exercice1 : Trouver une station radio

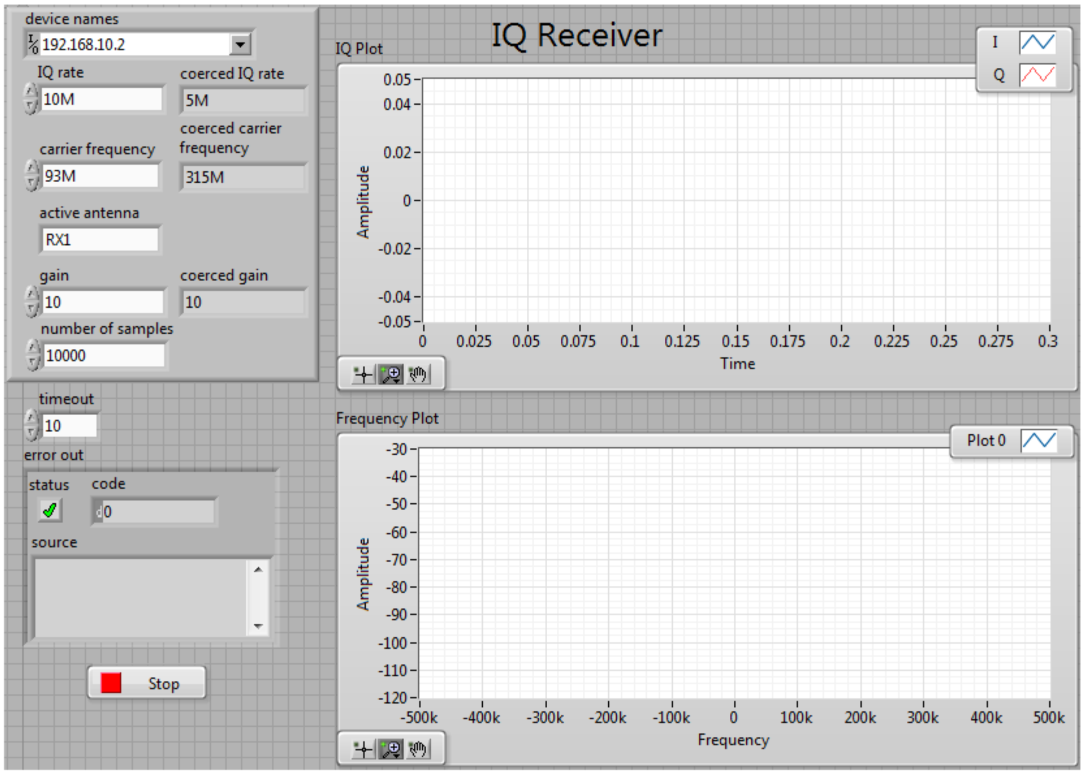
Remarque : cet exercice part du principe que vous avez déjà installé tous les logiciels et configuré tous les matériels.

Objectifs

a) Utiliser un exemple pour trouver toutes les stations de radio dans la région
b) Analyser ces stations de radio en utilisant le graphe de la face-avant

Partie A

1. Sur le bureau, ouvrez le dossier **niUSRP Hands-on**
    a. À l'intérieur du dossier Exercise, ouvrez **Exercise 1A.VI**

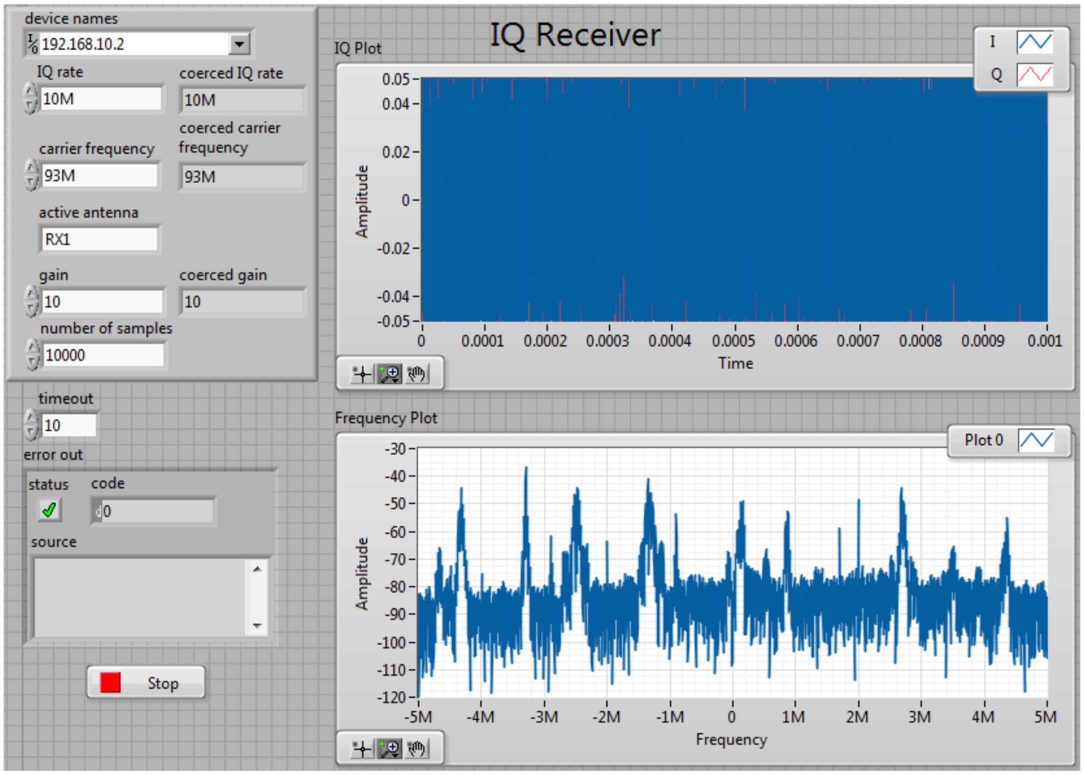


2. Configurez la face-avant comme suit :

Paramètre	Valeur
Device names	192.168.10.2
IQ Rate	10M
Carrier frequency	93M
Active antenna	RX1
Gain	1
Number of samples	20k
Timeout	10

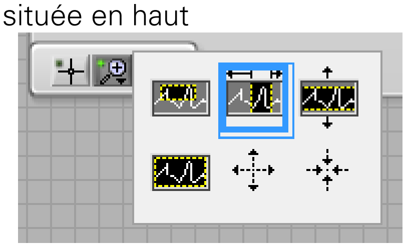
Remarque : faites attention à la différence entre M et m. LabVIEW interprète m comme milli et M comme méga

3. Appuyez sur la flèche d'exécution et vous verrez un graphe similaire à celui ci-dessous. Si vous n'avez pas un assez grand nombre de pics dans votre graphe, arrêtez le programme, modifiez le gain (jusqu'à 30) et exécutez-le à nouveau



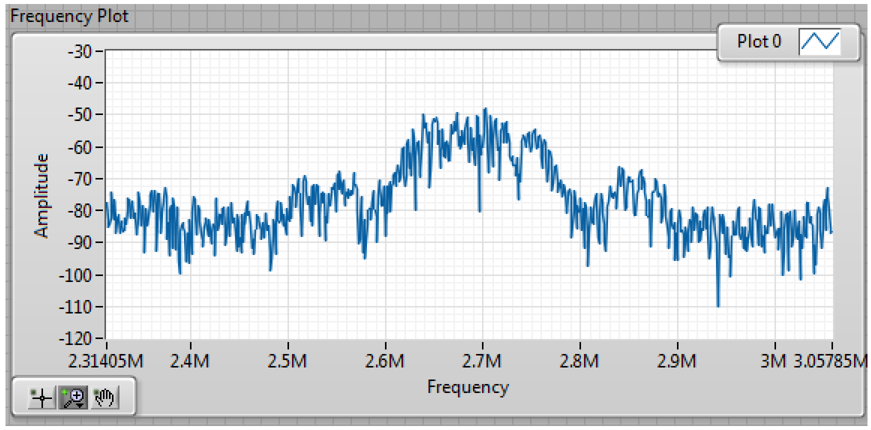
4. Vous regardez une portion du spectre d'une radio FM. En France, chaque station radio se voit attribuer une fréquence centrale comprise entre 87,5 MHz et 108 MHz. Observons une seule et même station
   Remarque : votre graphe de fréquence aura une autre apparence que celui illustré ci-dessus car l'activité des stations de radio dépend des régions.

5. Sur le tracé Frequency, cliquez sur l'icône **magnifier**, puis à partir du menu local cliquez sur l'icône centrale située en haut

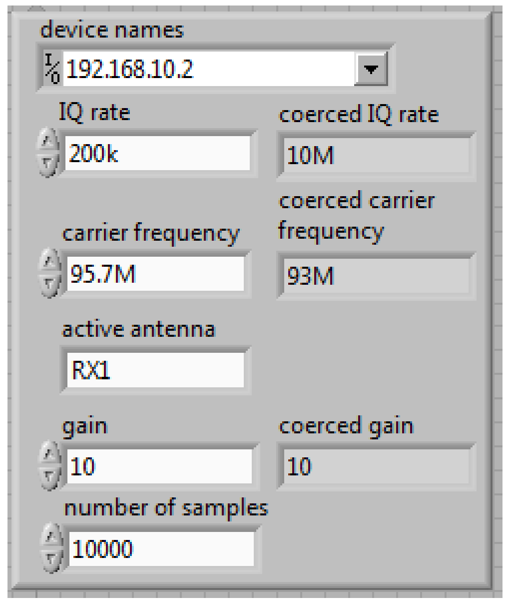


6. Choisissez une station de radio sur le graphe, puis faites glisser votre curseur de la gauche vers la droite autour du pic

7. Sur cette illustration, la station de radio FM va environ de 2,6 MHz à 2,8 MHz. La bande passante de cette station est de 200 kHz, ce qui sera notre nouvelle fréquence IQ. La fréquence centrale de cette station est à +2,7 MHz de notre fréquence porteuse actuelle de 93 MHz (95,7M)

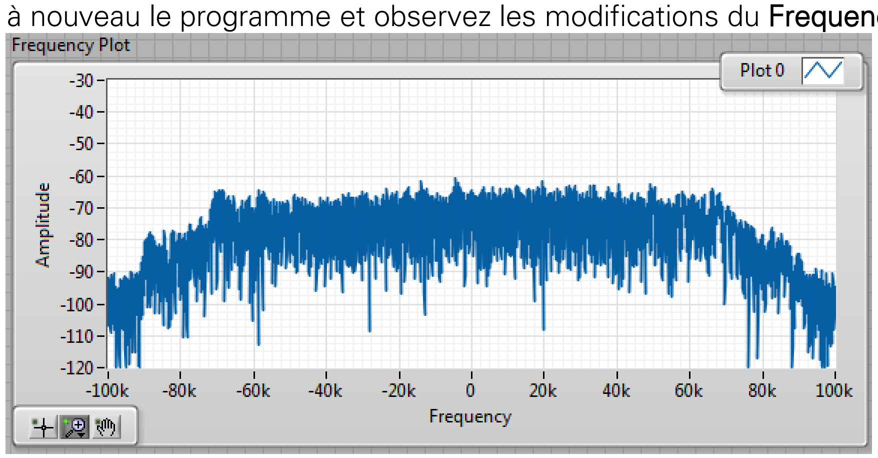


8. Pour régler uniquement cette station
    a. Arrêtez le programme
    b. Définissez **carrier frequency** à 95,7M
    c. Définissez **IQ rate** à 200k



Remarque : cette station peut ne pas être active dans votre région mais le même processus reste valable pour les autres

9. Exécutez à nouveau le programme et observez les modifications du **Frequency Plot**



Exercice 2 : Démoduler une radio FM

Remarque : cet exercice part du principe que vous avez déjà installé tous les logiciels et configuré tous les matériels.

Objectifs

a) Acquérir, représenter sous forme de graphe et écouter une station de radio FM
b) Comprendre les paramètres qui contrôlent la façon dont un programme acquiert les données
c) Découvrir les principes du flux de données LabVIEW

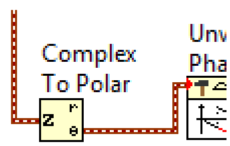
Partie A

1. Naviguez jusqu'au dossier **Exercises**
   a. Lancez **Exercise 2A.VI**

2. Sur la barre de menus, sélectionnez **Fenêtre»Afficher le diagramme**
   a. Les fonctions et VIs nécessaires pour démoduler, représenter sous forme de graphe et écouter la station de radio FM sont déjà placés sur le diagramme. Vous allez devoir les connecter dans le bon ordre pour mener à bien cet exercice

3. À l'intérieur de la boucle While, trouvez le VI **niUSRP Fetch Rx Data (poly)**
   a. Câblez la sortie **data** à l'entrée **z** de la fonction **Complex To Polar**

4. Câblez la sortie **theta** de la fonction **Complex To Polar** à l'entrée **Phase** du VI **Unwrap Phase**



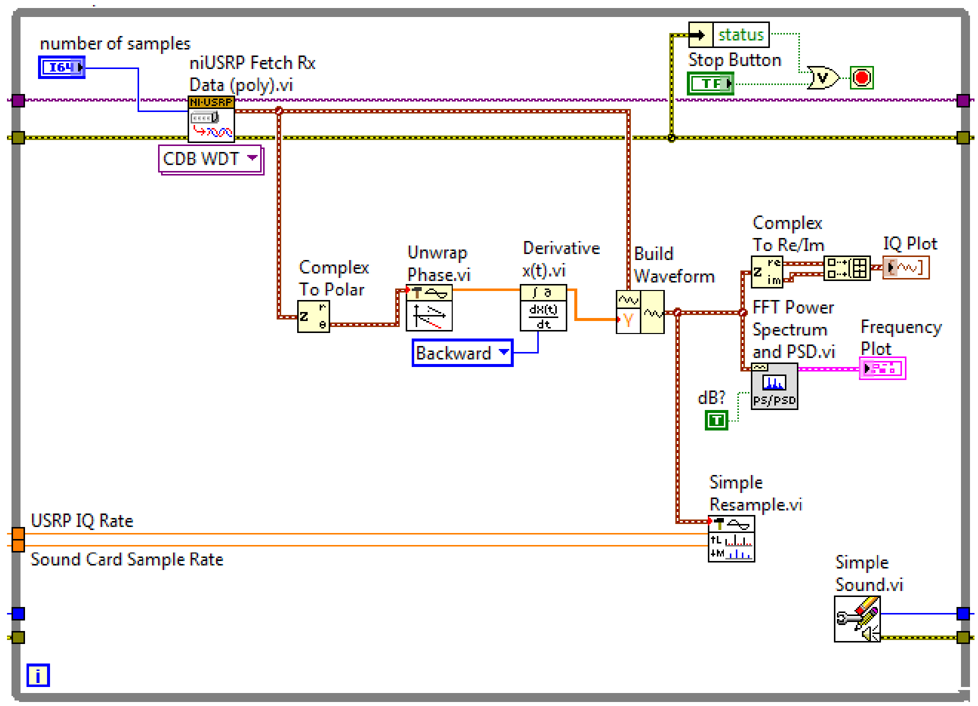
(Assurez-vous que vous n'avez pas câblé la sortie r)

5. Câblez **Phase déroulée** à l'entrée **X** du VI **Derivative x(t)**

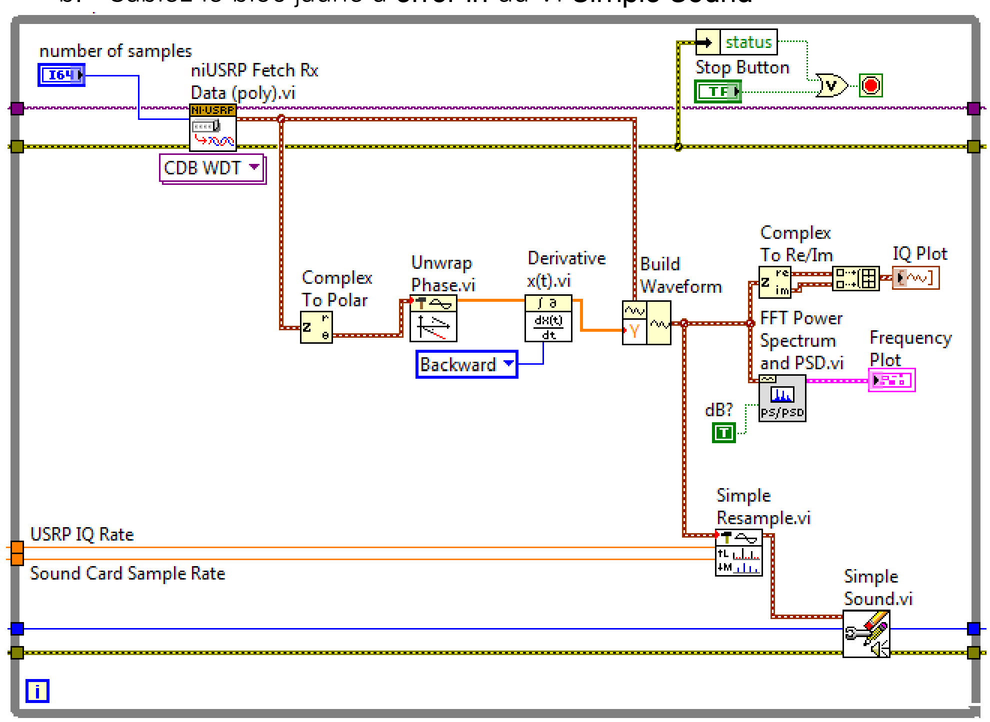
6. Câblez la sortie **dX/dt** à l'entrée **Y** de la fonction **Build Waveform**

7. Câblez **waveform en sortie** à plusieurs blocs :
   a. à l'entrée **z** de la fonction **Complex To Re/Im**
   b. à l'entrée **signal temporel** du VI **FFT Power Spectrum and PSD**
   c. à **input waveform** du VI **Simple Resample**

8. Câblez **output waveform** du VI **Simple Resample** à l'entrée **data** du VI **Simple Sound**



9. Configurez la carte son
    a. Câblez le bloc bleu en bas à gauche de la boucle While à l'entrée **task ID** du VI **Simple Sound**
    b. Câblez le bloc jaune à **error in** du VI **Simple Sound**



10. Sur la barre de menus, sélectionnez **Fenêtre»Afficher la face-avant**

11. Enregistrez le VI et nommez-le Exercise2A.vi

12. Exécutez le VI

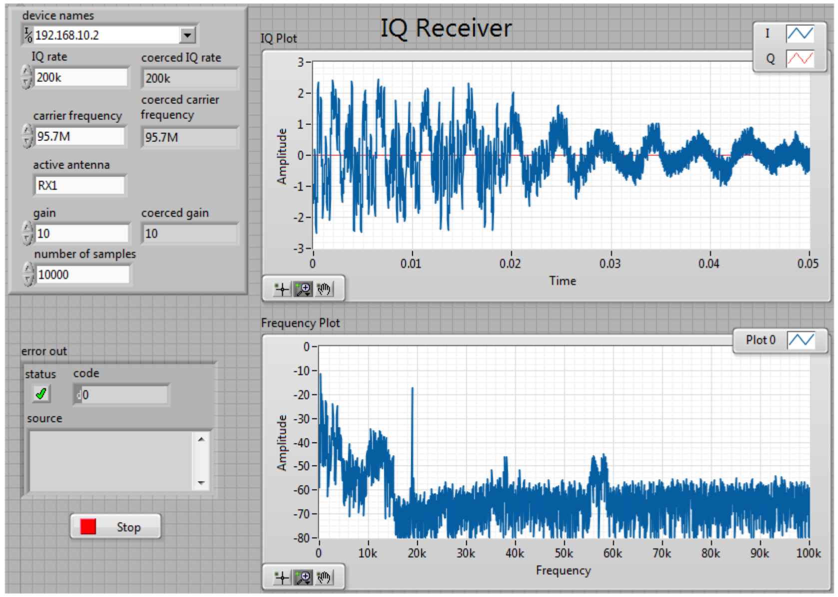


Figure 24. Face-avant achevée

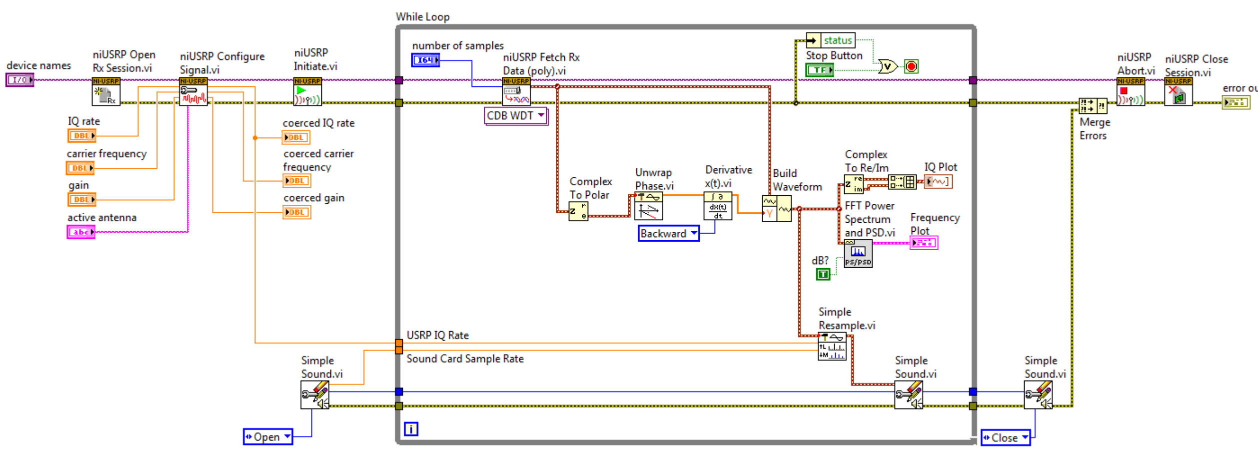
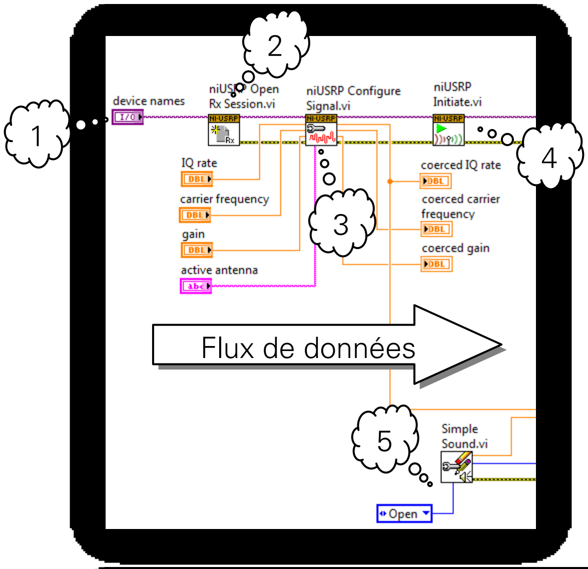
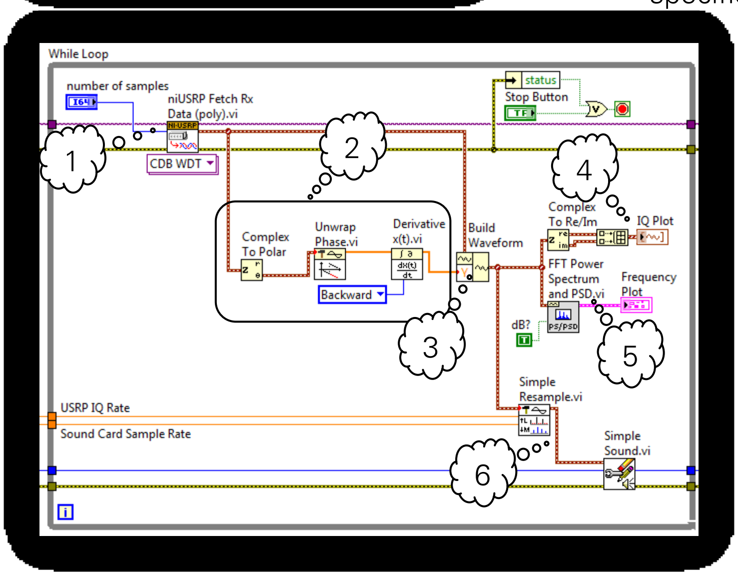


Figure 25. Diagramme achevé

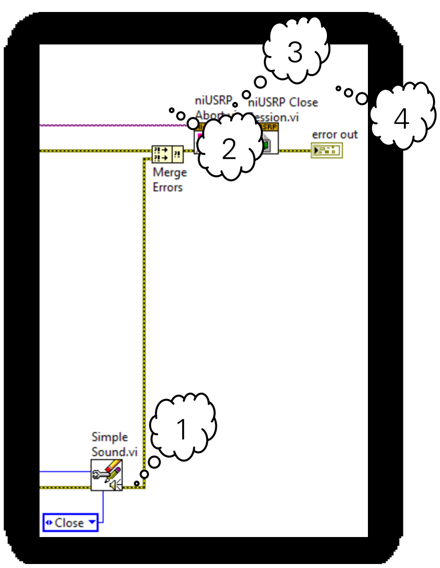
Explication sur le flux de données



1. Le terminal **device names** est relié à une commande de la face-avant et fournit l'adresse IP de l'USRP
2. Ce VI crée un lien de communication entre le programme et l'USRP. Il retourne un descripteur de session sur le fil de liaison violet.
3. Le VI **Configure** définit les paramètres pour l'acquisition de notre signal. Ces paramètres proviennent de plusieurs *commandes* différentes sur la face-avant. Les paramètres réels utilisés par l'USRP (contraints) sont affichés par les *indicateurs* de la face-avant.
4. Le VI **Initiate** indique à l'USRP qu'il doit commencer à acquérir les données.
5. Le VI **Simple Sound** vous permet d'ouvrir, d'écrire ou de fermer la carte son en fonction de l'**Action** spécifiée.



1. Le VI **Fetch Rx Data** acquiert un certain **nombre d'échantillons (number of samples)** du buffer de réception.
2. Ces trois VIs et fonctions effectuent la démodulation FM.
3. La fonction **Build Waveform** prend la valeur **dt** du VI Fetch Rx Data de façon à ce que le tracé de la FFT et celui de l'IQ s'adaptent correctement à l'échelle de l'axe des x.
4. Représente les composantes I et Q sur un seul et même tracé du graphe.
5. Représente la FFT du signal sous forme de graphe.
6. Ré-échantillonne les données démodulées à 44,1 kHz pour la carte son.



1. Le VI Simple Sound ferme la carte son pour permettre aux autres programmes de l'utiliser.
2. Le VI **Merge Errors** combine plusieurs fils de liaison d'erreur provenant de branches de code parallèles.
3. Le VI **Abort** cesse d'acquérir les données et le VI **Close Session** libère toute mémoire utilisée.
4. Affiche toutes les erreurs sur un indicateur de la face-avant.

Explication sur l'algorithme

Le principe qui sous-tend la modulation de fréquence (FM) est que l'amplitude d'un signal analogique fait varier la fréquence d'un signal porteur. Il existe des équations bien connues qui modélisent un modulateur de fréquences en tant que modulateur de phase et vice-versa. Nous préférons une équation en termes de phase car la phase est facile à obtenir à partir d'un signal IQ en utilisant la fonction arc tangente.

$$FM\ signal = f_c + k_f m(t)$$

Où f_c est la porteuse démodulée, k_f est la sensibilité de la fréquence et

m(t) est le signal de message

Intégrer le signal FM en fonction du temps et multiplier par 2π nous donne

$$\theta_i(t) = \cancelto{0}{2\pi f_c t} + 2\pi k_f \int_0^t m(\tau)d\tau$$

L'USRP se charge de la conversion par abaissement de fréquence RF, de sorte que f_c est désormais égale à zéro et que notre premier terme est éliminé de l'équation ci-dessus. Il ne reste que la constante sensible à la fréquence k_f multipliée par l'intégrale du signal de notre message m(t). Si nous pouvons obtenir la phase $\theta_i(t)$ de la forme d'onde IQ, alors nous pouvons prendre la dérivée pour récupérer notre signal de message m(t).

Nous nous heurtons à un problème car la sortie de l'arctangent est aux alentour de ±180 degrés. Nous pouvons résoudre ce problème en ajoutant des multiples de 360 degrés de sorte que la différence de phase ne sera jamais interrompue. Cela s'appelle le déroulement de phase.

Dans l'image ci-dessous, le signal sur le graphe de gauche est limité entre -180 degrés et +180 degrés. Le signal sur la droite illustre la phase d'un signal déroulé de 0 à +720 degrés.

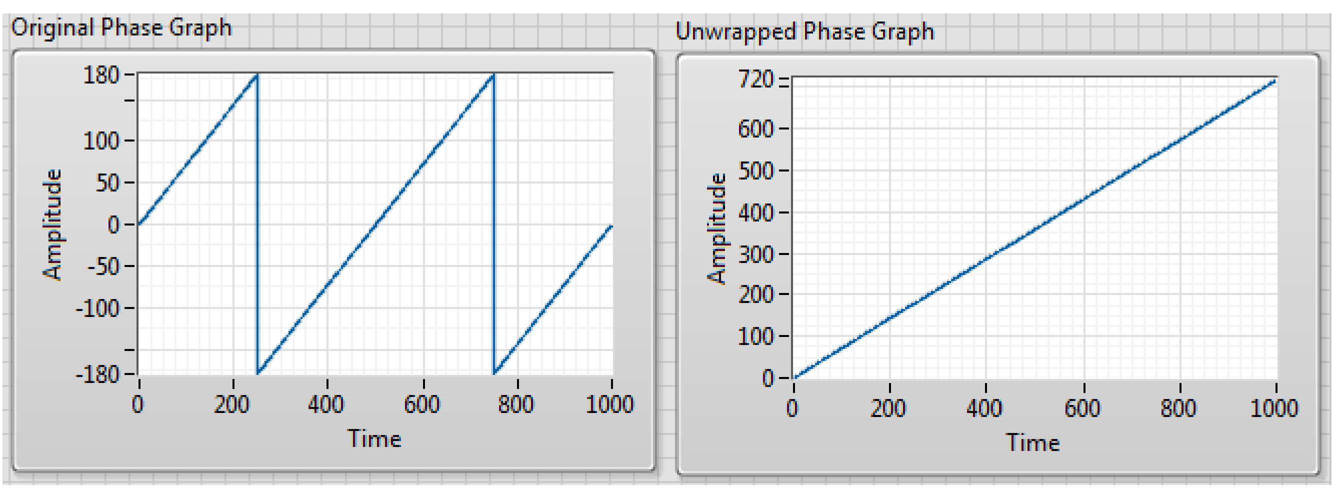


Figure 26. Comparaison d'un signal original et d'un signal déroulé

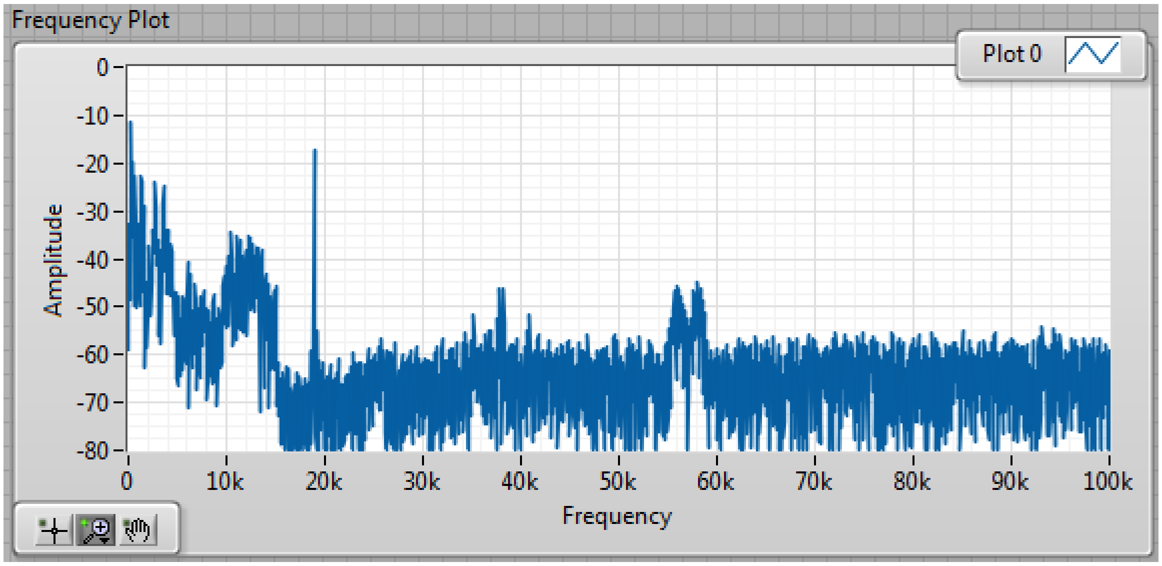


Figure 27. Tracé de fréquence illustrant un signal FM démodulé

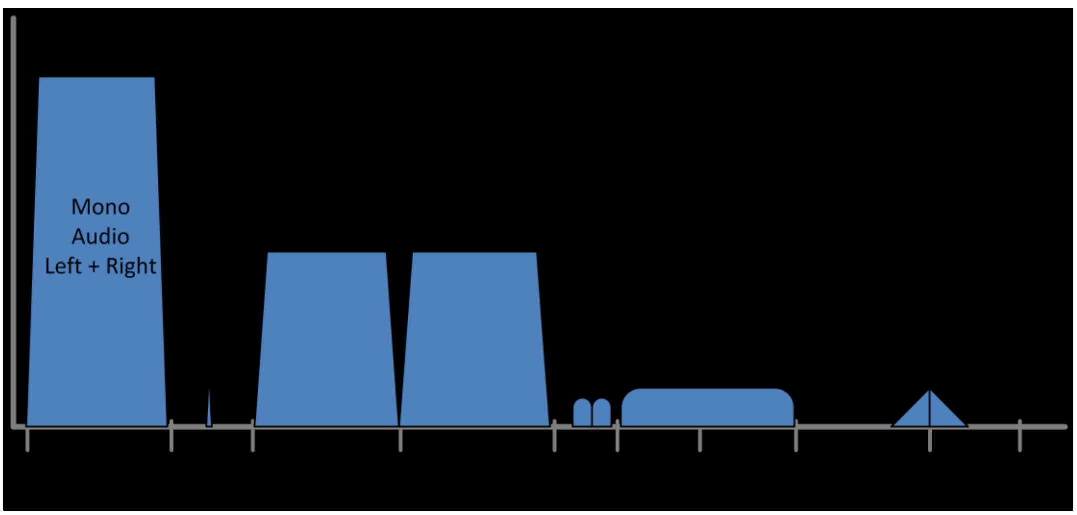


Figure 28. Image provenant de l'[entrée Wikipedia intitulée FM Broadcasting](http://en.wikipedia.org/w/index.php?title=FM_broadcasting&oldid=533382011)[2]

Figure 28 montre le graphe du spectre de fréquence idéal d'une station de radio FM qui a été démodulée. Le bloc sur le côté gauche de 30 Hz à 15 kHz est le son que nous écoutons dans nos haut-parleurs. Lorsque nous ré-échantillonnons la forme d'onde pour notre carte son, tout ce qui va au-delà de l'audio mono est filtré.

Il y a une tonalité pilote de 19 kHz qui signifie l'existence d'audio stéréo. Cet audio centré à 38 kHz peut être décalé en bande de base, puis ajouté ou soustrait à l'audio mono pour obtenir les voies stéréo gauche ou droite.

Il y a d'autres signaux numériques à droite qui ne seront pas abordés dans cette session.

[2] FM broadcasting. (16 janvier 2013). *Wikipedia, l'encyclopédie libre*. Daté du 17:13, 16 janvier 2013, sur http://en.wikipedia.org/w/index.php?title=FM_broadcasting&oldid=533382011

Objectifs

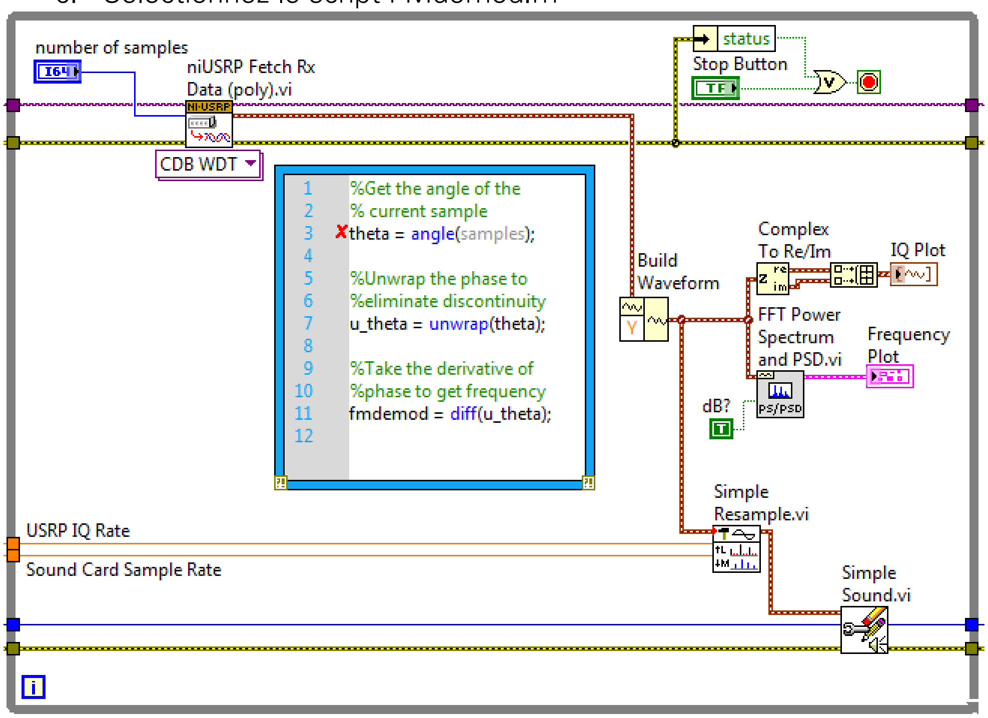
a) Utiliser un script de fichier .m pour démoduler une station de radio FM en utilisant MathScriptRT
b) Comprendre la multiplicité des modèles de calculs de LabVIEW

Partie B

1. Remplacez les fonctions de démodulation par un nœud MathScript
   a. Supprimez **Complex To Polar**, **Unwrap Phase**, **Derivative x(t)** et la constante **Backward**
   b. Appuyez sur <Ctrl-B> pour supprimer tous les fils de liaison brisés

2. Placez un nœud MathScript
   a. Effectuez un clic droit sur un espace blanc vide
   b. Naviguez jusqu'à **Programmation » Structures** et cliquez sur Nœud MathScript
   c. Placez le nœud MathScript à l'endroit où se trouvaient les fonctions de LabVIEW que vous venez de supprimer

3. Cliquez avec le bouton droit sur la bordure du nœud MathScript
   a. Cliquez avec le bouton droit sur la bordure du nœud MathScript et choisissez **Import**
   b. Naviguez jusqu'au dossier Exercises
   c. Sélectionnez le script FMdemod.m



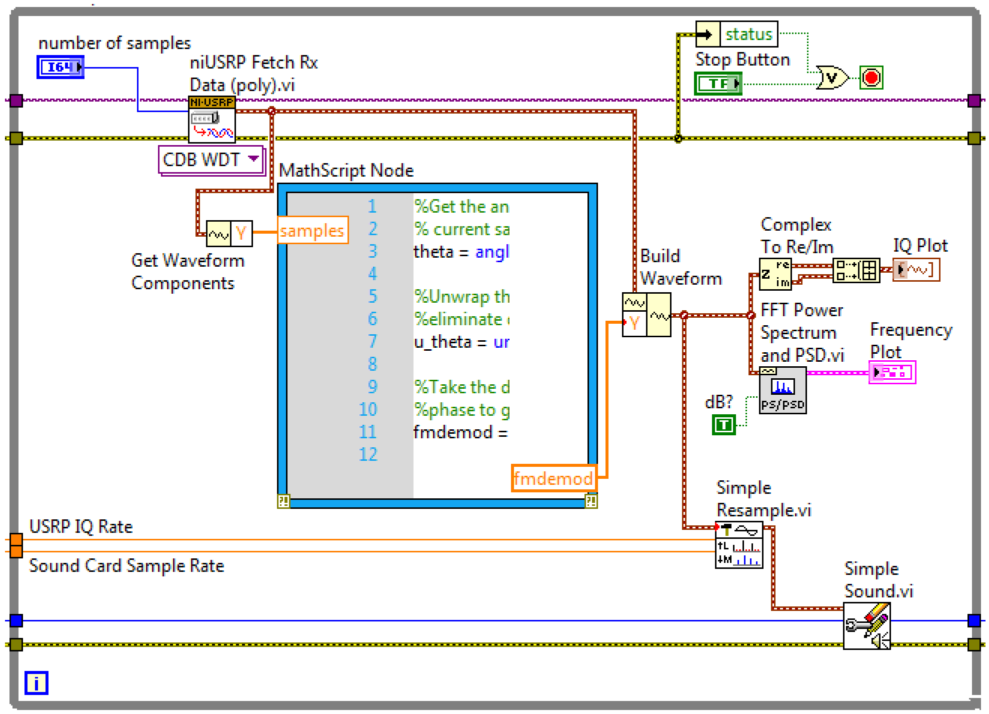
4. Cliquez avec le bouton droit sur la bordure gauche du nœud MathScript et choisissez **Ajouter une entrée**
   a. Dans la prochaine boîte de texte qui apparaît saisissez **samples**

5. Cliquez avec le bouton droit sur la bordure droite du nœud MathScript et cliquez sur **Add Output » fmdemod**

6. Naviguez jusqu'à la palette **Programmation » Waveform**
   a. Placez une fonction **Obtenir les composantes d'une waveform** à gauche de l'entrée des échantillons MathScript

b. Câblez la sortie **data** du VI **niUSRP Fetch Rx Data** à l'entrée de la fonction **Obtenir les composantes d'une waveform**
c. câblez la sortie **Y** à l'entrée **samples**

7. Câblez la sortie **fmdemod** à l'entrée **Y** de la fonction **Build Waveform**



8. Enregistrez le VI et nommez-le Exercise 2B

9. Revenez à la face-avant et exécutez le VI

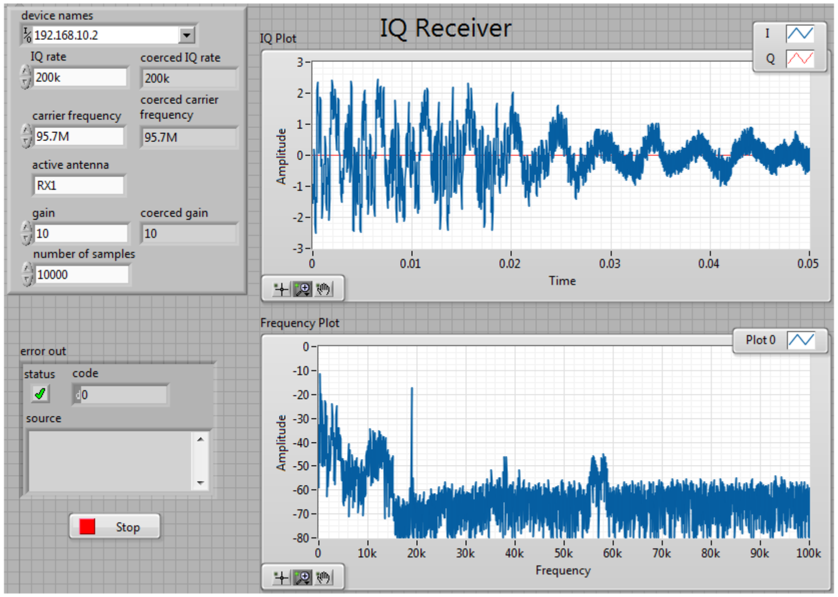


Figure 29. Face-avant achevée

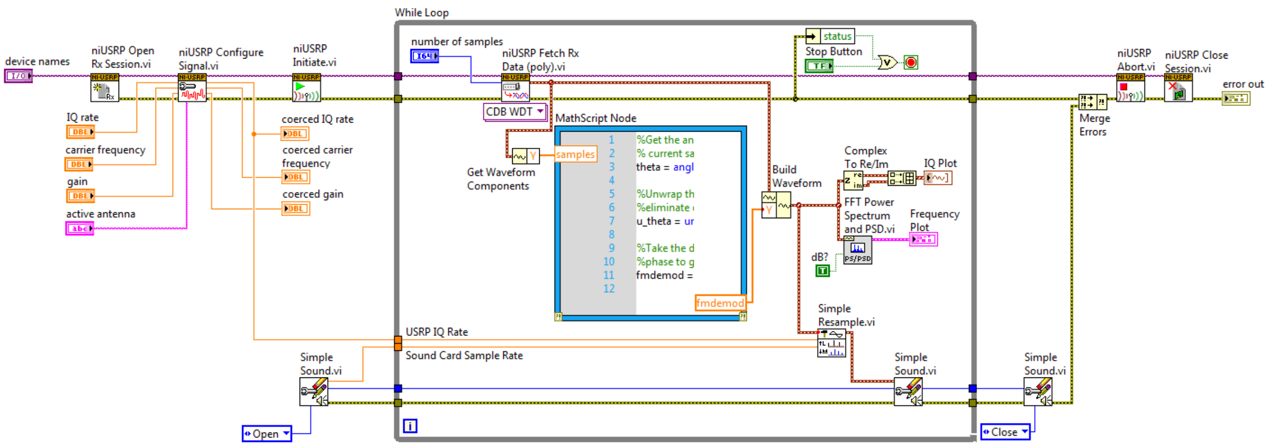


Figure 30. Diagramme achevé

Objectifs

a) Démoduler rapidement une station de radio FM à l'aide du Toolkit LabVIEW Modulation
b) Utiliser des propriétés intellectuelles (IP) existantes issues de toolkits pour accomplir rapidement des tâches

Partie C

1. Supprimez la fonction **Obtenir les composantes d'une waveform**, le nœud **MathScript** et la fonction **Build Waveform**

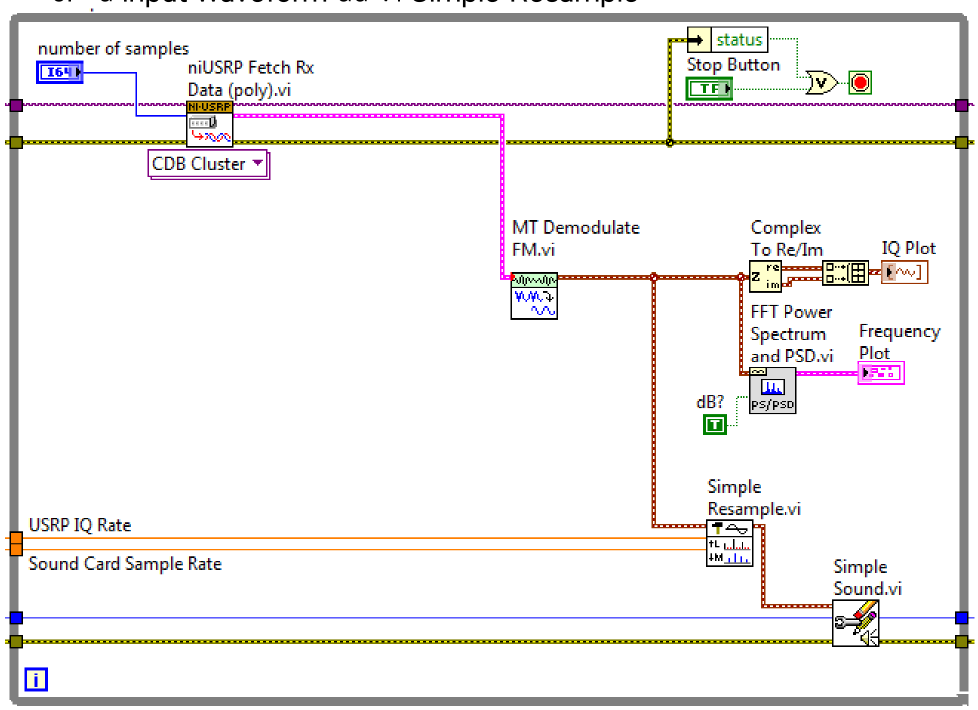
2. Appuyez sur <Ctrl-B> pour supprimer tous les fils de liaison brisés

3. Cliquez sur la flèche vers le bas du VI **niUSRP Fetch Rx Data**
    a. Sélectionnez **Single Channel » Complex Double Cluster**

4. Faites apparaître la palette des fonctions
    a. Naviguez jusqu'à **RF Communications » Modulation » Analog » Demodulation**
    b. Faites glisser la deuxième icône (**MT Demodulate FM**) sur le diagramme à l'endroit où se trouvait le nœud MathScript

5. Câblez la sortie **data** de **niUSRP Fetch Rx Data** à l'entrée **FM modulated waveform** du VI **MT Demodulate FM**

6. Câblez **FM demodulated waveform** à plusieurs blocs :
    a. à l'entrée **z** de la fonction **Complex To Re/Im**
    b. à l'entrée **signal temporel** du VI **FFT Power Spectrum and PSD**
    c. à **input waveform** du VI **Simple Resample**



7. Enregistrez le VI et nommez-le Exercise 2C

8. Revenez à la face-avant et exécutez le VI
Astuce : vous devrez peut-être effectuer un clic droit sur le tracé de la fréquence et mettre automatiquement à l'échelle l'axe des y

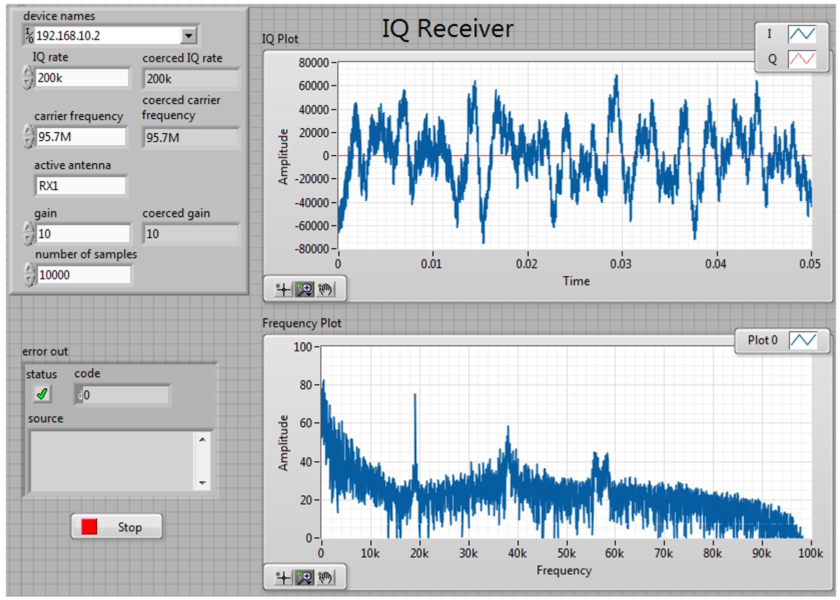


Figure 31. Face-avant achevée

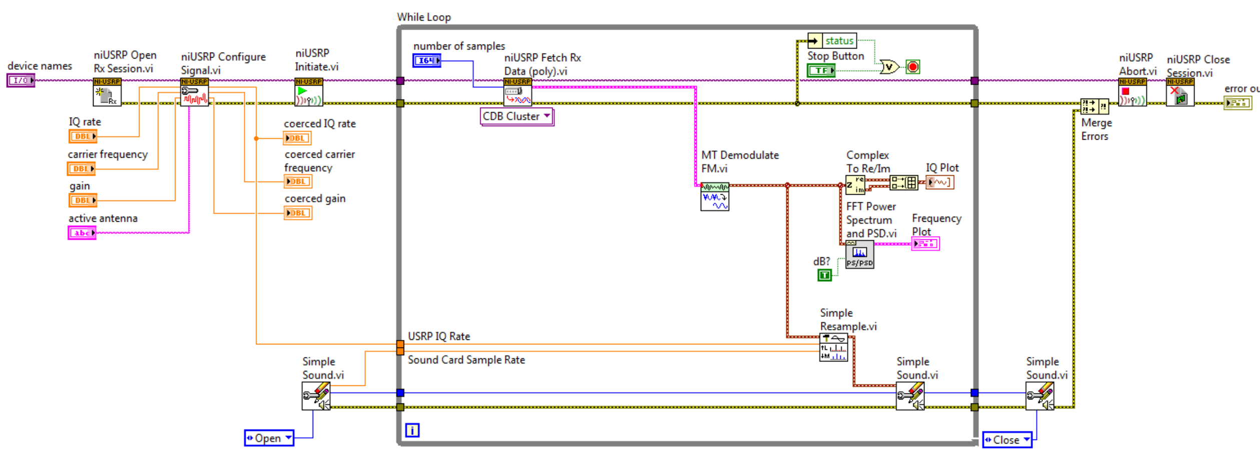
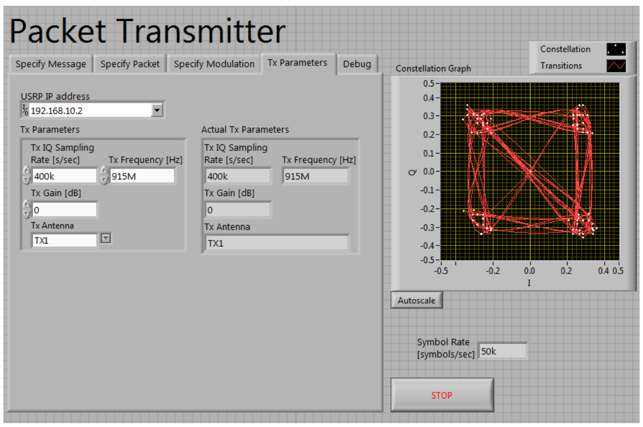


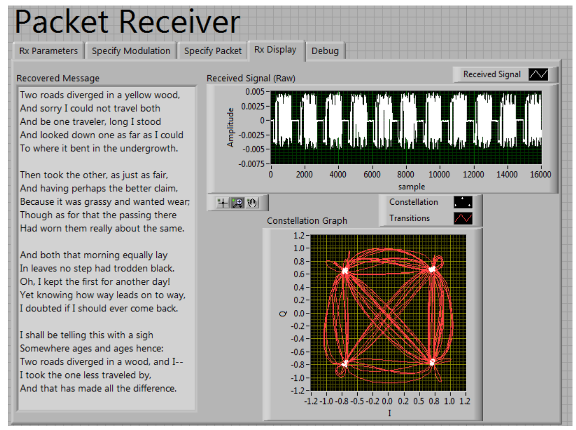
Figure 32. Diagramme achevé

Exercice 3 : Communications numériques

Remarque : cet exercice part du principe que vous avez déjà installé tous les logiciels et configuré tous les matériels.

Remarque : assurez-vous d'avoir connecté un câble de rebouclage et un atténuateur 30 db avant d'émettre tout signal.





Objectifs

a) Ouvrir et exécuter un exemple de communications numérique
b) Identifier les portions du diagramme et les éléments du processus de modulation/démodulation

Partie A

1. Faites équipe avec la personne ou le groupe à côté de vous pour décider qui va ouvrir le VI transmis et qui va ouvrir le VI reçu

2. Dans le dossier Exercises, ouvrez soit le VI **USRP Packet Receiver**, soit le VI **USRP Packet Transmitter**

3. Configurez l'émetteur
   a. Passez à l'onglet **Tx Parameters**
   b. Assurez-vous que les paramètres **IQ Sampling Rate** et **Frequency** sont identiques à ceux du récepteur
   c. Passez à l'onglet **Specify Modulation**
   d. Assurez-vous que le même schéma de modulation est utilisé pour la réception
   e. Passez à l'onglet **Specify Packet**
   f. Assurez-vous que la structure des paquets est la même sur le récepteur
   g. Cliquez sur l'onglet **Rx Display**

4. Configurez le récepteur
   a. Passez à l'onglet **Rx Parameters**
   b. Assurez-vous que les paramètres **IQ Sampling Rate** et **Frequency** sont identiques à ceux de l'émetteur
   c. Passez à l'onglet **Specify Modulation**
   d. Assurez-vous que le même schéma de modulation est utilisé pour l'émission
   e. Passez à l'onglet **Specify Packet**
   f. Assurez-vous que la structure des paquets est la même sur l'émetteur
   g. Cliquez sur l'onglet **Specify Message**

5. Cliquez sur la flèche d'exécution

Remarque : vous devez redémarrer le programme pour que les modifications apportées aux commandes de la face-avant soient effectives.

6. Cliquez pour revenir à l'onglet **Specify Modulation** et modifiez la modulation de **QPSK** à **BPSK** ou **8-PSK** et observez les modifications du **Constellation Graph** sur l'émetteur et le récepteur

Remarque : des schémas de modulation d'ordre élevé sont susceptibles de nécessiter des optimisations telles que la pré-distorsion numérique qui n'est pas abordée dans ce cours

7. Exécutez à nouveau ce VI avec le nouveau schéma de modulation

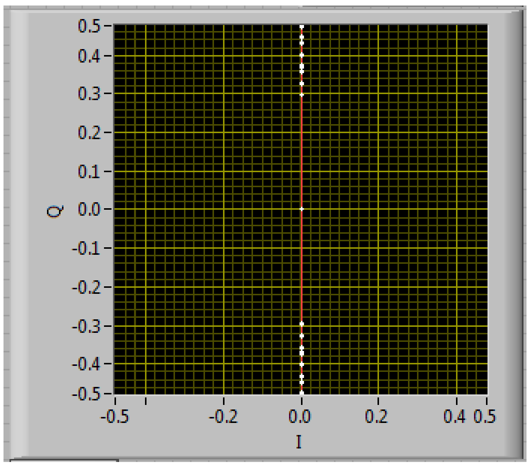


Figure 33. Signal numérique BPSK

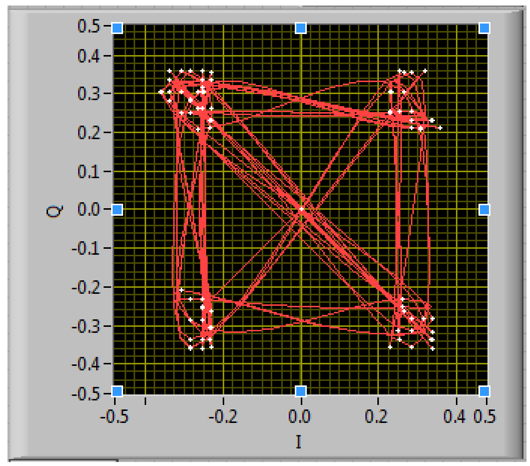


Figure 34. Signal numérique QPSK

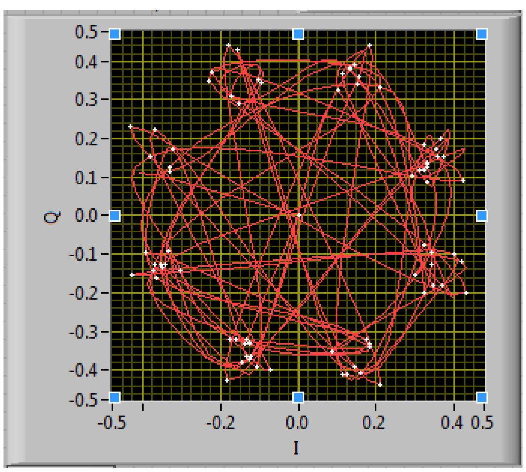


Figure 35. Signal numérique 8-PSK

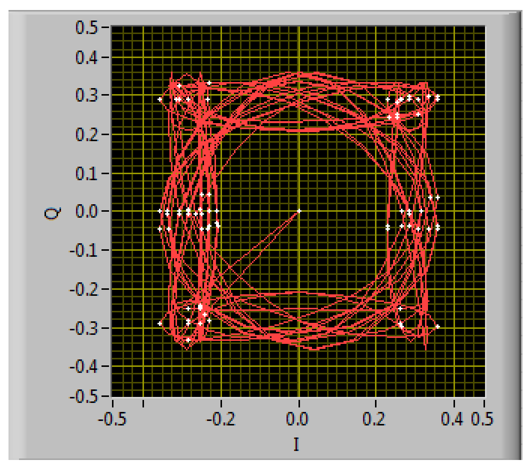


Figure 36. Signal numérique Offset QPSK

On Your Own Exercise 4: LabVIEW Tips and Tricks

Goals

a) Learn about time saving features in LabVIEW
b) Make LabVIEW graphs more interactive

Part A

This exercise should be completed on your own, after the hands-on class to give you more familiarity with the LabVIEW development environment

Tools Palette

By default, LabVIEW selects tools for you based on the context of your cursor. If you need more control over which tool is selected, use the Tools palette to select a specific tool to operate or to modify front panel and block diagram objects

1. Select **View » Tools Palette** to display the Tools palette

2. To display a temporary version of the Tools palette at the location of the cursor Press the <Shift> key and right-click

3. You can disable automatic tool selection by clicking the Automatic Tool Selection button on the Tools palette, shown as follows

   a. When you click the Automatic Tool Selection button to disable automatic tool selection, you can either select a tool on the palette or use the <Tab> key to move through the most commonly used tools in the sequence they appear on the palette. When you select a tool, the cursor changes to correspond to the icon of the tool

   b. To return to automatic tool section, press the <Tab> key or click the

Automatic Tool Selection button to enable automatic tool selection again

4. You can configure LabVIEW to disable automatic tool selection and move through the most commonly used tools whenever you press the <Tab> key and to toggle automatic tool selection only when you press the <Shift-Tab> keys or click the **Automatic Tool Selection** button
    a. Select **Tools » Options** to display the Options dialog box, select **Environment** from the Category list, and remove the checkmark from the **Lock automatic tool selection** checkbox

Automatically Wiring Objects

As you move a selected object close to other objects on the block diagram, LabVIEW draws temporary wires to show you valid connections. When you release the mouse button to add the object to the block diagram, LabVIEW automatically connects the wires. You also can automatically wire objects already on the block diagram. LabVIEW connects the terminals that best match and does not connect the terminals that do not match

1. Toggle automatic wiring by pressing the spacebar while you move an object using the Positioning tool. By default, automatic wiring is enabled when you select an object from the **Functions** palette or when you clone an object already on the block diagram. Automatic wiring is disabled by default when you use the Positioning tool to move an object already on the block diagram. When automatic wiring is enabled, the selected object retains its appearance when you drag it. When automatic wiring is disabled, the selected object appears as a dotted outline when you drag it
    a. If necessary, enable automatic wiring and set the maximum and minimum distances for objects to wire automatically
    b. Add a numeric control to the front panel window
    c. Press the <Ctrl-E> keys to display the block diagram. **(Mac OS X)** Press the <Command-E> keys. **(Linux)** Press the <Alt-E> keys
    d. Select the Add function from the **Functions** palette
    e. Move the left side of the function close to the right side of the numeric control terminal on the block diagram. As you move the function close to the terminal, LabVIEW draws temporary wires to show you valid connections. Make sure the function is within the maximum and minimum distances you set in step 1
    f. When a temporary wire appears between the numeric control terminal and the top left terminal of the Add function, click the mouse button to add the function and wire the objects
    g. Make sure you wire all required terminals. Otherwise, the VI is broken and will not run. Use the **context help** window to see which terminals a block diagram node requires. The labels of required terminals appear bold in the **Context Help** window

LabVIEW Graphs vs. Charts

Graphs and charts differ in the way they display and update data. VIs with a graph usually collect the data in an array and then plot the data to the graph. This process is similar to a spreadsheet that first stores the data then generates a plot of it. When the data is plotted, the graph discards the previously plotted data and displays only the new data. You typically use a graph with fast processes that acquire data continuously. In contrast, a chart appends new data points to those points already in the display to create a history. On a chart, you can see the current reading or measurement in context with data previously acquired. When more data points are added than can be displayed on the chart, the chart scrolls so that new points are added to the right side of the chart while old points disappear to the left. You typically use a chart with slow processes in which only a few data points per second are added to the plot

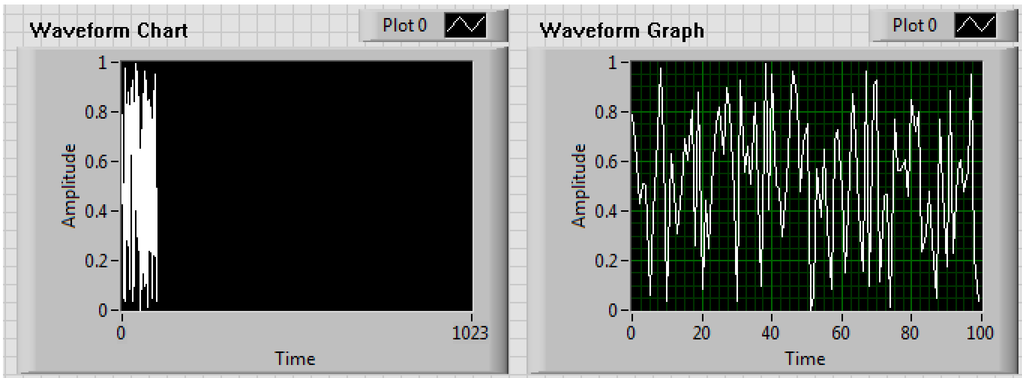


Figure 37. Waveform Chart and Graph after 1 Run

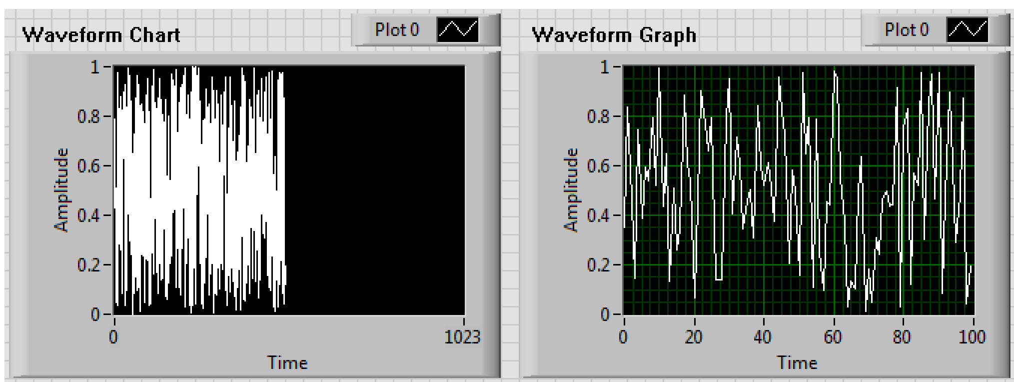
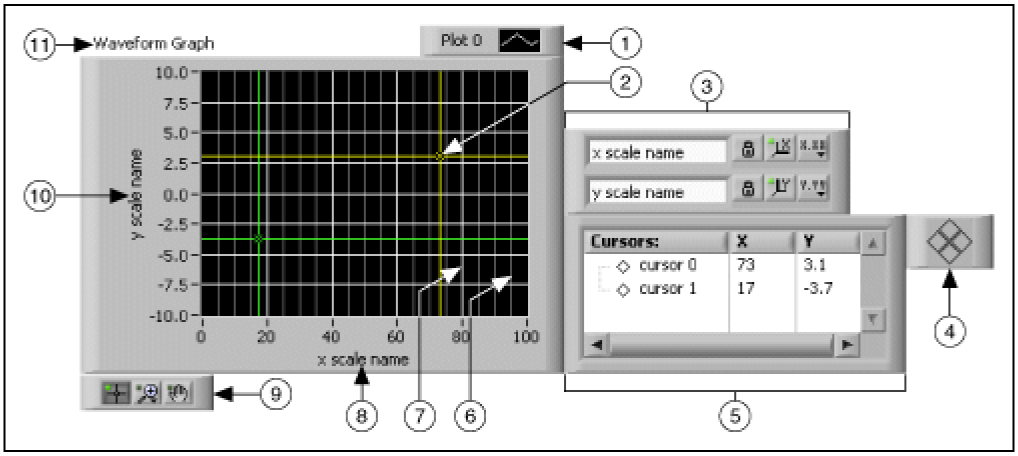


Figure 38. Waveform Chart and Graph after 5 Runs

Customizing Graphs

Each graph includes options that you can use to customize the graph to match your data display requirements. For example, you can modify the behavior and appearance of graph cursors or configure graph scales. The following illustration shows the elements of a graph



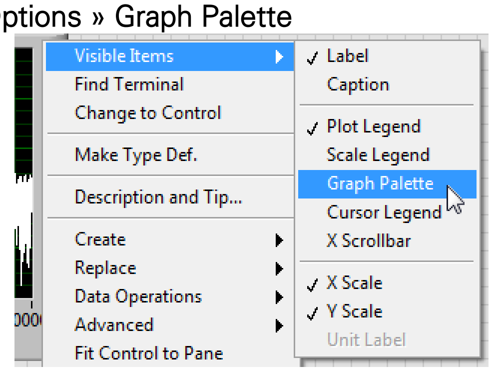
1	Plot Legend	6	Minor Grid Mark
2	Cursor	7	Grid Mark
3	Scale legend	8	X-scale
4	Cursor mover	9	Graph Palette
5	Cursor legend	10	Y-scale
		11	Label

You can add the plot legend, scale legend, cursor legend, graph palette, and label by right-clicking the graph, selecting **Visible Items** from the shortcut menu, and selecting the appropriate element

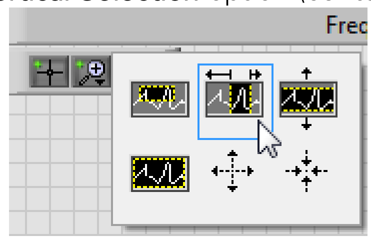
Working with Graphs

You should follow this exercise after running the program in Exercise 2 and stopping the program so there is data on your plot to work with

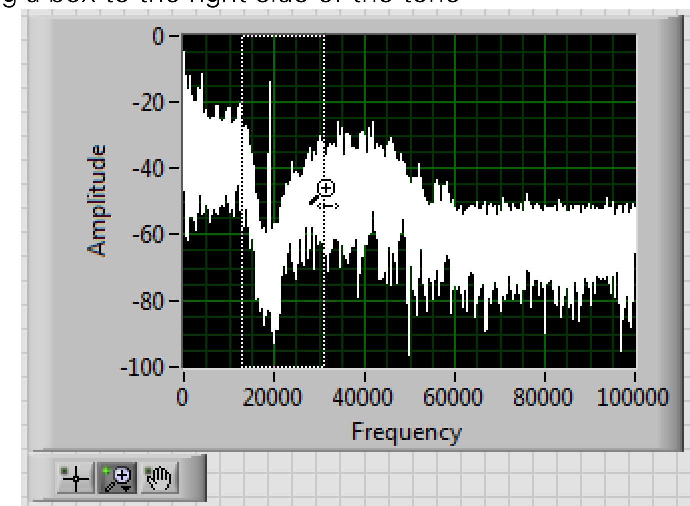
1. Add the graph palette to the waveform graph
   a. Right-click the graph
   b. Select **Visible Options » Graph Palette**



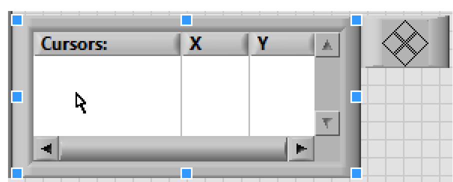
2. Zoom into the 19 kHz pilot tone
    a. Click the magnifier icon (center) to bring up the zoom menu
    b. Choose the **Zoom Vertical Selection** option (center top)



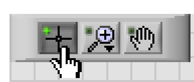
    c. Create a selection by clicking on the left side of the 19 kHz tone and dragging a box to the right side of the tone



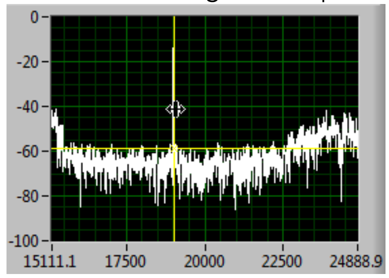
    d. We can repeat this procedure until we can get the X-axis to give us a better approximation of the frequency, but let's use a cursor instead
3. Show the **Cursor Legend** by right-clicking the waveform graph and navigating to **Visible Items**



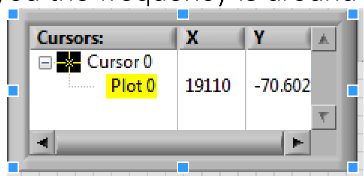
    a. Right-click inside the cursor palette and select **Create Cursor » Single-Plot** to create a cursor that is bound to the plot on the graph
    b. Change from the zoom tool to the select tool by clicking the first icon on the graph palette



c. Grab the vertical yellow line and drag it on top of the pilot tone



d. Does the cursor tell you the frequency is around 19 kHz?



Sometimes you may expect LabVIEW to work in a certain way but it doesn't. We just learned how to use the zoom vertical selection, to zoom in to a particular region of a waveform, but we did this while the program was not running

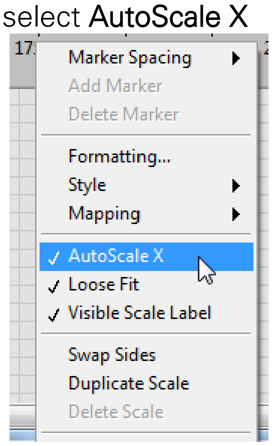
1. Click the run arrow to run the program

2. Change back to the zoom vertical selection tool

3. Try to zoom in. What happens? For a very small amount of time, the graph did zoom in, but it "reset." What actually happened is the x-axis and the y-axis *auto scaled*. To have the ability to zoom in to the waveform as the program is running you should turn off auto scaling

4. Right click the x-axis numbers

5. From the shortcut menu, unselect **AutoScale X**



6. Do the same for the y-axis

7. Try to zoom back in with the program still running. Does it work now?

Additional Training

Self-Paced Video Training for Students

Use this set of step-by-step tutorials, example projects, and short videos to get started with NI tools. Begin by learning how to work within the LabVIEW environment. Understand how to work within the NI LabVIEW software environment. Watch video modules to learn the fundamental building blocks of programming in LabVIEW. Test your understanding of LabVIEW concepts with the LabVIEW Basics Test and assess your understanding of programming within the LabVIEW environment by completing the LabVIEW Basics Exercise.

Visit ni.com/academic/students/learn-labview to learn the basics of LabVIEW today!

Learn LabVIEW

To begin, you need to understand how to work within the NI LabVIEW software environment. Watch these video modules to learn the fundamental building blocks of programming in LabVIEW.

- Free LabVIEW Student Edition for 6 Months
- Purchase LabVIEW (Special Student Pricing)

Learn LabVIEW

1. LabVIEW Environment
Learn how to navigate LabVIEW and use the block diagram, front panel, and Functions and Controls palettes. Then explore how to use graphs and charts and build a user interface.

2. Loops and Execution Structures
See how to loop code and conditionally execute code in LabVIEW using For Loops, While Loops, and Case structures.

3. Data Types and Structures
Explore the different data types and methods to organize and group data, controls, and indicators in LabVIEW.

4. Graphical Programming
Examine the basics of graphical programming and methods to determine data flow and order of execution in the LabVIEW environment.

5. Programming Tools
Discover how to use important tools in LabVIEW that can save you time.

6. Debugging and Handling Errors
Learn how to use key debugging and troubleshooting tools that optimize code.

7. Signal Processing
Discover how some applications require analyzing the frequency components of signals and learn how to filter and perform an FFT on an array of data in LabVIEW.

8. MathScript and Text-Based Programming
Examine how to reuse existing .m file or C++ based code for the processing and scaling of signals.

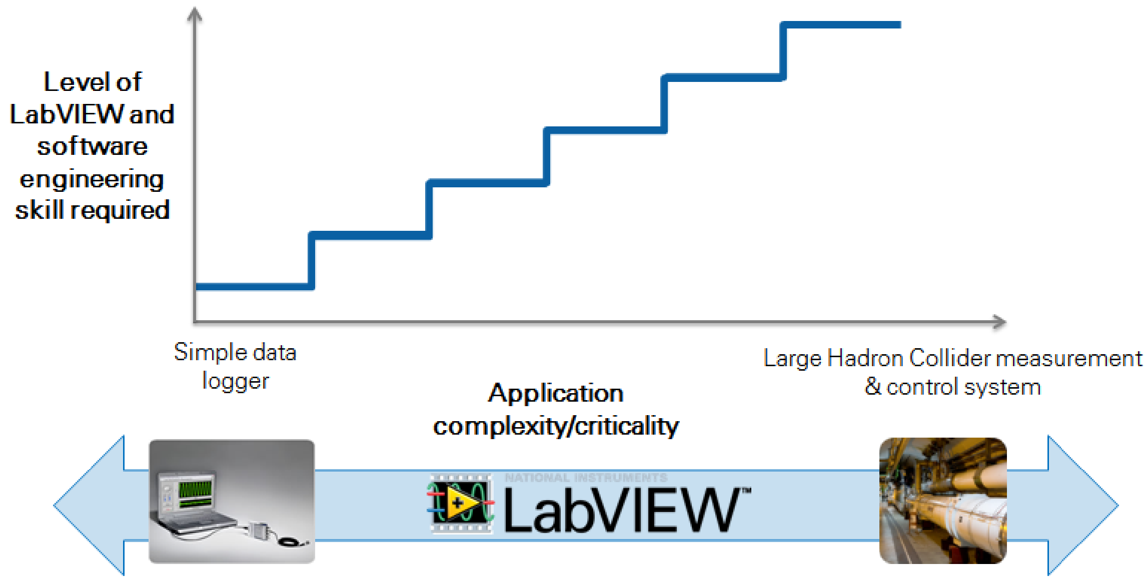
9. Help Within LabVIEW
Learn how to use Help resources in LabVIEW to better understand concepts and coding techniques.

Cumulative Exam: LabVIEW Basics
Test your understanding of LabVIEW concepts by attempting the LabVIEW Basics Test. This test is recommended after completing the modules found in the Learn LabVIEW section. The test includes 20 multiple choice questions.
Begin the exam

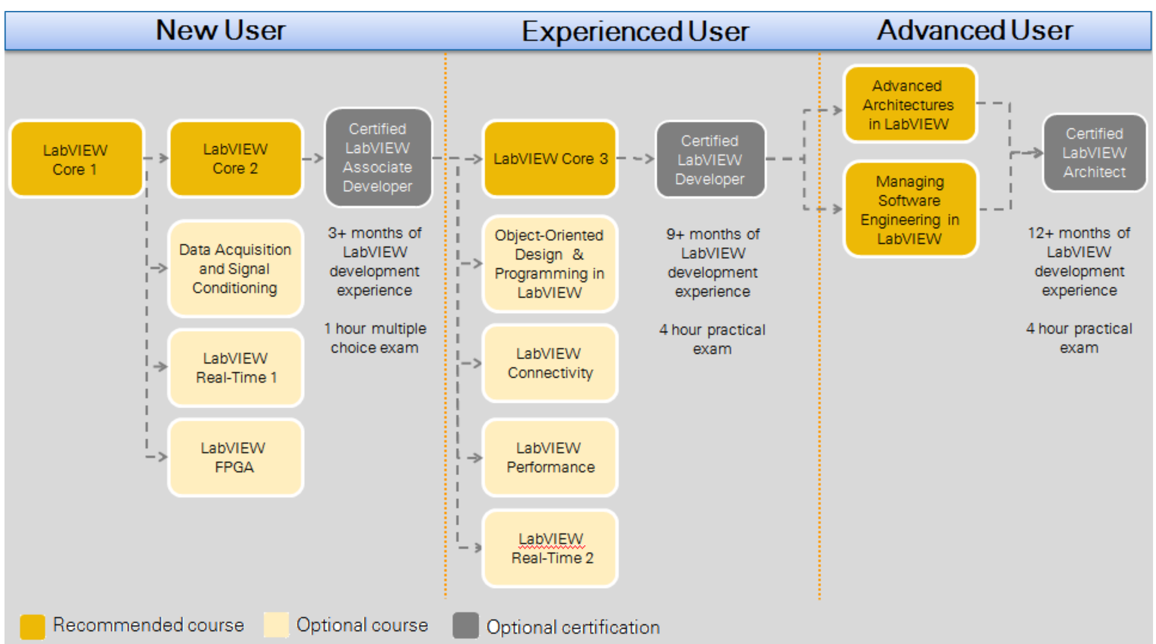
Cumulative Exercise: LabVIEW Basics
Assess your understanding of programming within the LabVIEW environment by completing the LabVIEW Basics Exercise. This exercise is recommended after completing the modules found in the Learn LabVIEW section. The exercise involves building a calculator in LabVIEW.
Start the exercise

Professional Training Options

There are many courses that we offer for training in LabVIEW, as well as our Data Acquisition and other hardware platforms. We also offer three levels of LabVIEW certification along with a few other certification paths. These can help to boost your resume, and also solidify your knowledge of NI software and hardware.



NI Certification Path



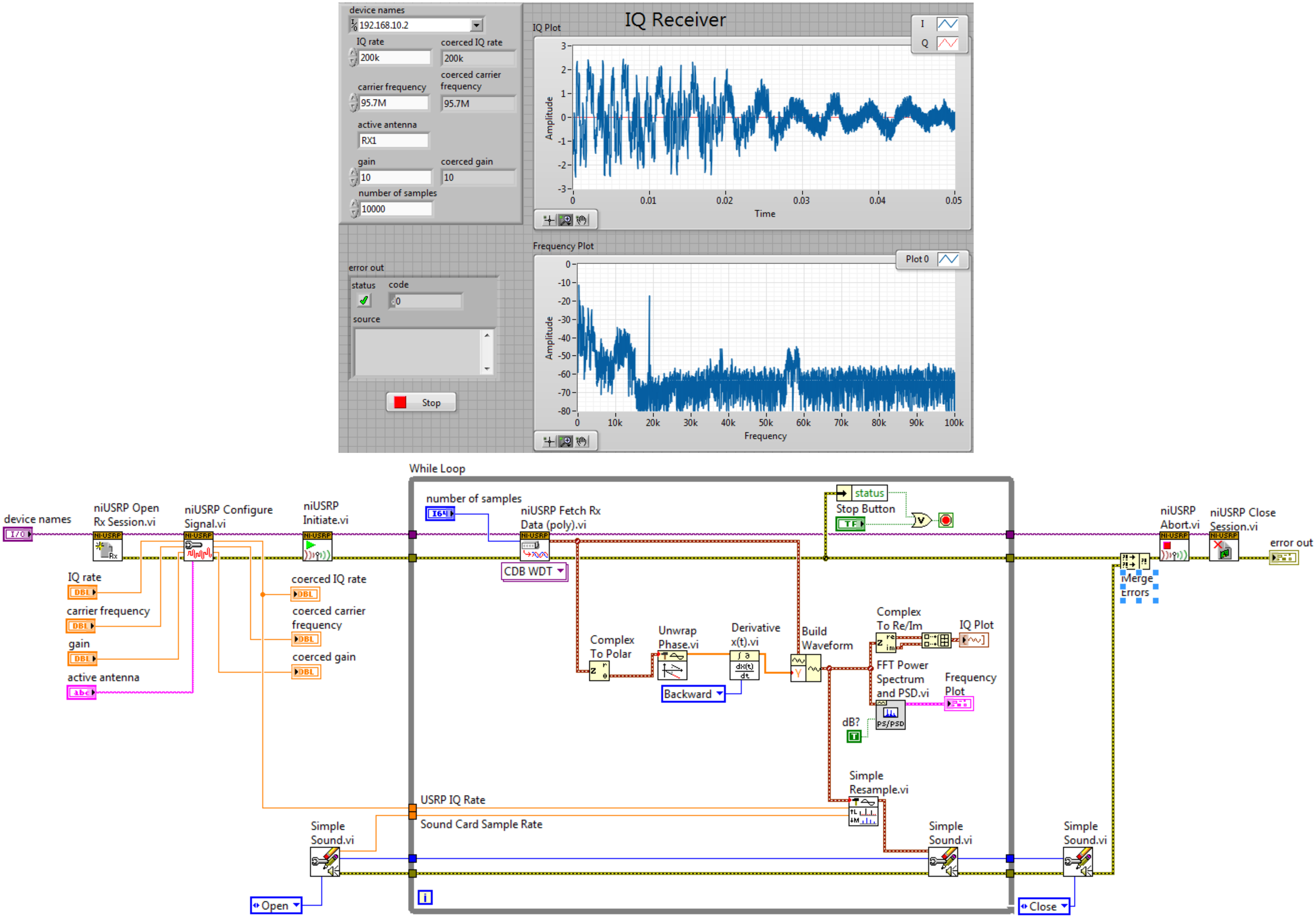
For more information, visit ni.com/training today. Be sure to ask about our special pricing for academic customers.

Appendix A: Exercise Solution Screenshots

Exercise 1

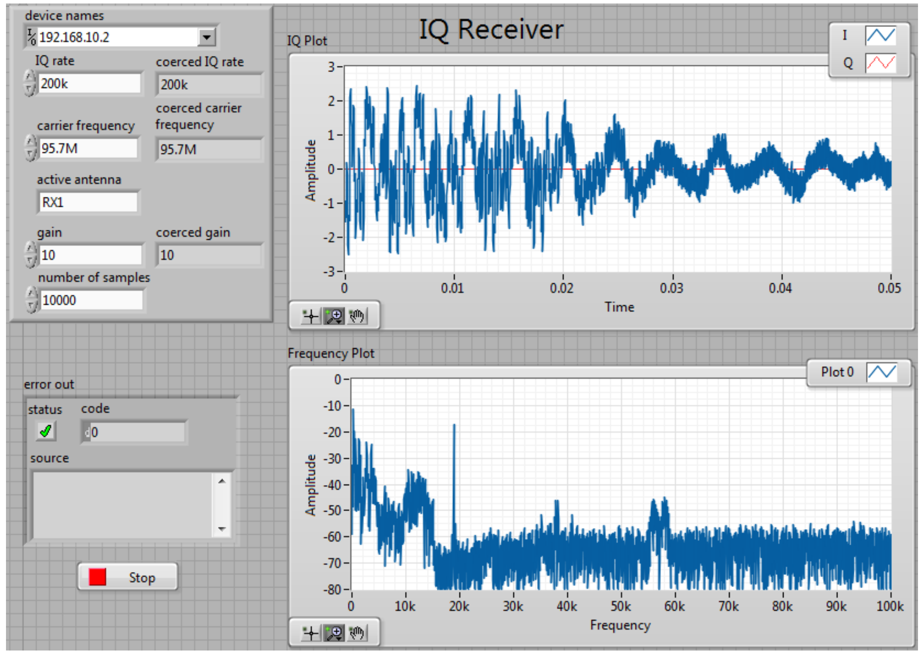
No programming necessary

Exercise 2 A

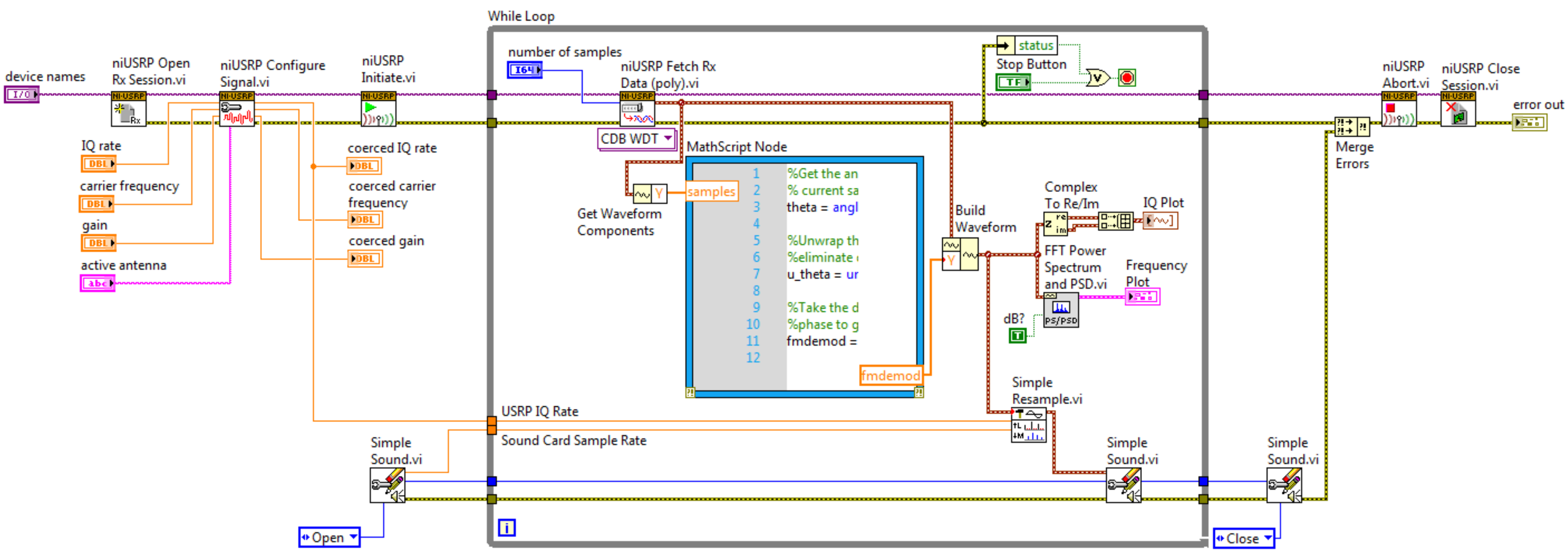


device names
192.168.10.2
IQ rate
200k
coerced IQ rate
200k
carrier frequency
95.7M
coerced carrier frequency
95.7M
active antenna
RX1
gain
10
coerced gain
10
number of samples
10000
IQ Receiver
IQ Plot
Amplitude
Time
Frequency Plot
Plot 0
Frequency
error out
status
code
source
Stop
While Loop
number of samples
niUSRP Fetch Rx Data (poly).vi
CDB WDT
status
Stop Button
niUSRP Open Rx Session.vi
niUSRP Configure Signal.vi
niUSRP Initiate.vi
IQ rate
carrier frequency
gain
active antenna
coerced IQ rate
coerced carrier frequency
coerced gain
Complex To Polar
Unwrap Phase.vi
Backward
Derivative x(t).vi
Build Waveform
Complex To Re/Im
IQ Plot
FFT Power Spectrum and PSD.vi
Frequency Plot
dB?
Simple Resample.vi
USRP IQ Rate
Sound Card Sample Rate
Simple Sound.vi
Open
Simple Sound.vi
Simple Sound.vi
Close
Merge Errors
niUSRP Abort.vi
niUSRP Close Session.vi
error out

Exercise 2 B

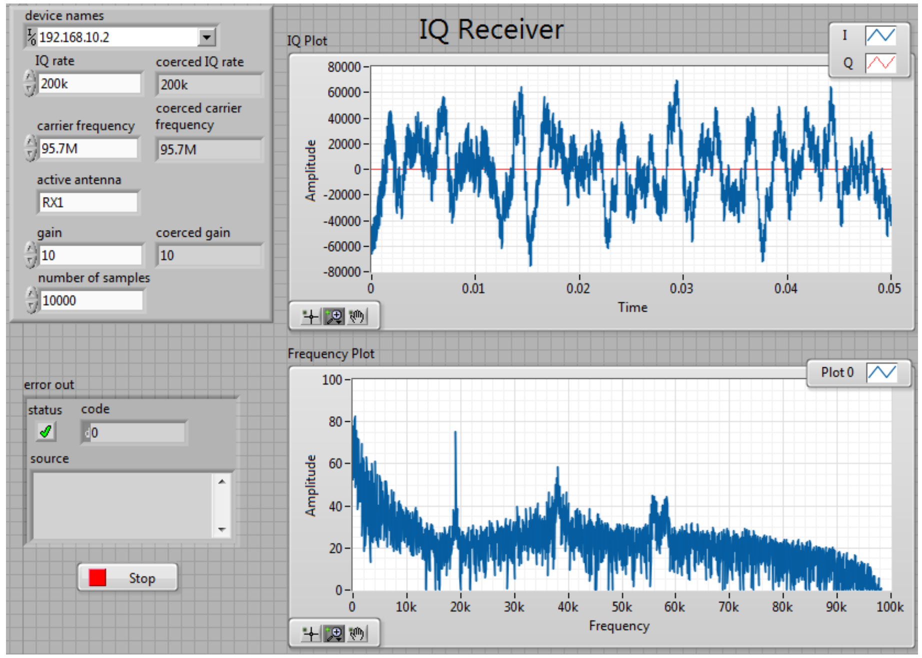


device names
192.168.10.2
IQ rate
200k
coerced IQ rate
200k
carrier frequency
95.7M
coerced carrier frequency
95.7M
active antenna
RX1
gain
10
coerced gain
10
number of samples
10000
error out
status
code
0
source
Stop
IQ Receiver
IQ Plot
I
Q
Amplitude
Time
Frequency Plot
Plot 0
Amplitude
Frequency

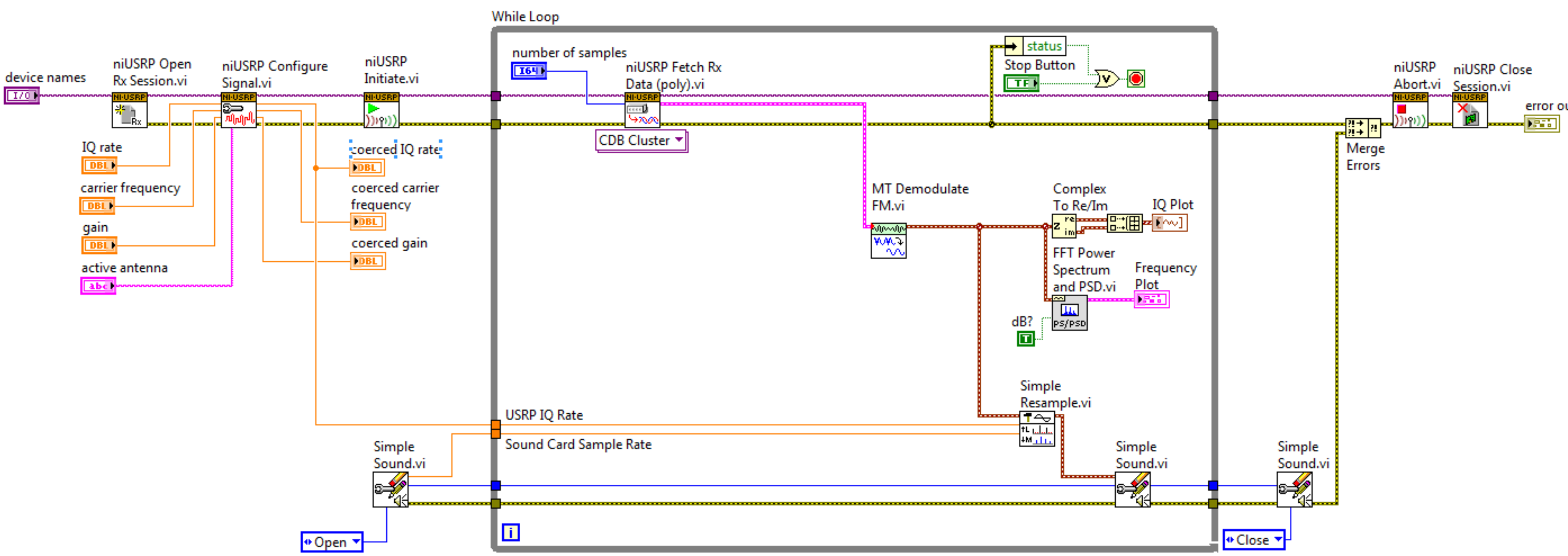


While Loop
device names
niUSRP Open Rx Session.vi
niUSRP Configure Signal.vi
niUSRP Initiate.vi
number of samples
niUSRP Fetch Rx Data (poly).vi
status
Stop Button
niUSRP Abort.vi
niUSRP Close Session.vi
error out
IQ rate
carrier frequency
gain
active antenna
coerced IQ rate
coerced carrier frequency
coerced gain
CDB WDT
MathScript Node
Get Waveform Components
samples
%Get the an
% current sa
theta = angl
%Unwrap th
%eliminate
u_theta = ur
%Take the d
%phase to g
fmdemod =
fmdemod
Build Waveform
Complex To Re/Im
IQ Plot
FFT Power Spectrum and PSD.vi
Frequency Plot
dB?
Merge Errors
Simple Resample.vi
USRP IQ Rate
Sound Card Sample Rate
Simple Sound.vi
Simple Sound.vi
Simple Sound.vi
Open
Close

Exercise 2 C



device names
192.168.10.2
IQ rate
200k
coerced IQ rate
200k
carrier frequency
95.7M
coerced carrier frequency
95.7M
active antenna
RX1
gain
10
coerced gain
10
number of samples
10000
error out
status
code
0
source
Stop
IQ Receiver
IQ Plot
I
Q
Amplitude
Time
Frequency Plot
Plot 0
Amplitude
Frequency



While Loop
device names
niUSRP Open Rx Session.vi
niUSRP Configure Signal.vi
niUSRP Initiate.vi
number of samples
niUSRP Fetch Rx Data (poly).vi
CDB Cluster
status
Stop Button
IQ rate
carrier frequency
gain
active antenna
coerced IQ rate
coerced carrier frequency
coerced gain
MT Demodulate FM.vi
Complex To Re/Im
IQ Plot
FFT Power Spectrum and PSD.vi
Frequency Plot
dB?
Simple Resample.vi
USRP IQ Rate
Sound Card Sample Rate
Simple Sound.vi
Open
Simple Sound.vi
Simple Sound.vi
Close
niUSRP Abort.vi
niUSRP Close Session.vi
error out
Merge Errors

Exercise 3

No programming necessary

Exercise 4

No programming necessary

Glossary A: RF & Communication Reference

Symbols

º	Degrees
dB	Decibel
Hz	Hertz
Ω	Ohms
/	per
V	Volts
W	Watts

A

Amplitude Modulation (AM)	A process that varies the amplitude of a radio frequency (RF) carrier signal according to the amplitude of the message signal.
Amplitude-Shift Keying (ASK)	Refers to a type of amplitude modulation, which assigns bit values to discrete amplitude levels. The carrier signal is then modulated among the members of a set of discrete values to transmit information.
analog signal	An **analog signal** is any continuous signal for which the time varying feature (variable) of the signal is a representation of some other time varying quantity.[3]
ADC	**Analog-to-Digital Converter**—A hardware component that converts analog voltages to digitized values. An ADC can convert an analog signal to a digital signal representing equivalent information.

[3] Analog signal. (2013, January 11). In *Wikipedia, The Free Encyclopedia*. Retrieved 18:33, January 16, 2013, from http://en.wikipedia.org/w/index.php?title=Analog_signal&oldid=532543697

B

bandwidth	The measure of a circuit or transmission channel to pass a signal without significant attenuation over a range of frequencies. Bandwidth can also refer to the information rate (in bits per second) that can pass through a circuit or transmission channel.
baseband signal	The baseband is the range in the frequency spectrum occupied by the unmodulated message signal. Both the message signal and the downconverted complex I/Q signal are referred to as baseband signals.
binary signal	A signal that carries information by varying between two possible values, corresponding to 0 and 1 in the binary system.

C

carrier	The signal that carries the information encoded or modulated on it. Typically, the carrier is a fixed frequency sine wave, which can be amplitude-, phase-, or frequency-modulated.
carrier frequency	The frequency of the carrier signal that is a sinusoidal signal upon which the desired signal to be transmitted is modulated. The sinusoidal signal "carries" the modulation.
Carson's rule	Defines the approximate modulation bandwidth required for a carrier signal that is frequency-modulated by a spectrum of frequencies rather than a single frequency. The Carson bandwidth rule is expressed by the relation CBR = $2(\Delta f + f_m)$ where CBR is the bandwidth requirement, Δf is the carrier peak deviation frequency, and f_m is the highest modulating frequency.
CCDF	The **Complementary Cumulative Distribution Function** (CCDF) is a statistical characterization of the time-domain waveform that completely describes the power characteristics of a signal.
center frequency	The middle frequency of the channel bandwidth. In frequency modulation, the center frequency is equal to the *rest frequency*—specifically, the frequency of the unmodulated carrier wave.
complex envelope	A complex representation of the baseband modulated signal.

component (I/Q) — The real (I—in-phase) and imaginary (Q—quadrature) parts of a complex number are referred to as *components*. The Modulation Toolkit VIs can use complex components to describe signal properties. Use the NI-USRP functions Write Tx Data VI and NI-USRP Fetch Data VI in applications that use Modulation Toolkit VIs, because they also use complex components.

For example, you can represent a two-dimensional vector of length S by its components $S = A + iB$, where A and B are the vector x- and y-components. The real part of the vector corresponds to the x-component (A), while the imaginary part corresponds to the y-component (B).

D

DC offset — A complex signal impairment that shifts the locus of ideal symbol coordinates off-center in the I/Q plane. A DC offset can be added to the baseband I component, the Q component, or both. The DC offset can be either positive or negative, with the sign indicating direction of the shift. DC offset is expressed as a percentage of full scale, where "full scale" is the amplitude of the baseband QM waveform.

DAC — Digital-to-analog converter—An electronic device, often an integrated circuit, that converts a digital number into a corresponding analog voltage or current.

Decoding — Data decoding involves removing redundant bits from the sequence and correcting for any errors that might have happened during transmission. The signal decoding process is usually more complicated than the encoding process and can be very computationally intensive.

demodulation — Describes the recovery from a modulated wave of a signal having the same characteristics as the original message signal.

The down converted signal undergoes a demodulation process. This step is the opposite of modulation and refers to the process required to extract the original information signal from the modulated signal

dB, decibel — The unit for expressing a logarithmic measure of the ratio of two signal levels: $dB = 20 \log_{10}(V1/V2)$, for signals in volts.

dBm — decibel milliwatt—Absolute power level referenced to 1 mW.

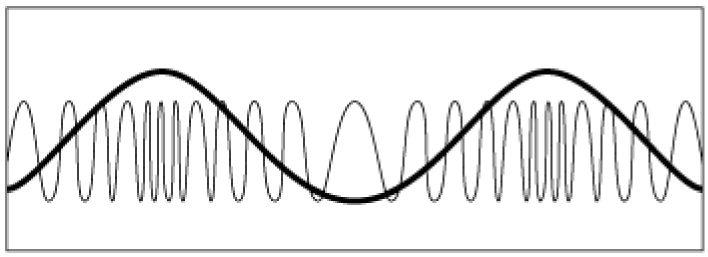
downconverter	A signal conditioning device that converts a specific band of high-frequency (RF) signals to lower, more manageable intermediate frequencies (IF) that can be digitized.
Down conversion	The first step in the demodulation process is down conversion from a real passband waveform to a complex I/Q baseband waveform. This process involves mixing the real-valued passband waveform with a locally generated carrier tone, followed by lowpass filtering to generate the I/Q baseband waveform.
DSP	Digital Signal Processing—The computation of signal or system transfer characteristics using numeric algorithms. Examples of areas where DSP techniques may be applied include: digital filters, echo detection or echo cancellation, speech synthesis, FFT for spectrum analysis, correlation computations, imagine recognition, and servo-feedback control.

E

encoding	A data source generates the information signal sent to a particular receiver. This signal may be either an analog signal, such as speech, or a digital signal, such as a binary data sequence. The information signal is typically a baseband signal represented by a voltage level.
equalization	In an adaptive feed-forward equalizer, the taps of a filter that acts as an equalizer that continuously adapts its coefficients to compensate for the action of the channel filter. At the start of the equalization process, you typically supply training bits to train the equalizer. After training, the equalizer switches to decision-directed feedback mode, where the equalizer trains itself based on its own decisions.

F

fetch	Process that transfers data from device onboard memory to PC memory.
filtering	In a digital communication system, digital information can be sent on a carrier through changes in its fundamental characteristics such as phase, frequency, and amplitude. In a physical channel, these transitions can be smoothed, depending on the filters implemented during transmission. In fact, filters play an important part in a communications channel because they can eliminate spectral leakage, reduce channel width, and eliminate adjacent symbol interference known as inter-symbol interference (ISI).

FIR filter	**Finite Impulse Response (FIR)** is a term that describes a filter with no feedback elements, hence, its impulse response is finite. In contrast, IIR (infinite impulse response) circuitry does use feedback. FIR filters can be implemented by using analog or digital shift registers, or by using software algorithms. For a tutorial of IIR and FIR filters, search for *digital filter* in the NI Developer Zone.
FFT	The **Fast Fourier Transform (FFT)** is an efficient algorithm often used for spectrum analysis.
FM	**Frequency Modulation (FM)** is a process that varies the frequency of a sinusoidal carrier wave from a center frequency by an amount proportional to the instantaneous value of the message signal. In FM, the center frequency is the carrier frequency. 
FPGA	A **Field-Programmable Gate Array (FPGA)** is a semi-conductor device which contains a large quantity of gates (logic devices), which are not interconnected, and whose function is determined by a wiring list, which is downloaded to the FPGA. The wiring list determines how the gates are interconnected, and this interconnection is performed dynamically by turning semiconductor switches on or off to enable the different connections. Search for ***FPGA*** in the NI Developer Zone for a more complete explanation.
frequency	**Frequency** refers to a basic unit of rate measured in events or oscillations per second. Frequency also refers to a number representing a specific point in the electromagnetic spectrum. Frequency is measured as the number of cycles per unit time. The SI unit of measure for frequency is Hertz (Hz) where 1 Hz equals one cycle per second.
frequency span	Typically refers to a range of frequencies, for example from 10 kHz to 20 kHz. This term often describes the range that an instrument, such as a spectrum analyzer, is set to measure, or the range over which a set of frequencies of interest are located. For example, an FM modulated signal may cover a span from 100.5 MHz to 100.7 MHz.

G

gain	The factor by which a signal is amplified, often expressed in dB. **Gain** as a function of frequency is commonly referred to as the magnitude of the frequency response function.

H

Hertz, Hz	**Hz** is the SI unit for measurement of frequency. One **Hertz** (Hz) equals one cycle per second. It can be used to represent the number of scans read or updates written per second.

I

IIR filter	**Infinite Impulse Response Filter**—A filter that is designed using feedback or storage elements so that its impulse response will, in principle, last forever. In contrast, FIR filters have a finite impulse response.
impairments	All transmission media (including wireless, fiber optic, and copper) introduce some form of distortion (**impairments**) to the original signal. Different types of channel models have been developed to mathematically represent such real-world distortions.
in-phase signal	The **I (in phase)** component of a signal is typically the real component of a complex-valued signal.
I/Q data	The translation of the magnitude and phase data of a signal from a polar coordinate system to a complex Cartesian (X,Y) coordinate system. The I (In-phase) component is typically the real-valued component of the signal, while the Q component is typically the imaginary component of the signal.
I/Q modulation	**In-Phase/Quadrature Modulation (I/Q modulation)** is a modulation technique where a signal is modulated by two signals 90 degrees out of phase with each other.
I/Q signal	A control signal for changing an RF carrier signal.
IF	**Intermediate Frequency (IF)**—In radio receivers or spectrum analyzers, the original high-frequency signal is often mixed to an intermediate frequency before demodulation.
information signal	Contains the data for transmission. The information signal is used to

modulate the carrier wave to create the modulated wave for transmission. The information signal data is recovered from the modulated wave by a process of demodulation.

The information signal is often referred to as the baseband signal or *message signal*.

impedance	The electrical characteristic of a circuit that opposes flow of current through that circuit. It can be expressed in ohms and/or capacitance/inductance.
intersymbol interference	In telecommunication, **intersymbol interference** (**ISI**) is a form of distortion of a signal in which one symbol interferes with subsequent symbols. This is an unwanted phenomenon as the previous symbols have similar effect as noise, thus making the communication less reliable. ISI is usually caused by multipath propagation or the inherent non-linear frequency response of a channel causing successive symbols to "blur" together.[4]
ISM	The **Industrial, Scientific and Medical (ISM)** bands are low-power radio frequencies for industrial, scientific, and medical processes, and license-free wireless communication.

L

Local Oscillator (LO)	**Local Oscillator** refers to an internal oscillator in a radio, receiver, or instrument that tunes the frequency of the device. The local oscillator is mixed with a fixed frequency oscillator, which generates a sum and difference component.

M

message signal	Contains the data for transmission. The **message signal** is used to modulate the carrier wave to create the modulated wave for transmission. The message signal data is recovered from the modulated wave by a process of demodulation. The message signal is often referred to as the baseband signal or *information signal*.

[4] Intersymbol interference. (2012, December 15). In *Wikipedia, The Free Encyclopedia*. Retrieved 18:54, January 16, 2013, from http://en.wikipedia.org/w/index.php?title=Intersymbol_interference&oldid=528200950

MIMO	**Multiple input, multiple output**—A measurement technique, most often used in acoustics and vibration, to identify signal paths and frequency response functions from multiple inputs to multiple outputs. Associated with this measurement class are the techniques of partial and multiple coherence, which help identify which parts of an output signal are due to a specific input signal or combinations of signals.
modulation	A process, or the result of a process, by which characteristics of a carrier wave are altered according to information in the baseband signal to generate a modulated wave that is transmitted. In a diagram of the components of a Software Defined Radio system, the modulation block converts the information signal bit stream into in-phase (I) and quadrature phase (Q) data components. This block typically also involves pulse shaping to minimize intersymbol interference and reduce bandwidth.

P

passband	The range of frequencies that a device can properly propagate or measure.
phase	The fraction of the wave cycle which has elapsed relative to the origin[5]
phase-locked loop (PLL)	An electronic circuit that controls an oscillator so that the circuit maintains a constant phase angle relative to a reference signal.
PPS	Pulse Per Second
pulse-shaping filter	By applying a pulse-shaping filter to the modulated sinusoid, sharp transitions in the signal are smoothed and the resulting signal is limited to a specific frequency band. In addition, pulse-shaping filters are applied to communication signals to reduce intersymbol interference.

[5] Phase (waves). (2012, December 25). In *Wikipedia, The Free Encyclopedia*. Retrieved 19:02, January 16, 2013, from http://en.wikipedia.org/w/index.php?title=Phase_(waves)&oldid=529664818

Q

quadrature signal	The Q (**quadrature**) component of a complex-valued signal is typically the imaginary-valued component of the signal.
QAM	**Quadrature-Amplitude Modulation (QAM)** is a form of quadrature modulation in which the two carriers are both amplitude-modulated.

R

radio frequency (RF)	Refers to the radio frequency range of the electromagnetic spectrum. RF is often used to describe a range of sub-infrared frequencies from the tens of MHz to several GHz.
reference clock	Clock to which a device phase locks another, usually faster, clock. A common source for the Reference clock is the 10 MHz oscillator present on the PXI backplane.
RX, Rx	Receive data or signals. RX refers to the hardware receiver; Rx refers to receive operations in software.

S

sample rate	The rate at which a device acquires an analog signal, expressed in samples per second (S/s). The sample rate is typically the clock speed of the analog-to-digital converter (ADC).
SDR	**Software-Defined Radio**
SerDes	**Serializer/Deserializer**
SFDR	**Spurious Free Dynamic Range**—The separation or distance, expressed in dB, from the amplitude of the fundamental frequency and the next highest spur.
SMA	A small type of threaded coaxial signal connector typically used in higher frequency applications.
symbol rate	Expresses the number of symbols transmitted per second (symbols/s). To convert symbol rate into bit rate, which expresses the number of bits transferred per second, multiply the symbol rate by the number of bits per symbol used in the digital modulation scheme of interest. Symbol rate is

also known as *baud rate*.

T

TTL	**Transistor-Transistor Logic**—A digital circuit composed of bipolar transistors wired in a certain manner. A typical medium-speed digital technology. Nominal TTL logic levels are 0 and 5 V.
TX, Tx	Transmit data or signals. **TX** refers to the hardware transmitter; **Tx** refers to transmit operations in software.

U

UHD	**Universal Hardware Driver**, a software driver that is called on by the LabVIEW driver to provide control over a connected USRP device.
USRP	**Universal Software Radio Peripheral**—A computer-hosted hardware peripheral used to create software-defined radio systems.
upconverter	A signal conditioning device that converts a specific band of IF frequencies to high-frequency (RF) signals.
Upconversion	In a diagram that shows the components of a Software Defined Radio, the baseband modulated signal undergoes analog up conversion to frequency-translate the signal to the RF frequency at which the signal is transmitted.

V

VDC	**Voltage Direct Current**