An Introduction to Digital Communications Lab

Getting Started With LabVIEW

What you need to know to do the Lab…

LabVIEW Vocabulary

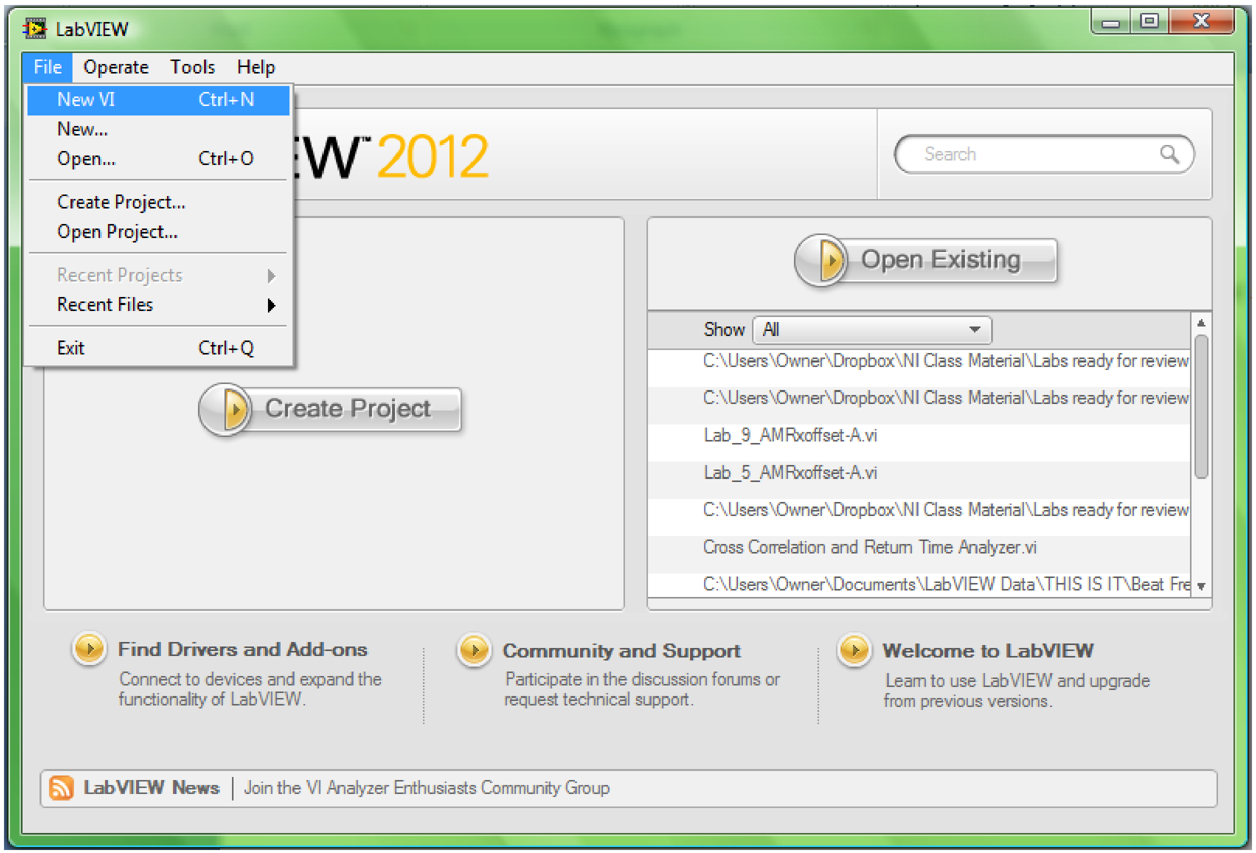
- LabVIEW is a Graphical Programming Language. The elements of the language are defined as
  - Each Application is referred to as a "Virtual Instrument" or VI.
    - Front Panel (user interface) and a block diagram.
    - Block Diagram is composed of signals (lines) and subVIs (blocks or reusable objects).
  - A subVI is a software object with inputs and outputs that and is configured using constants and controls.
    - Constant can be either a number, an array or a data structure.
    - Controls are constants and are visible on the front panel.
    - Organized into palettes so they can be selected and placed.
  - Signals are like wires and allow for the movement of data from the output of one subVI to the input of another subVI.
    - Composed of a single value, an array of values, a cluster (data structure), a waveform , or a signal.
    - Must have a source and sink point. (LabVIEW is very good at reminding you of this.)

THE UNIVERSITY of NEW MEXICO

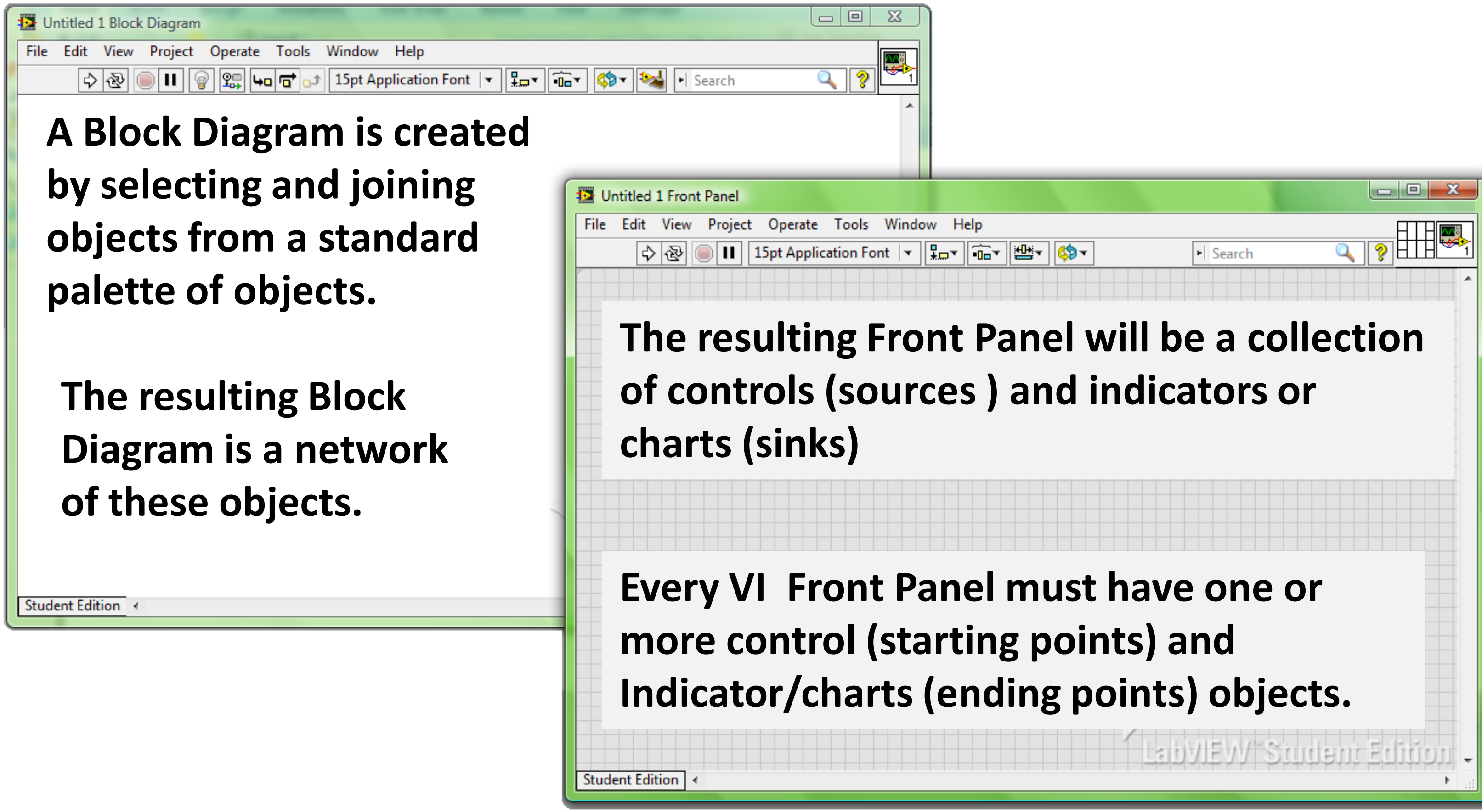
Creating A Virtual Instrument

We are now going to create a Virtual Instrument so that you can experiment and visualize how the LabVIEW works.

Select File and then New VI



New VI Screens



A Block Diagram is created by selecting and joining objects from a standard palette of objects.

The resulting Block Diagram is a network of these objects.

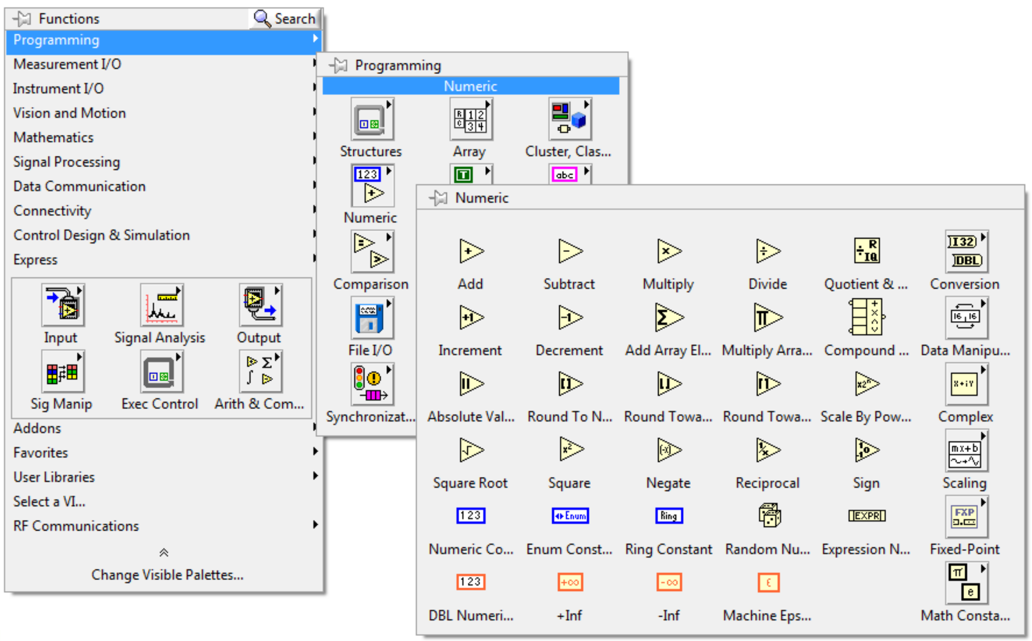
The resulting Front Panel will be a collection of controls (sources) and indicators or charts (sinks)

Every VI Front Panel must have one or more control (starting points) and Indicator/charts (ending points) objects.

Accessing Palettes

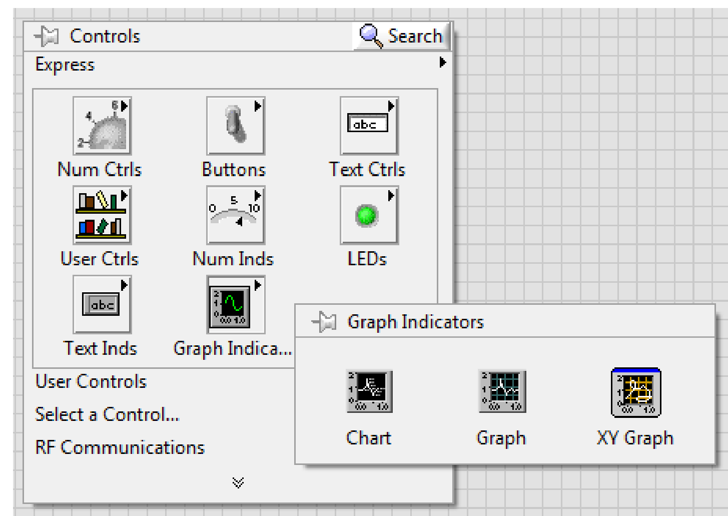
- The subVIs have been organized into a system of palettes with icons.
- A Diagram or Front Panel is build by dragging the icons from the palettes and dropping on the

Block Diagram



Front Panel

(Controls and Indicators)

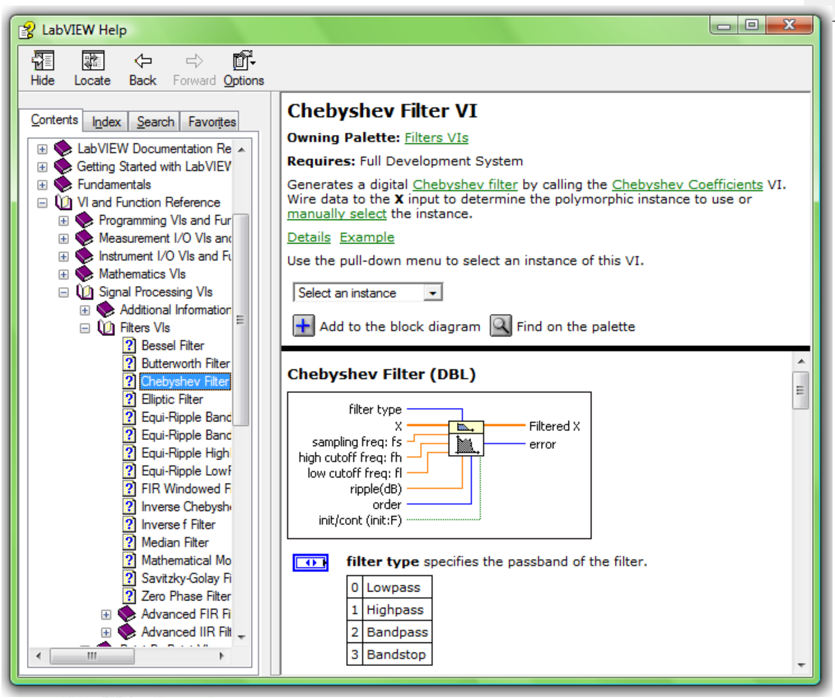


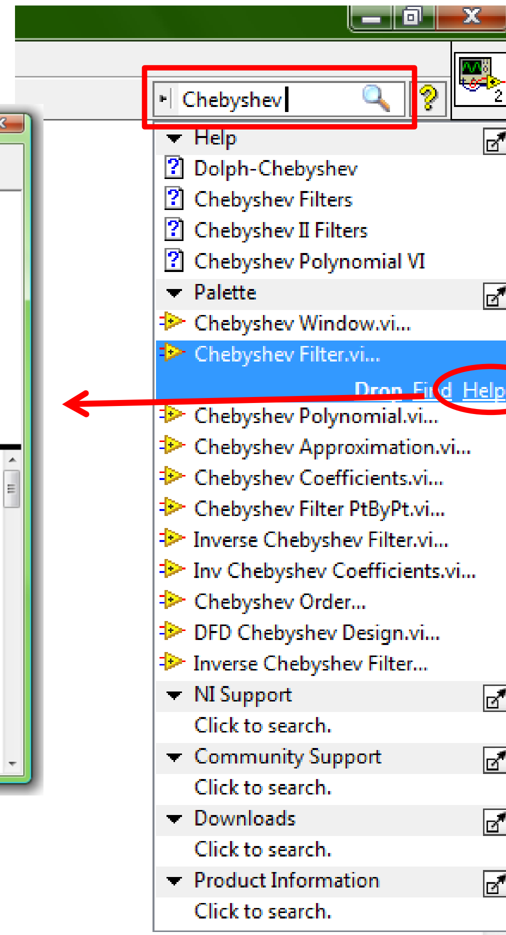
THE UNIVERSITY of NEW MEXICO

Accessing Help System (Using Search Field)

- Using the help search field in the toolbar.

3. Help Screen for topic.



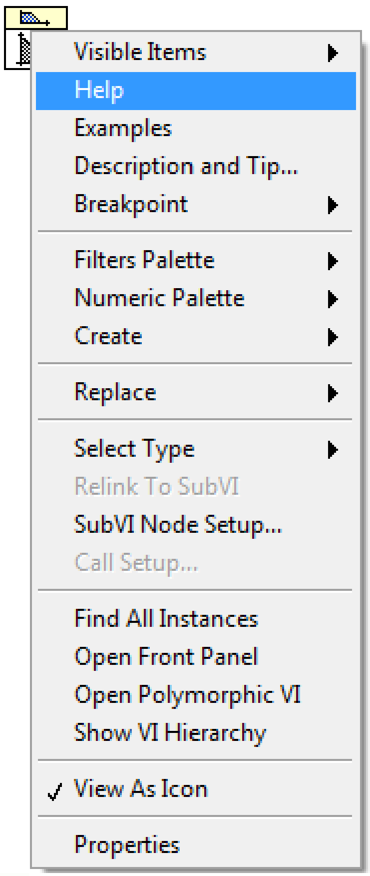


1. Enter what you want to find in the field. (Chebyshev Filter)

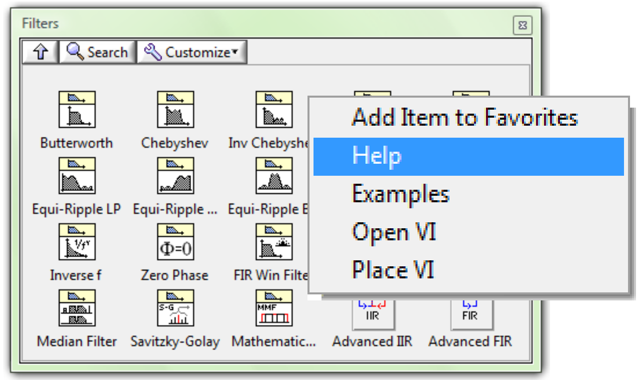
2. Select help on topic.

Other Ways to Access Help

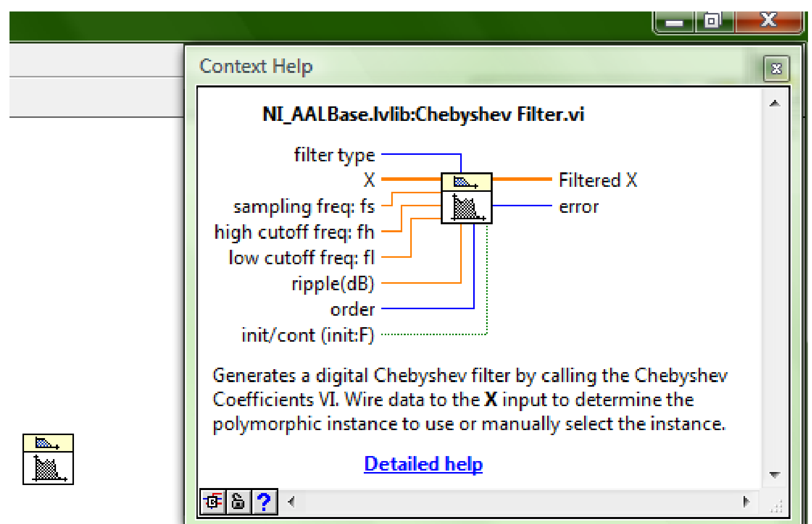
1. Right click on Icon in diagram



2. Right click on Icon in Palette



3. Placing cursor on icon and typing <Ctrl> H



THE UNIVERSITY *of* NEW MEXICO

Anatomy of A Help Screen

Chebyshev Filter VI

Owning Palette: Filters VIs

Requires: Full Development System

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

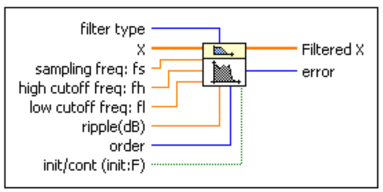
Details Example

Use the pull-down menu to select an instance of this VI.

Select an instance

Add to the block diagram Find on the palette

Chebyshev Filter (DBL)



filter type specifies the passband of the filter.

0	Lowpass
1	Highpass
2	Bandpass
3	Bandstop

X is the input signal to filter.

sampling freq: fs is the frequency in Hz at which you want to sample **X** and must be greater than 0. The default is 1.0 Hz. If **sampling freq: fs** is less than or equal to 0, this VI sets **Filtered X** to an empty array and returns an error.

high cutoff freq: fh is the high cutoff frequency in Hz. The default is 0.45 Hz. The VI ignores this parameter when **filter type** is 0 (Lowpass) or 1 (Highpass). When **filter type** is 2 (Bandpass) or 3 (Bandstop), **high cutoff freq: fh** must be greater than **low cutoff freq: fl** and observe the Nyquist criterion.

low cutoff freq: fl is the low cutoff frequency in Hz and must observe the Nyquist criterion. The default is 0.125 Hz. If **low cutoff freq: fl** is less than or equal to 0 or greater than half the value of **sampling freq: fs**, the VI sets **Filtered X** to an empty array and returns an error. When **filter type** is 2 (Bandpass) or 3 (Bandstop), **low cutoff freq: fl** must be less than **high cutoff freq: fh**.

ripple is the ripple in the passband. **ripple** must be greater than zero and expressed in decibels. The default is 0.1. If **ripple** is less than or equal to zero, the VI sets **Filtered X** to an empty array and returns an error.

order specifies the filter order and must be greater than 0. The default is 2. If **order** is less than or equal to 0, the VI sets **Filtered X** to an empty array and returns an error.

init/cont controls the initialization of the internal states. The default is FALSE. The first time this VI runs or if **init/cont** is FALSE, LabVIEW initializes the internal states to 0. If **init/cont** is TRUE, LabVIEW initializes the internal states to the final states from the previous call to this instance of this VI. To process a large data sequence that consists of smaller blocks, set this input to FALSE for the first block and to TRUE for continuous filtering of all remaining blocks.

Filtered X is the output array of filtered samples.

error returns any error or warning from the VI. You can wire **error** to the Error Cluster From Error Code VI to convert the error code or warning into an error cluster.

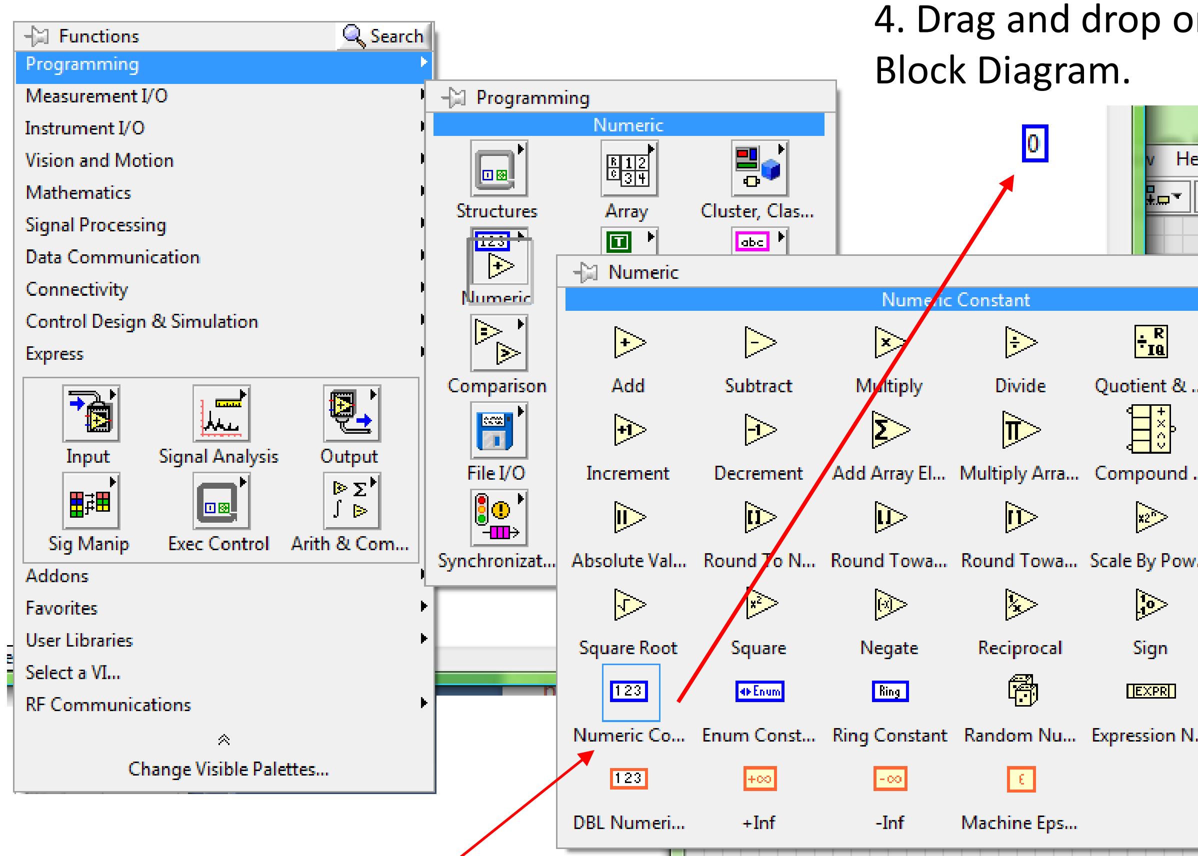
Location on palettes
System Requirements
Descriptions
Help Navigation Description
Palette Navigation Description
Connector Identifications
Connector Descriptions

GINET
Global Innovation Network for Entrepreneurship and Technology

Picking Source Objects (Block Diagram)

In this example we will place a constant in our diagram. The first step is to Right Click in an open area of the block diagram to launch the palette browser.

2. Select the palette with the constant object in it. This is done by navigating through the menu system as shown here. (Note constants are found on the numeric palette.)



4. Drag and drop onto Block Diagram.

3. Select subVI you wish place

THE UNIVERSITY of NEW MEXICO

Picking Source Objects (Front Panel)

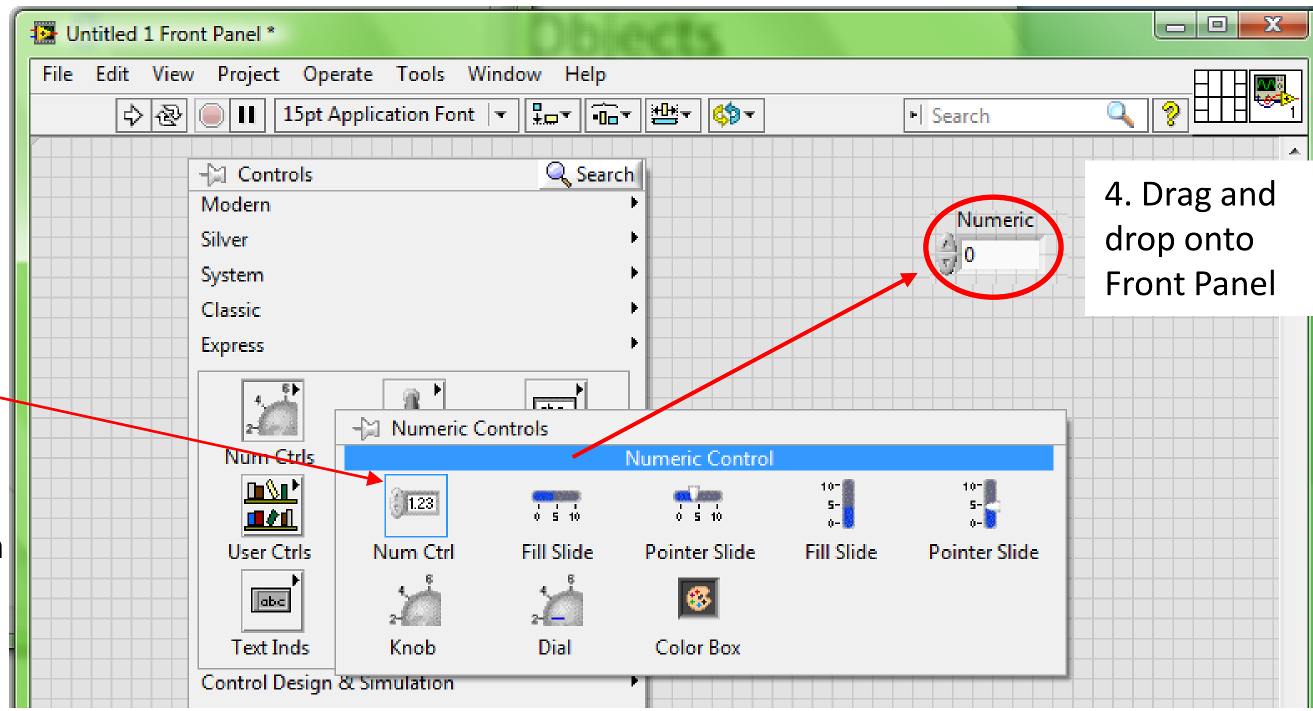
In this example we will place a constant in our Front Panel. The first step is to Right Click in an open area of the block diagram to launch the palette browser.

2. Select the palette with the constant object in it. This is done by navigating through the menu system as shown here. (Note constants are found on the numeric controls.)

3. Select subVI you wish place

Note: The selection of a control will also result in a block

being added to the block diagram.



4. Drag and drop onto Front Panel

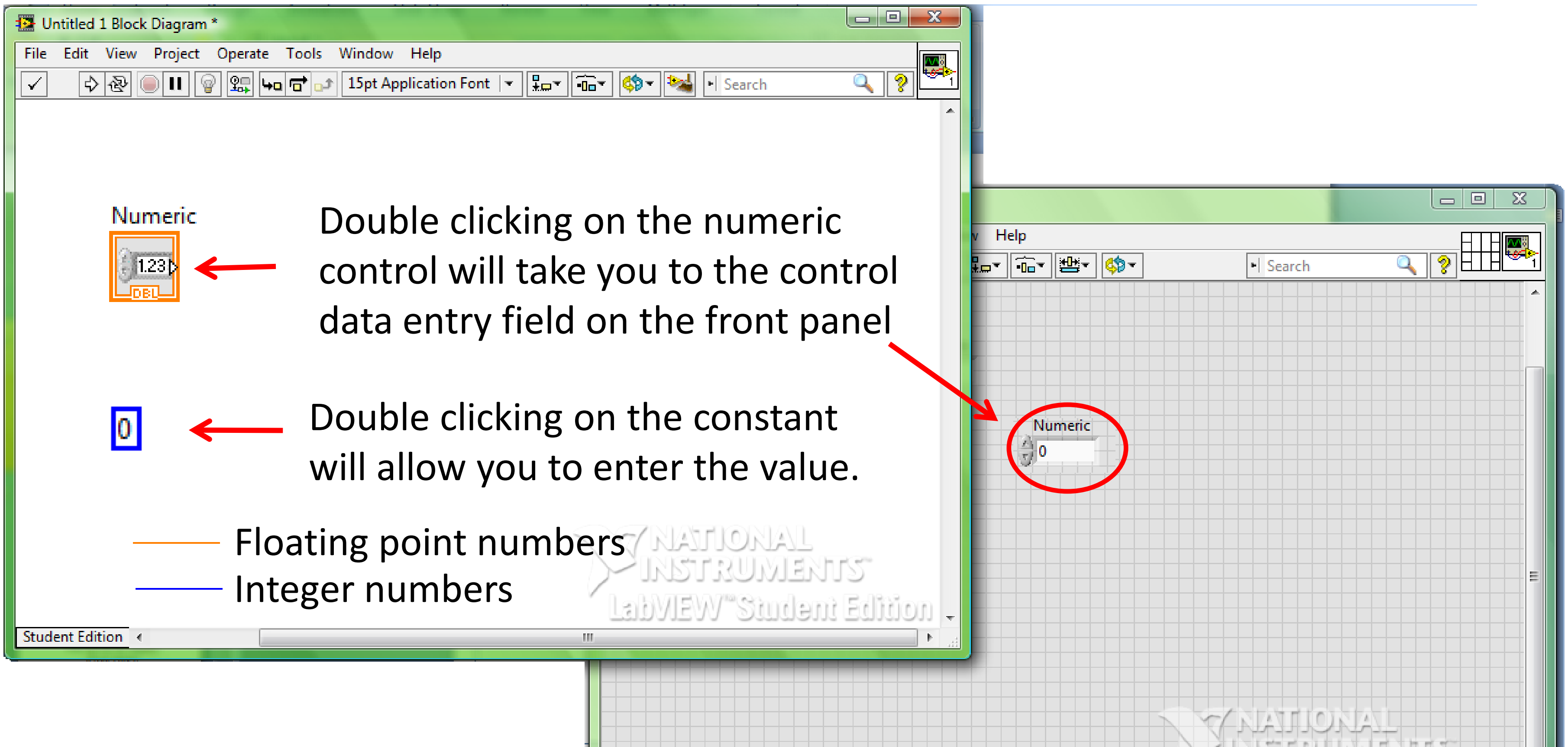
istec Ibero-American Science & Technology Education Consortium

Printed on 9/1/2014

GINET Global Innovation Network for Entrepreneurship and Technology

THE UNIVERSITY of NEW MEXICO

Setting Values for Constants and Controls



Double clicking on the numeric control will take you to the control data entry field on the front panel

Double clicking on the constant will allow you to enter the value.

Floating point numbers

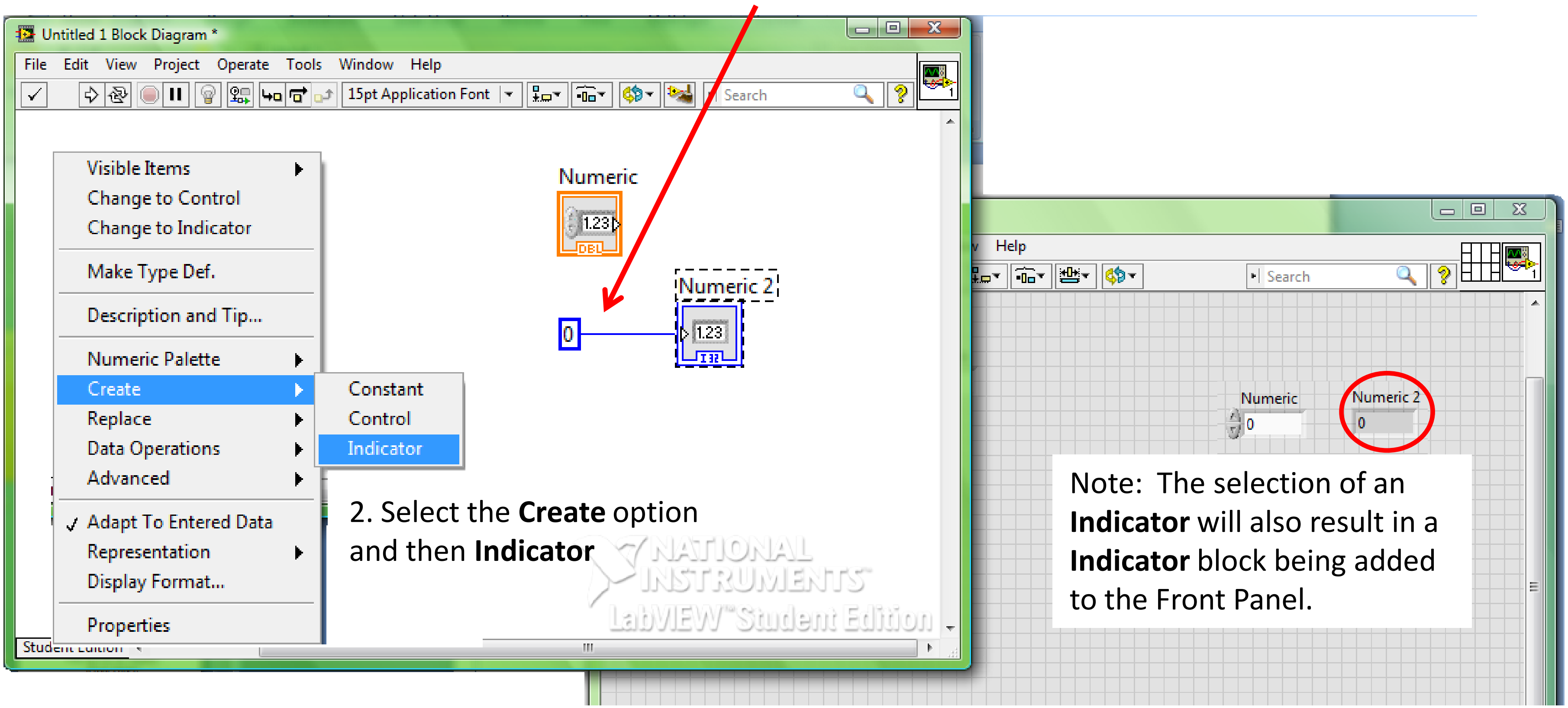
Integer numbers

istec Ibero-American Science & Technology Education Consortium

GINET Global Innovation Network for Entrepreneurship and Technology

Picking Sink Nodes

Right click on the connection point for the constant and the properties menu should appear.



Ibero-American Science & Technology Education Consortium

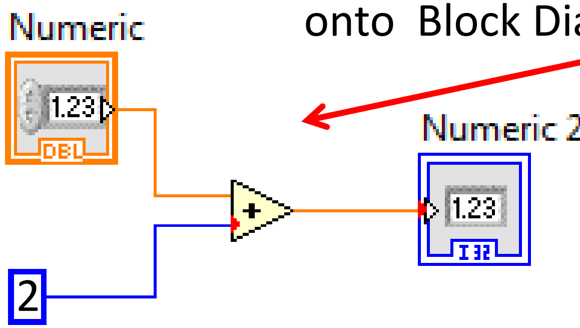
Global Innovation Network for Entrepreneurship and Technology

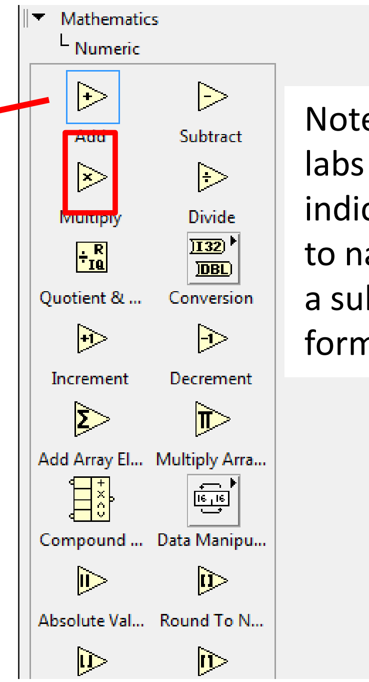
Adding an SubVI to the Diagram

In this example we will place an addition SubVI in our diagram. The first step is to Right Click in an open area of the block diagram to launch the palette browser.

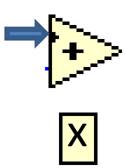
Finding connection points on subVIs. Placing the mouse cursor on the edge will cause the connection's label w appear on the drawing as shown below

Drag and drop **Add** subVI onto Block Diagram.



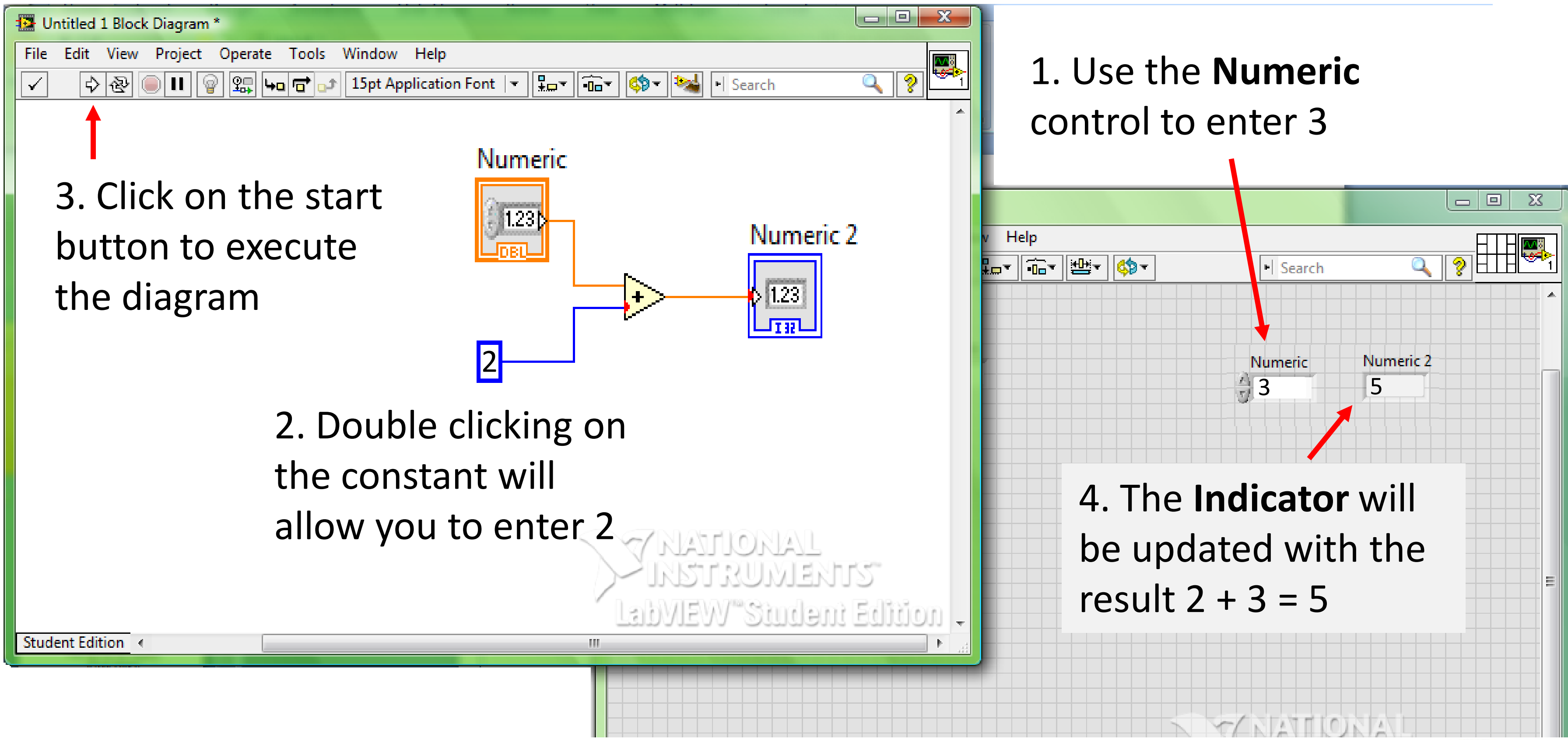


Note: In our labs we will indicate how to navigate to a subVI in this format



1) Move mouse to **Constant** block until connection appears
2) Click and hold left mouse button and drag over to **Add** subVI. A dashed line will mark the proposed path of the wire.
3) Release mouse button when conection point on edge of Add subVI appears. Dashed line will turn solid.

Causing Diagram to Execute



1. Use the **Numeric** control to enter 3

2. Double clicking on the constant will allow you to enter 2

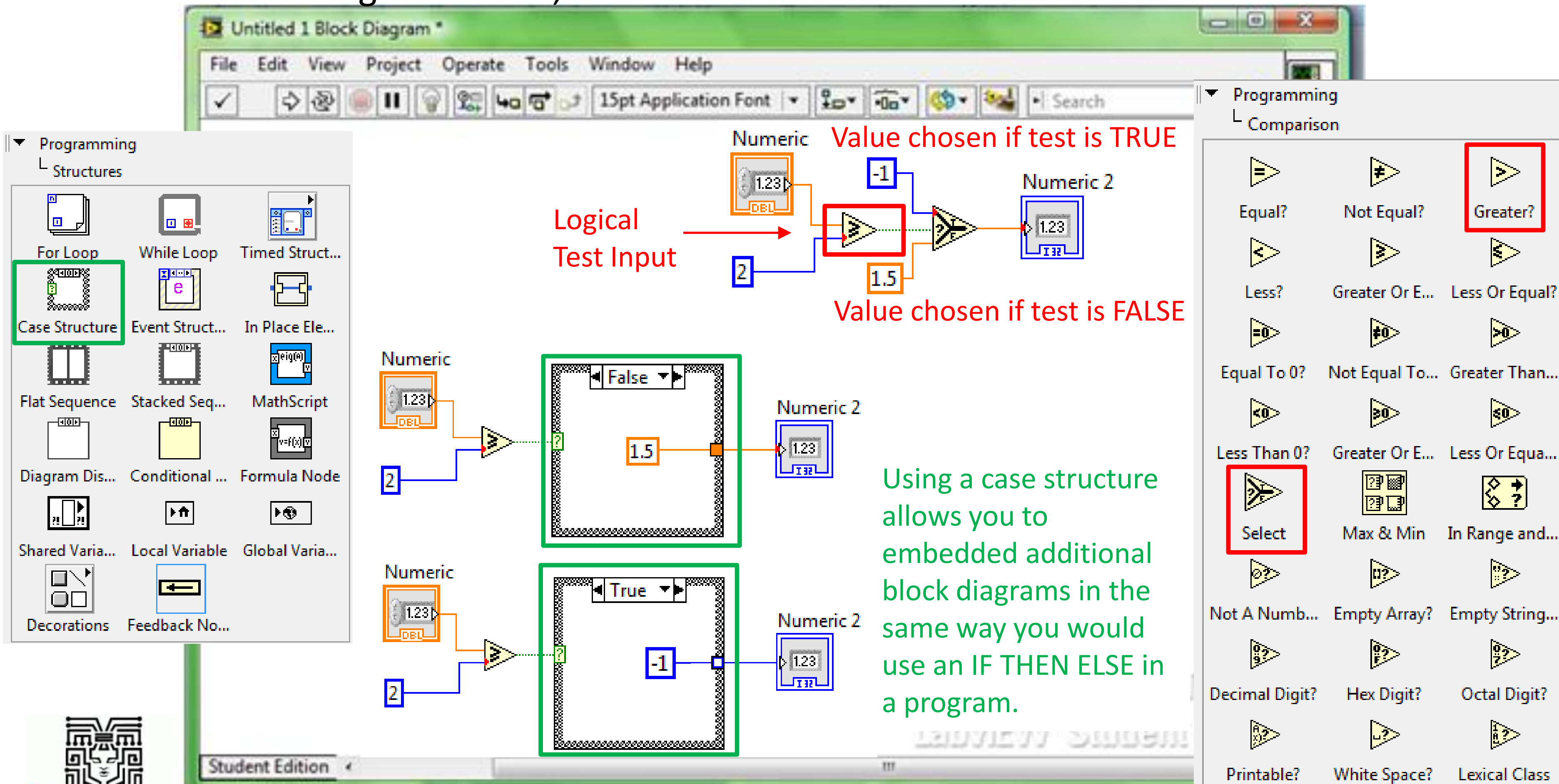
3. Click on the start button to execute the diagram

4. The **Indicator** will be updated with the result 2 + 3 = 5

Printed on 9/1/2014

Logic Structures (IF THEN ELSE)

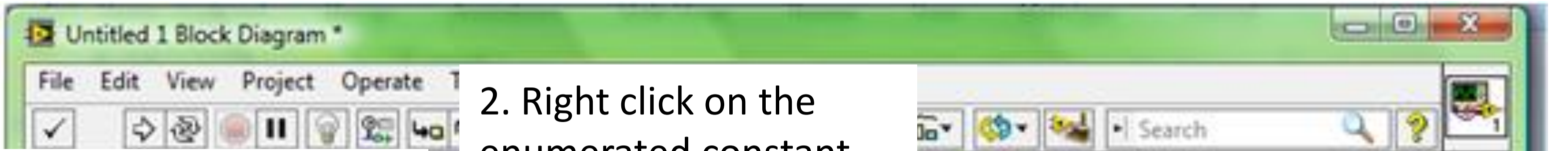
In the LabVIEW paradigm, signals are routed based on a logical test. For example, lets examine the following statement, IF Numeric ≥ 2 THEN Numeric 2 is -1 ELSE Numeric 2 is 1.5.



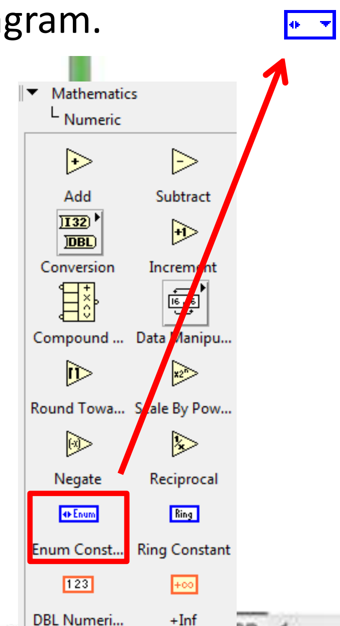
Using a case structure allows you to embedded additional block diagrams in the same way you would use an IF THEN ELSE in a program.

Enumerated Data (Block Diagram)

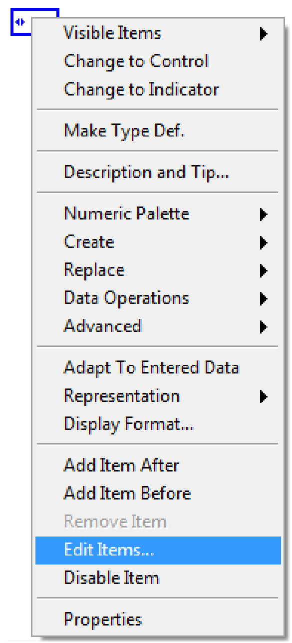
THE UNIVERSITY of NEW MEXICO



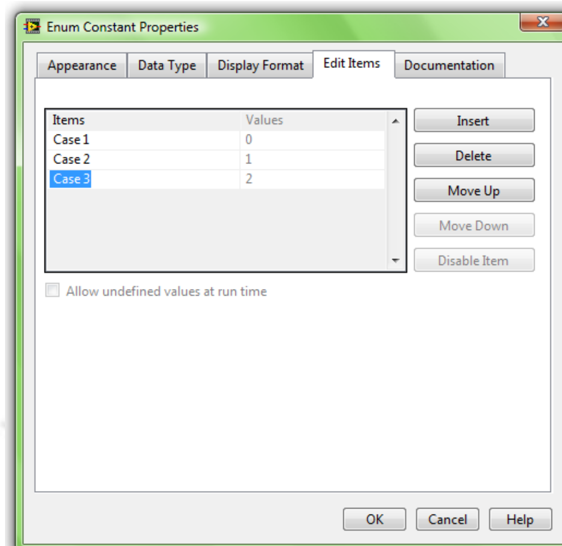
1. Select an enumerated constant from the Mathematics palette and drag and drop onto the block diagram.



2. Right click on the enumerated constant and selected "edit items ..." menu items.



3. Enter labels for each number value



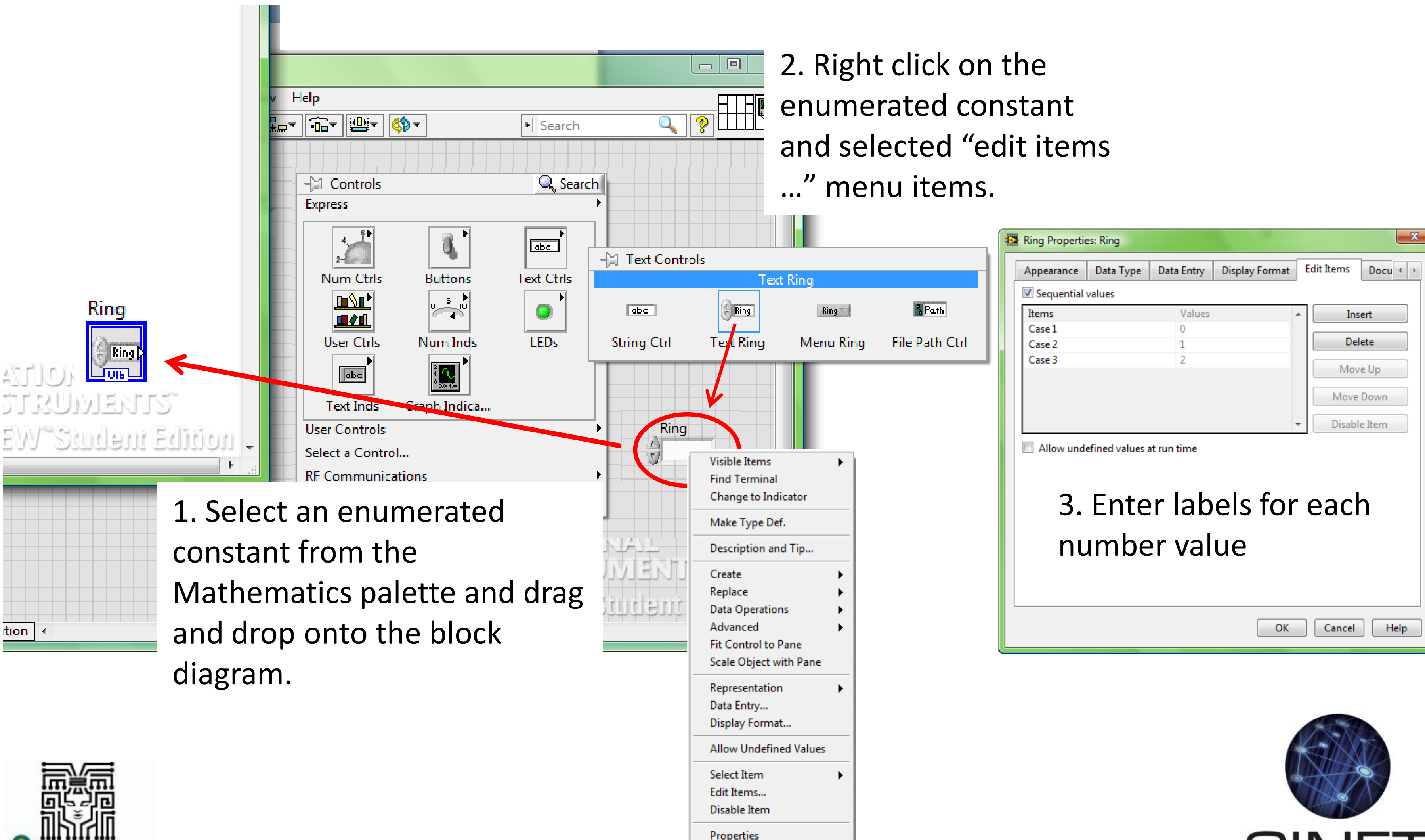
istec Ibero-American Science & Technology Education Consortium

Printed on 9/1/2014

GINET Global Innovation Network for Entrepreneurship and Technology

THE UNIVERSITY of NEW MEXICO

Enumerated Data (Front Panel – Text Ring)



2. Right click on the enumerated constant and selected "edit items …" menu items.

1. Select an enumerated constant from the Mathematics palette and drag and drop onto the block diagram.

3. Enter labels for each number value

istec Ibero-American Science & Technology Education Consortium

Printed on 9/1/2014

GINET Global Innovation Network for Entrepreneurship and Technology

Case Statements

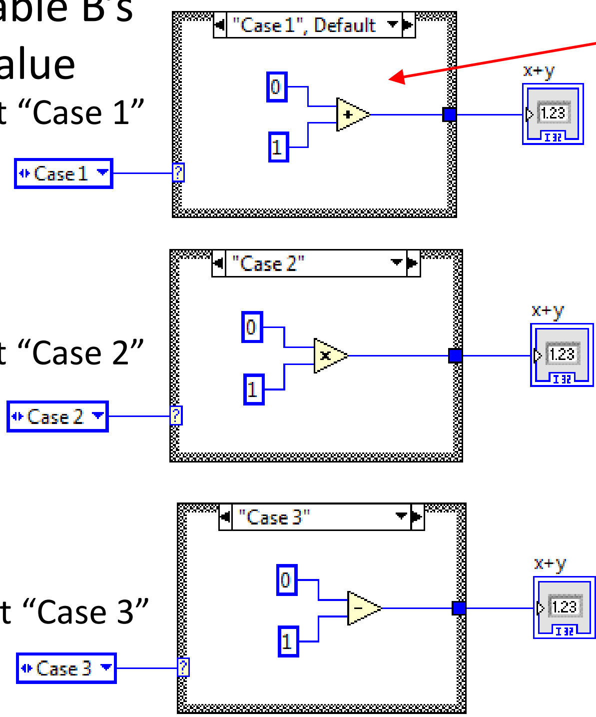
In the LabVIEW paradigm, signals are routed based on a logical test.

Variable B's Value

The input "Case 1"

The input "Case 2"

The input "Case 3"



The math function was selected from the Mathematics Library

Implements the following

switch (B)
case 1:
C = 0 + 1;
case 2:
C = 0 * 1;
case 3:
C = 0 – 1;
end

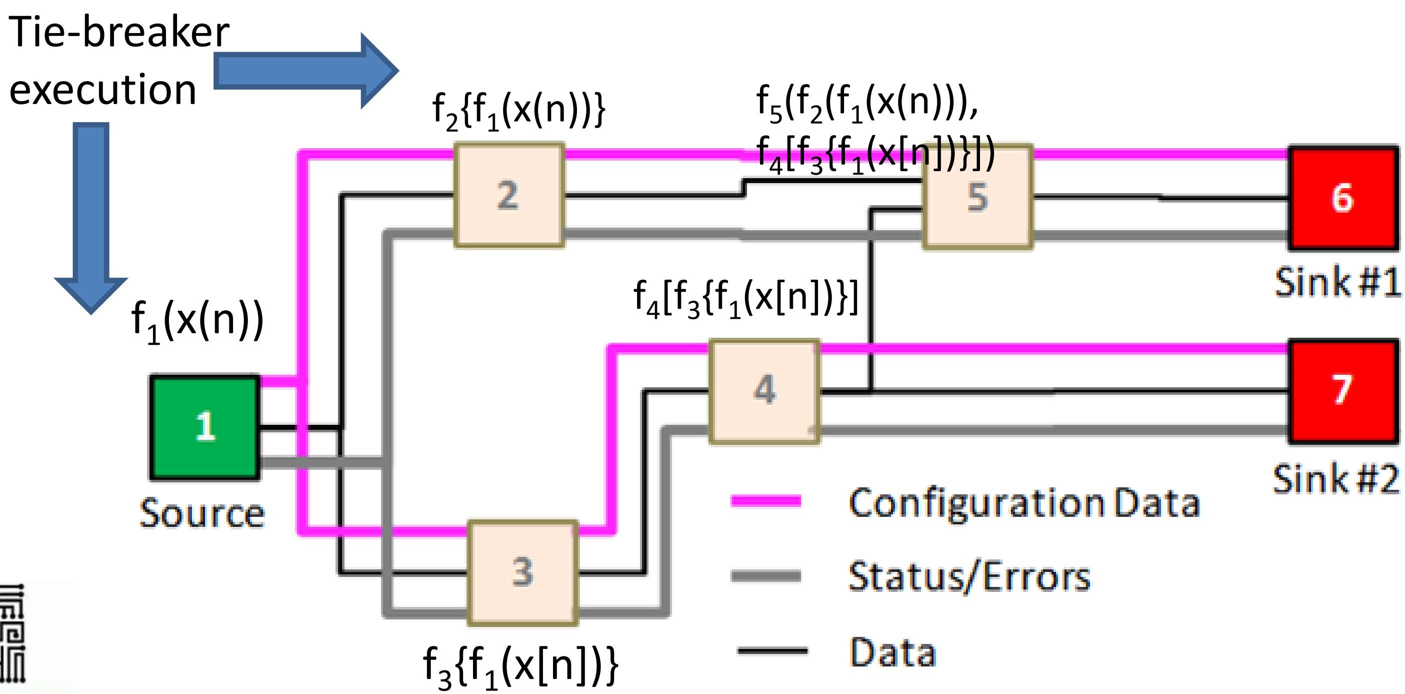
Iteration

- Generally software design uses iteration for
  - Moving data from one structure to another.
  - Repeating a set of instructions until some condition is TRUE.
  - Creating counts or accumulating data

- Moving Data
  - LabVIEW supports all these behaviors but in a different way than you are used to.
  - LabVIEW assumes that the native data structure is an n-dimensional array.
  - Diagram execution automatically transfers data from one subVI to another without the user having to do this explicitly.

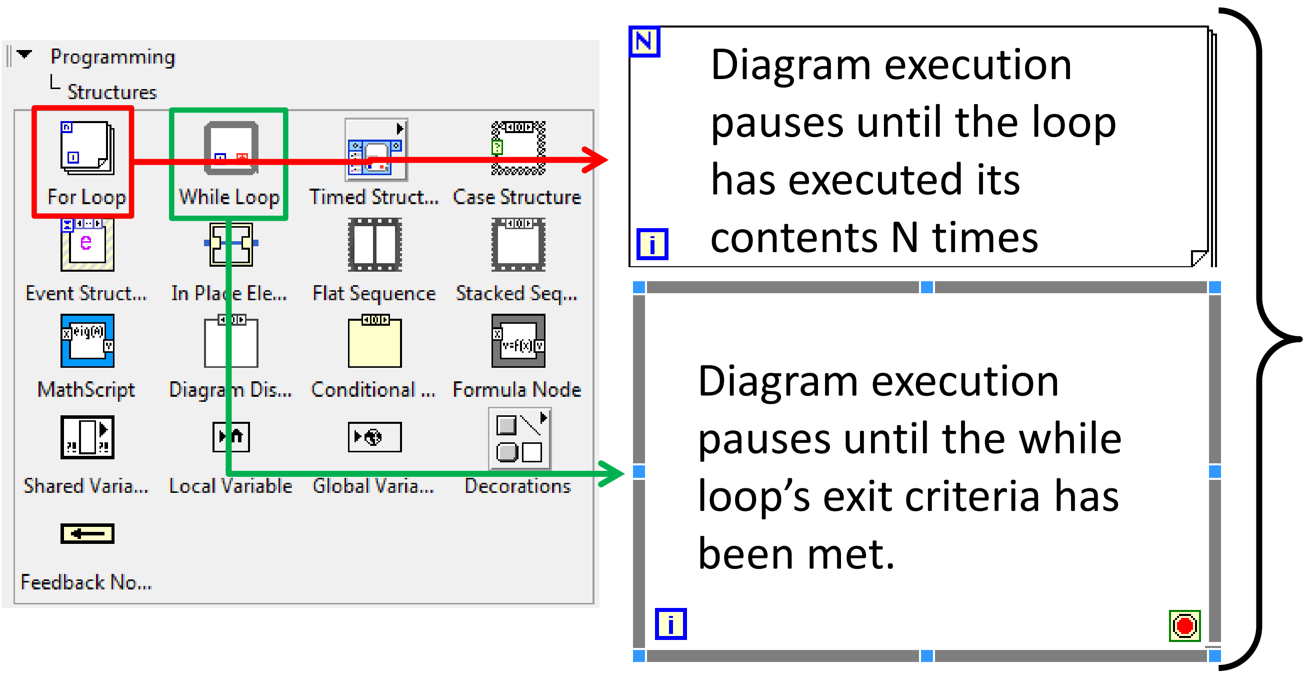
Diagram Execution Details

- LabVIEW the diagram is the set of instructions. LabVIEW executes at a default interval determined by the fastest rate needed for the subVIs to execute properly. (Without a looping structure the diagram executes only once.)
- You need to use a loop to get the diagram to execute repeatedly until the data collection task is complete.



Looping Structures

In this example we will place a for loop and a while loop in our diagram by dragging these from the Structures palette and dropping in the diagram.

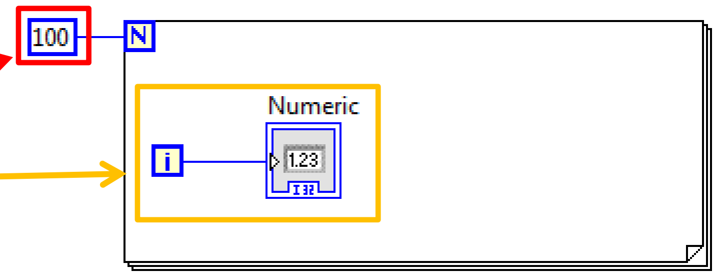


Only the indicators inside the loops will update

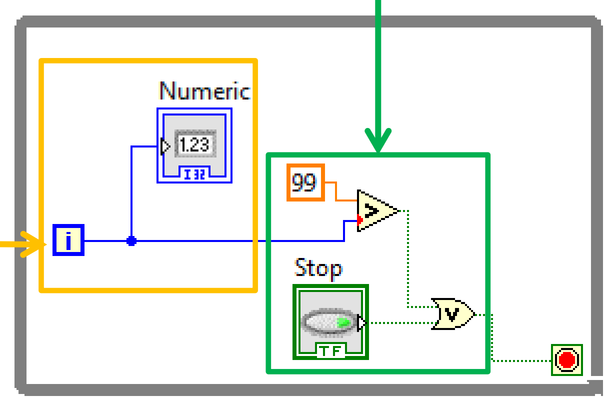
Details of Setting-up A Looping Structure

Implements the following

```
for i = 1 , 100 {
    print(i);
}
```




```
stop = 0;
i = 0;
while (i < 99) & stop == 0 {
    print(i);
}
```



Note: Stop control is defaulted to FALSE

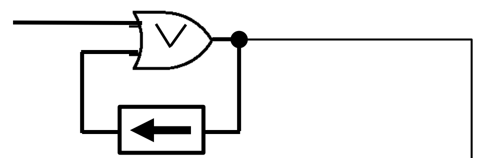
Data Latching & Counters (Logic Overview)

Counter start pulse is sent at t_2.

Digital representation of

```
start = 0;
if startPulse == 1{
    start = 1;
}
```

t_0	t_1	t_2	t_3
0	0	1	0



Delays signal by one sample interval.

t_0	t_1	t_2	t_3
0	0	0	1

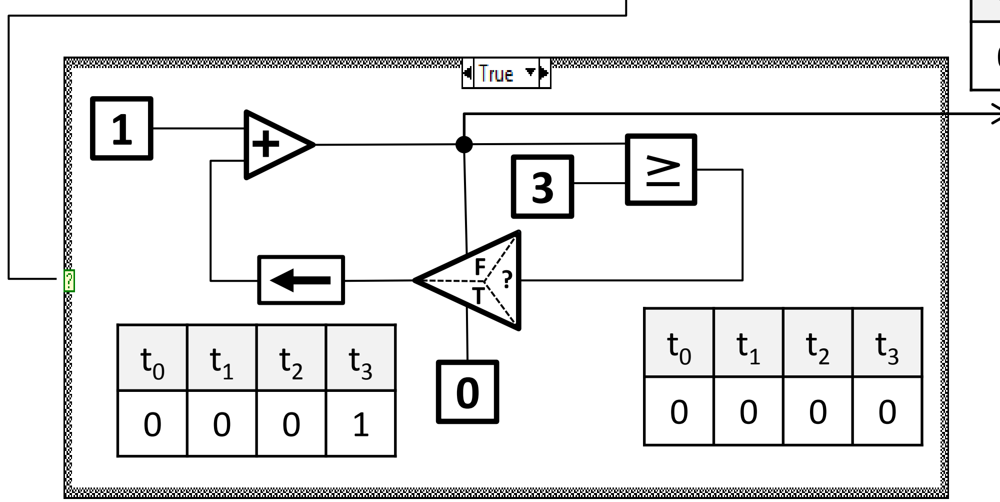
The Signal latches at t_2 seconds and remains so till the VI is stopped.

t_0	t_1	t_2	t_3
0	0	1	1

Digital representation of

```
if start == 1{
    sum[0] = 0;
    if sum[n] >=3 {
       sum[n] = 1;}
    else{
       sum[n] =  sum[n-1] + 1;}
  }
}
```

where, n is the current sample



t_0	t_1	t_2	t_3
0	0	0	1

t_0	t_1	t_2	t_3
0	0	0	0

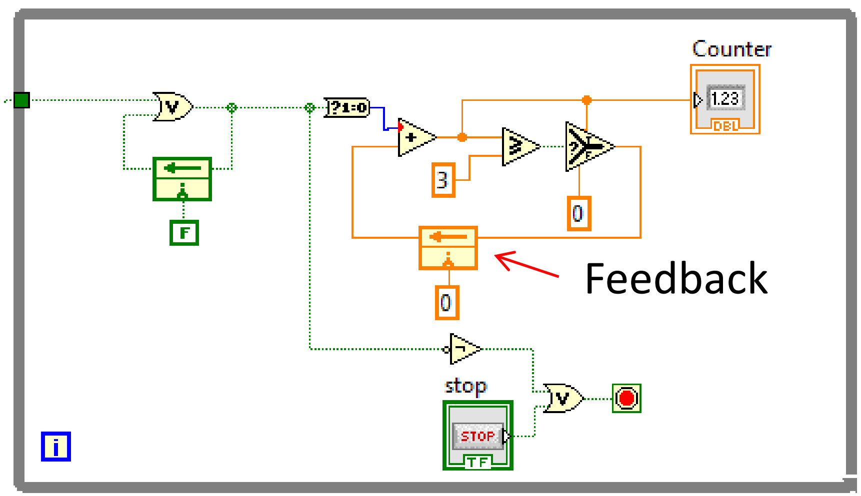
t_0	t_1	t_2	t_3
0	0	1	2

Note: “False” case outputs a “0”

Data Latching & Counters

Counter start pulse is sent at t_2.

t_0	t_1	t_2	t_3
0	0	1	0

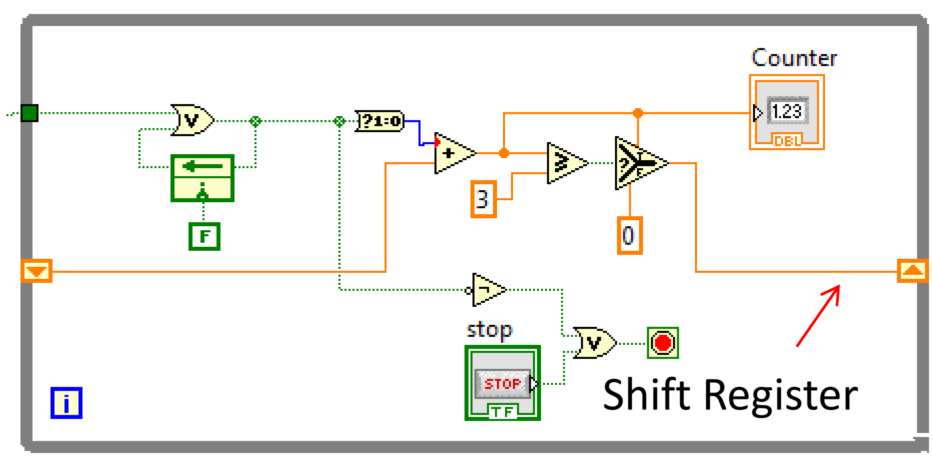


Counter outputs sequence 1,2,3 repeatedly.

t_0	t_1	t_2	t_3
0	0	1	2

Counter start pulse is sent at t_2.

t_0	t_1	t_2	t_3
0	0	1	0



Counter outputs sequence 1,2,3 repeatedly.

t_0	t_1	t_2	t_3
0	0	1	2

THE UNIVERSITY of NEW MEXICO

Data Routing

- Signals or data flows along the lines connecting the subVIs.
- It is strongly recommended that you think of the lines not as wires but as data flows. The following legend will help identify the data flowing along the line.

Floating point numbers
Array of Floating point numbers
Integer numbers (signed or unsigned)
Array of Integer numbers (signed or unsigned)
Boolean or Logical values
Array of Boolean or Logical values

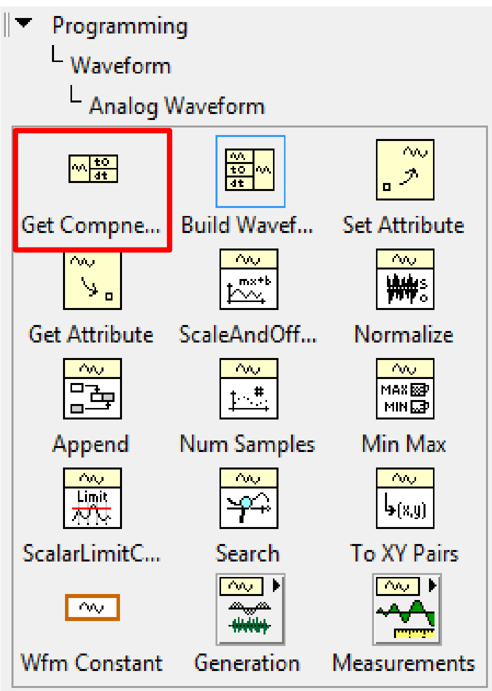
Waveform Cluster
Signal Cluster
USRP Status\Error
USRP Configuration Data

istec Ibero-American Science & Technology Education Consortium

GINET Global Innovation Network for Entrepreneurship and Technology

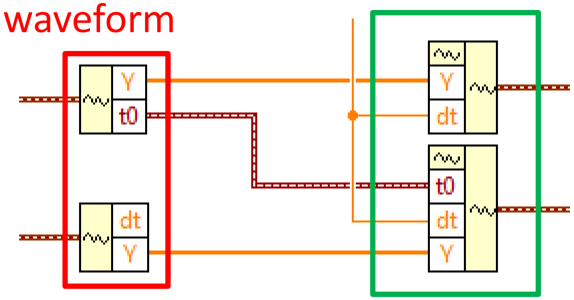
Accessing Data in Waveforms

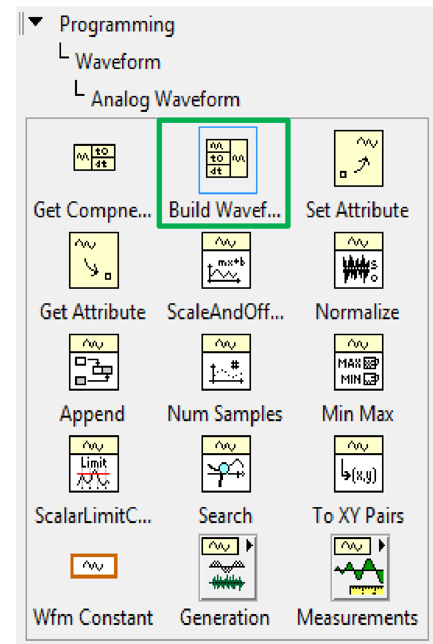
At times it may be necessary to access the data in a data flow. The data is always designated as Y.



Get Component subVIs allow you to access the elements that have been clustered to form the waveform

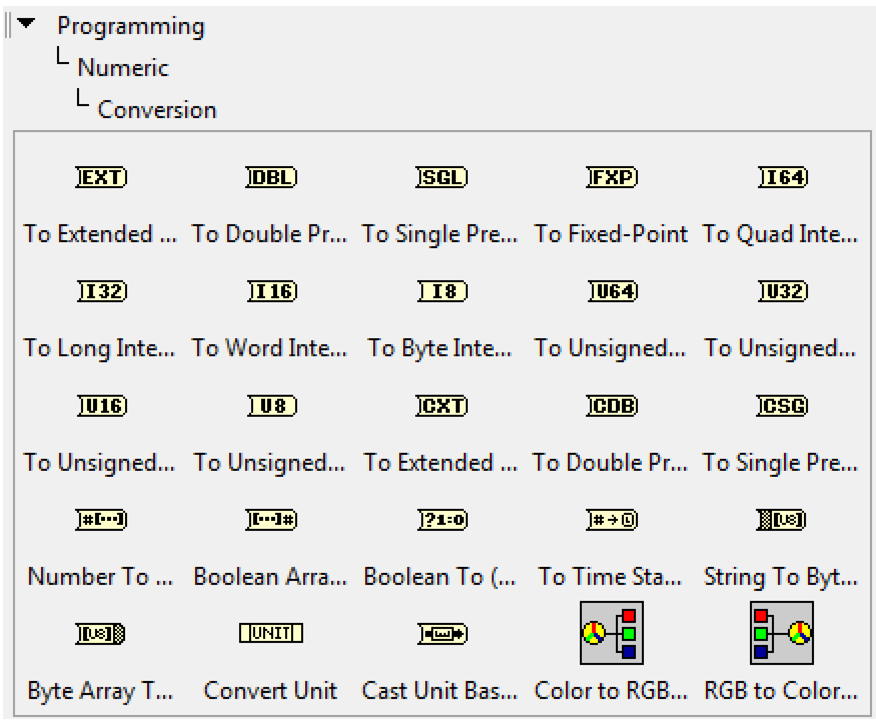
Build Waveform blocks allow you to cluster elements together to form the waveform

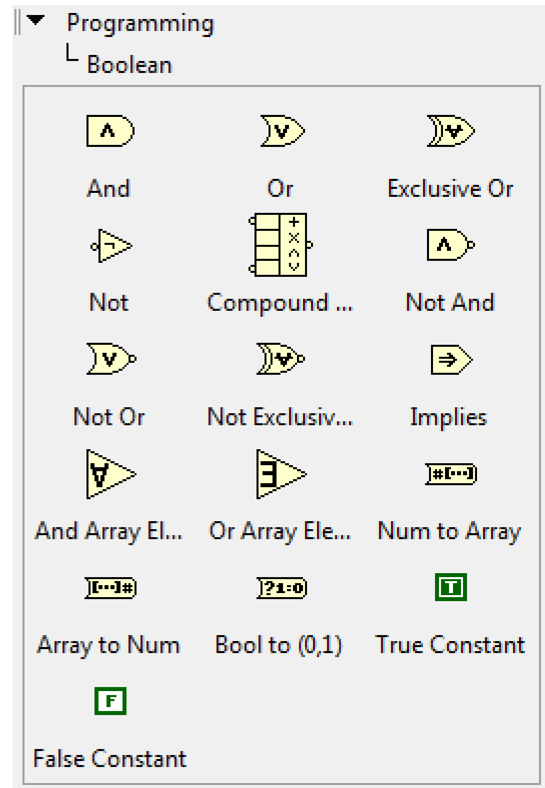




Converting Between Data Types

- Conversions between data types can be found on the Conversion palette and the Boolean Palette.

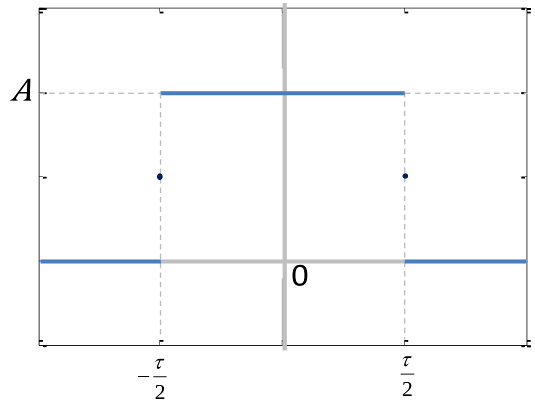


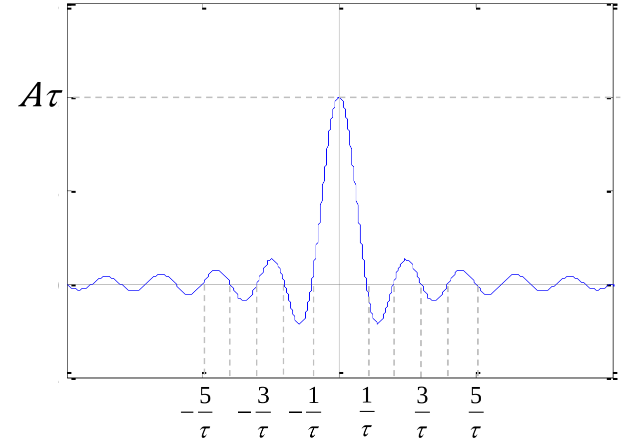


Frequency-domain Characterization of Signals: A Look at the Fourier Transform

What you need to know to do the Lab…

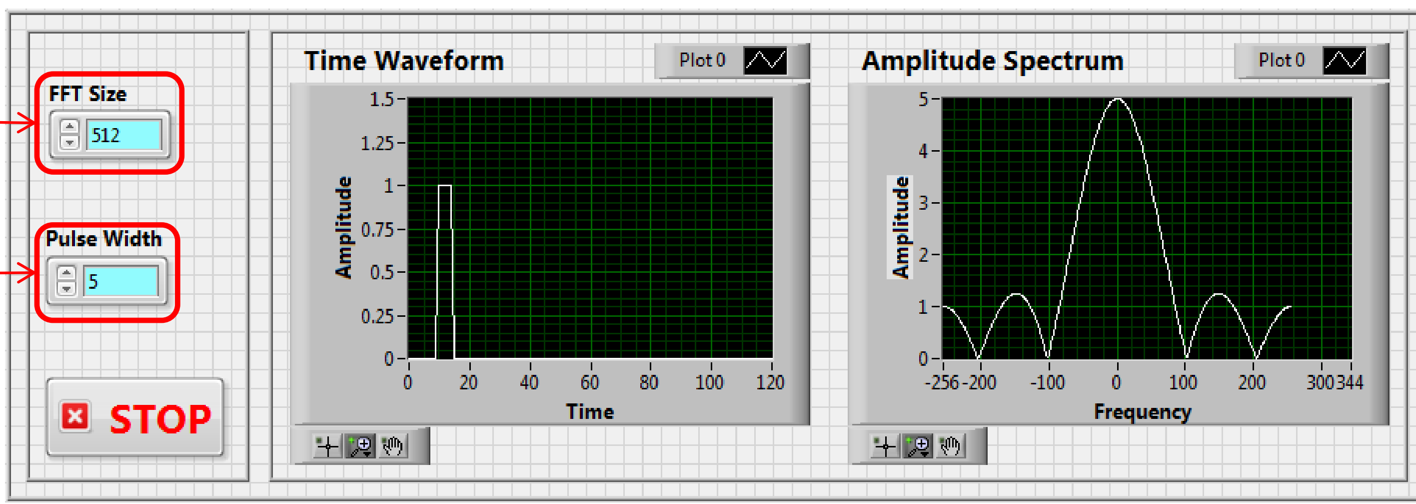
Fourier Transforms Using FFT





2. Number of samples

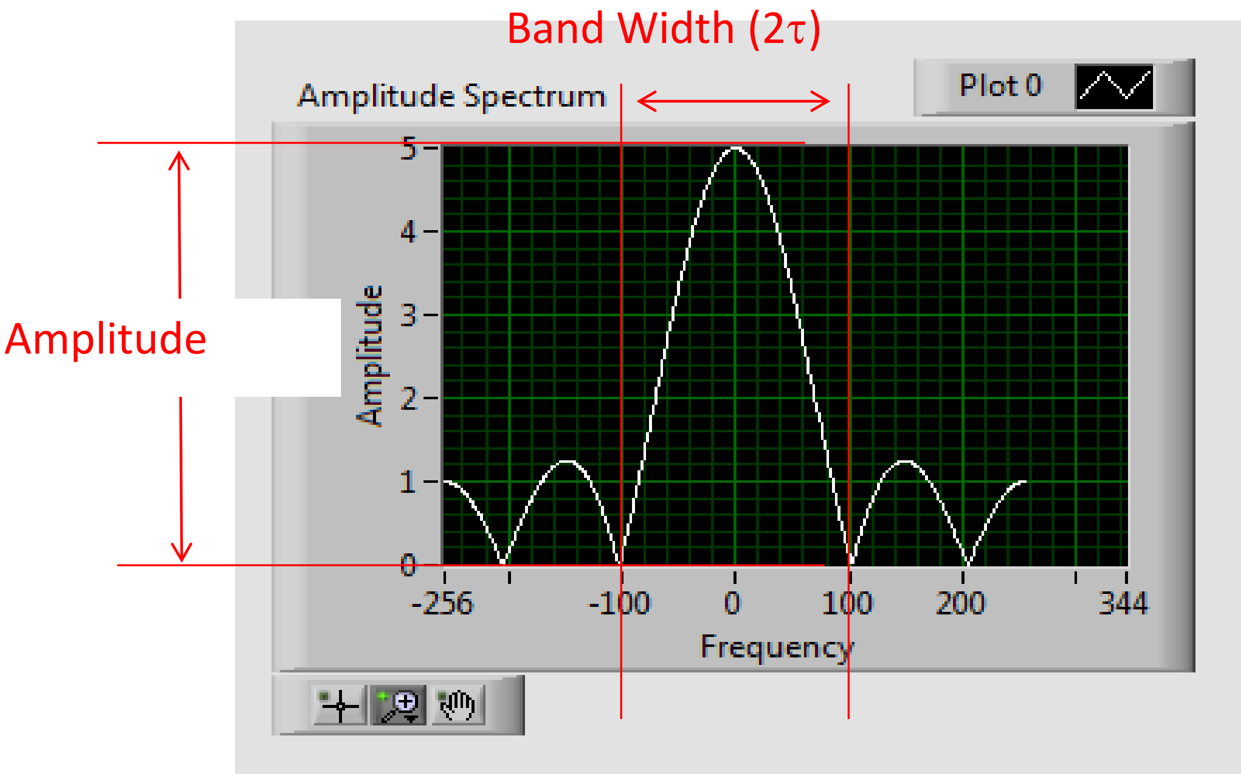
3. Pulse width (τ)



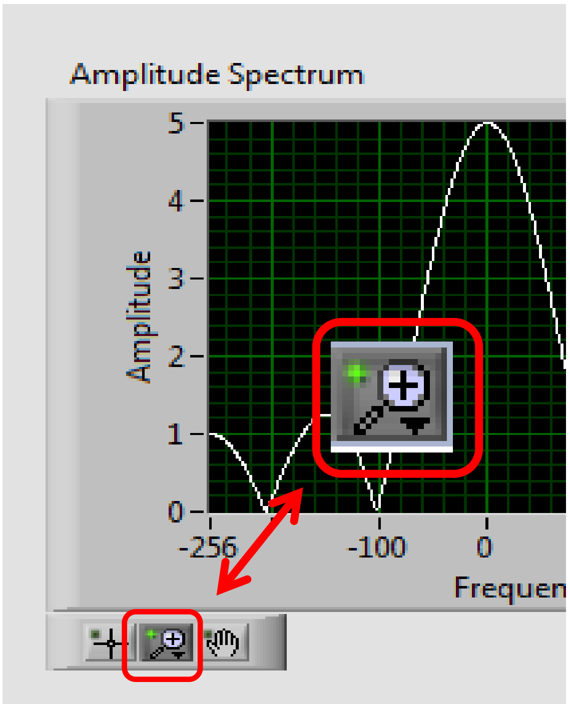
$$\Pi(t) = \begin{cases} 0, |t| > \tau/2 \\ \frac{A}{2}, |t| = \tau/2 \\ A, \ |t| < \tau/2 \end{cases}$$

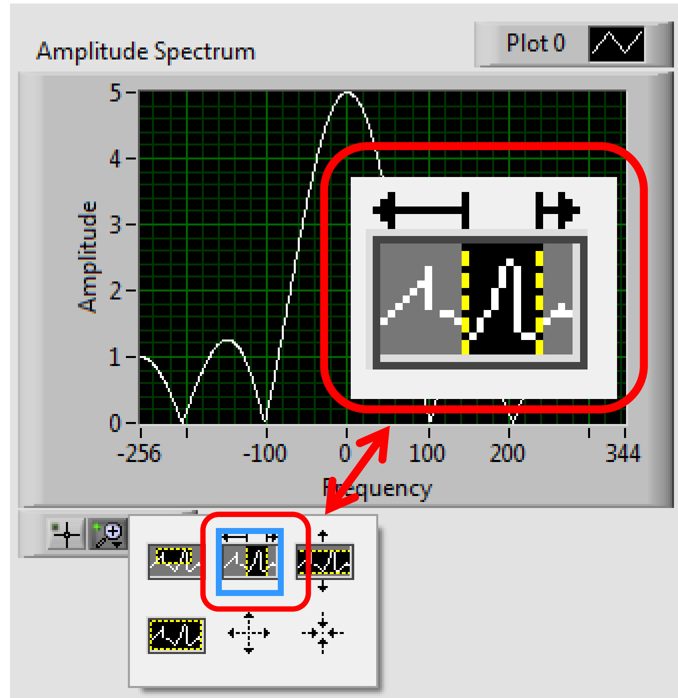
$$A\,\tau\,\text{sinc}\,(\pi f \tau) = \frac{A\sin(\pi f \tau)}{\pi f \tau}$$

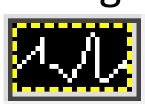
Measuring Bandwidth



Making Measurements Using Zoom Feature

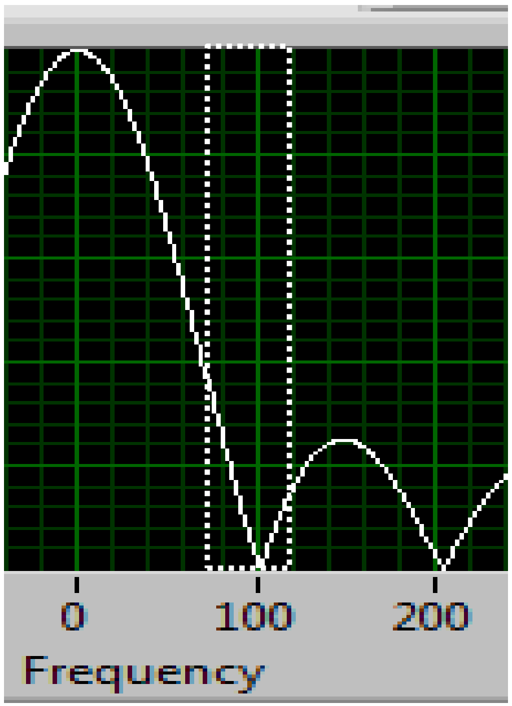


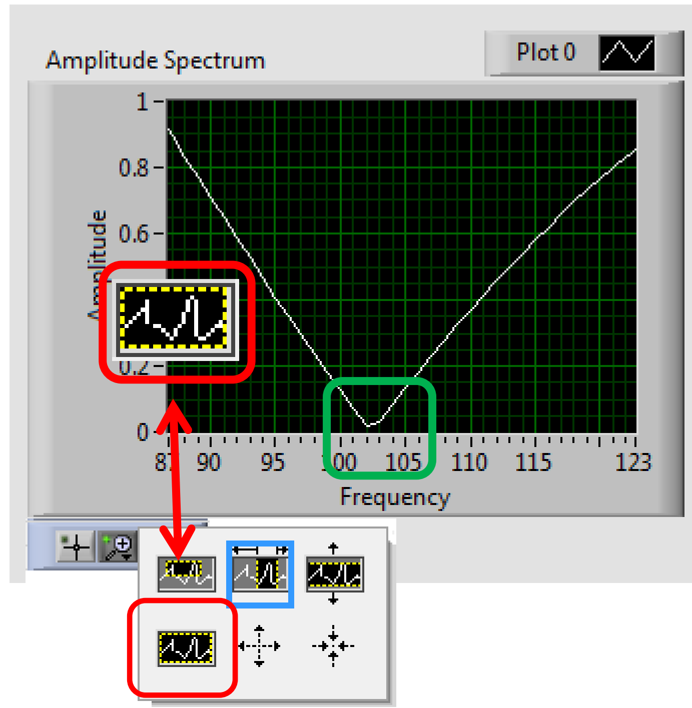


1. Select Magnification Button 

2. Select Horizontal Magnification

Making Measurements Using Zoom Feature (Concluded)

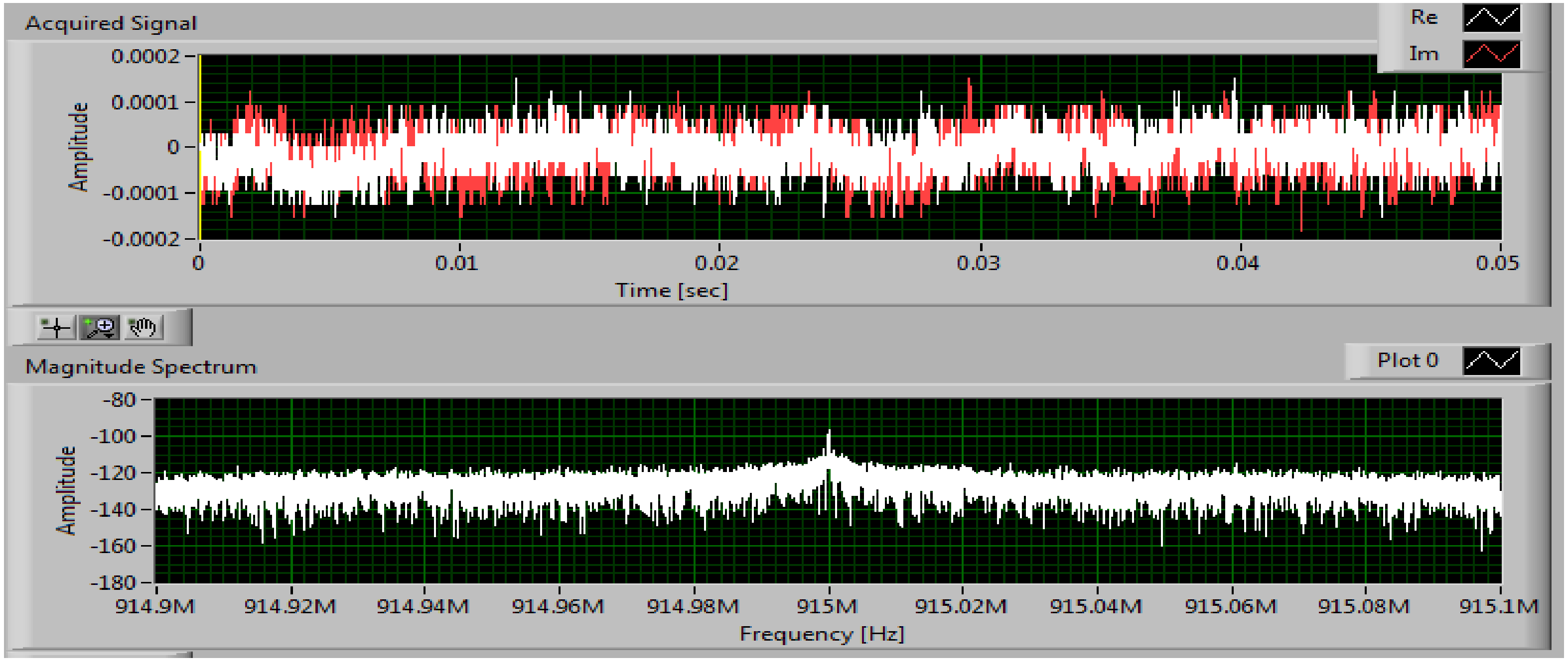




3. Select Horizontal Range to be magnified using tool's cursor

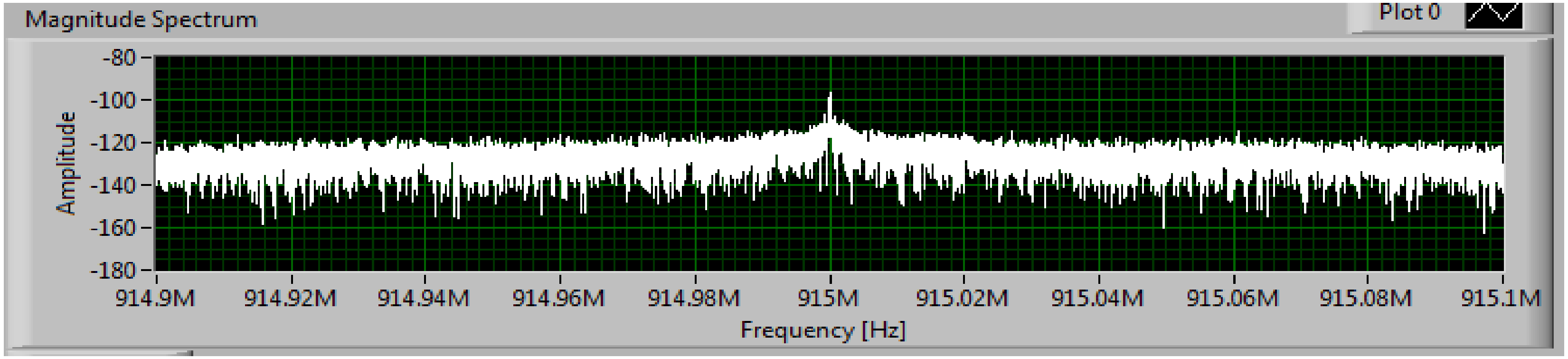
4. Observe data and return to non-magnified mode for next observation.

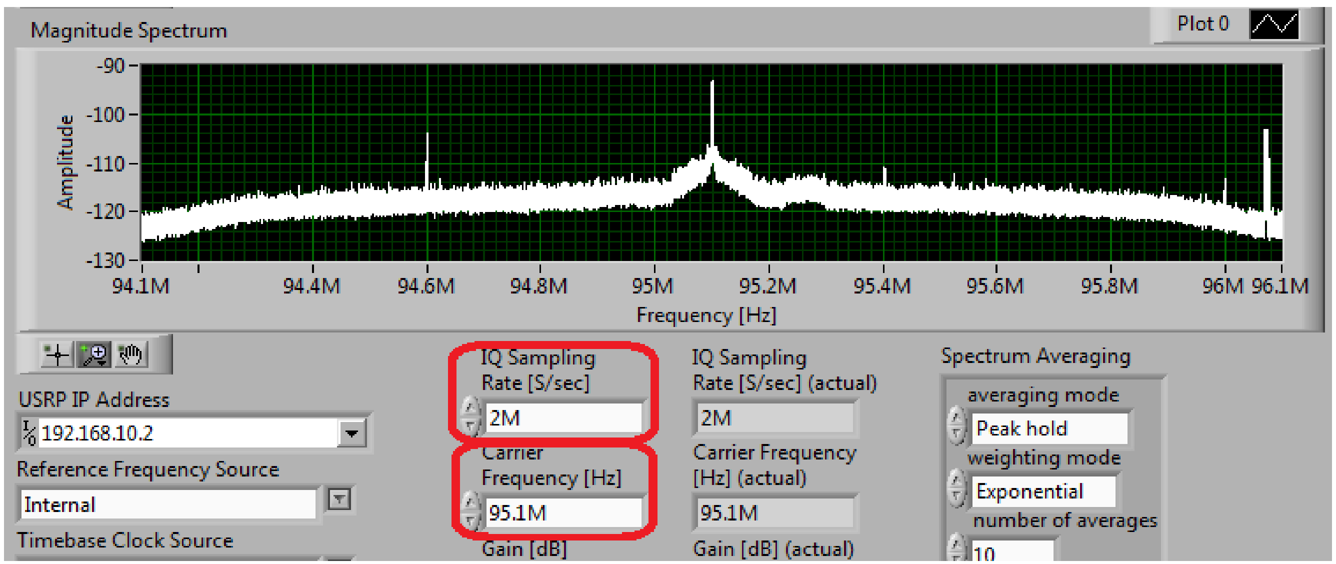
Making Measurements Using Zoom Feature (Concluded)



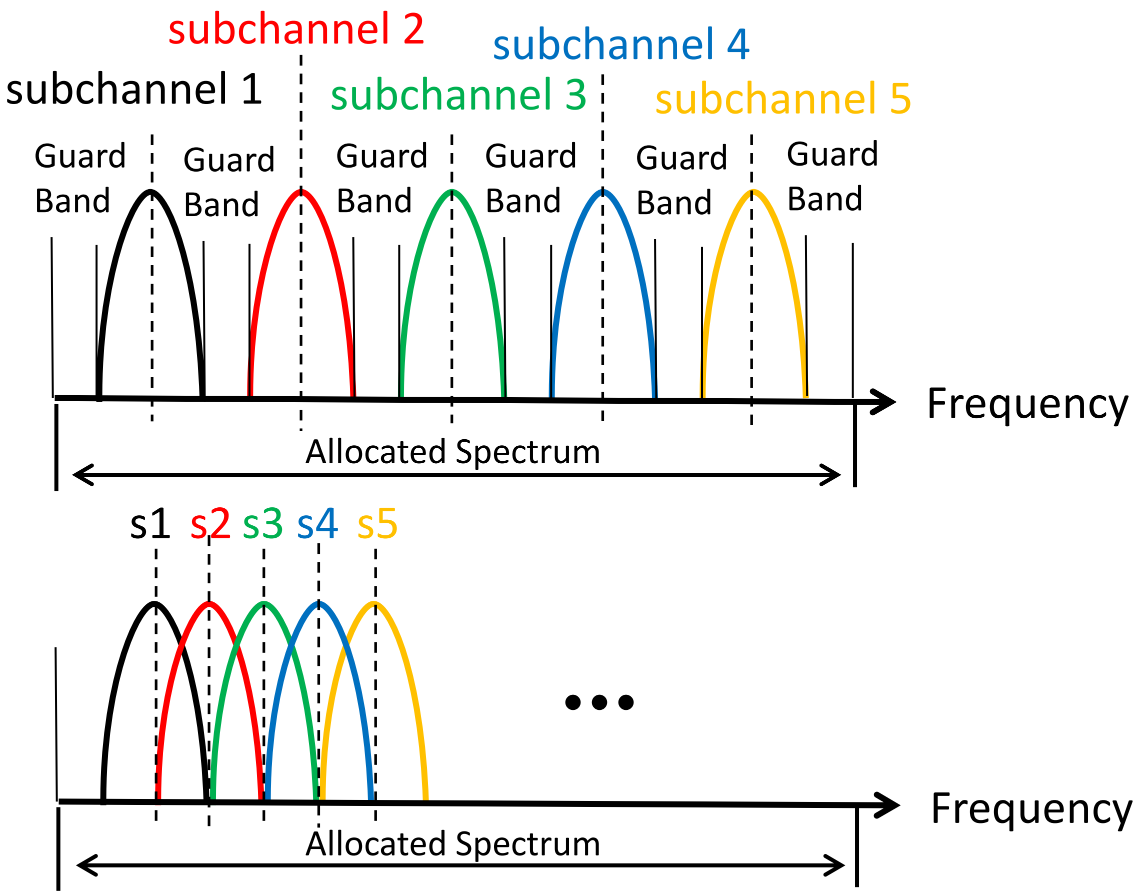
The top display (Acquired Signal) shows the quadrature signals (in-phase is shown in red, and out-of-phase in white) sensed by the radio. The USRP is designed use quadrature modulation and you will be using the radio's capability to adapt this modulation technique to support other modulation approaches.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

Observed Spectrum





Allocating Spectrum to Subchannels



Conventional Multicarrier Modulation (FMDA)

Orthogonal Frequency Division Multiplexing (OFDM)

Debugging Tools in NI LabVIEW

What you need to know to do the Lab...

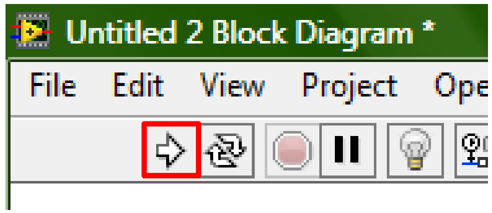
Introduction

- You may encounter two general types of software bugs:
  - Those that prevent the program from running
  - Those that generate bad results or incorrect behavior.
- If LabVIEW cannot run your VI
  - Provides an Error List window with the specific reasons why the VI is broken.
- Bad results or incorrect behavior is based on your desired behaviors for LabVIEW VI and fixing these will require that you use the interactive LabVIEW debugging tools
  - You can watch your code as it executes
  - Observe the data values in the dataflows
  - Control the execution

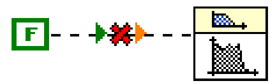
http://www.ni.com/gettingstarted/labviewbasics/debug.htm

Finding The Errors

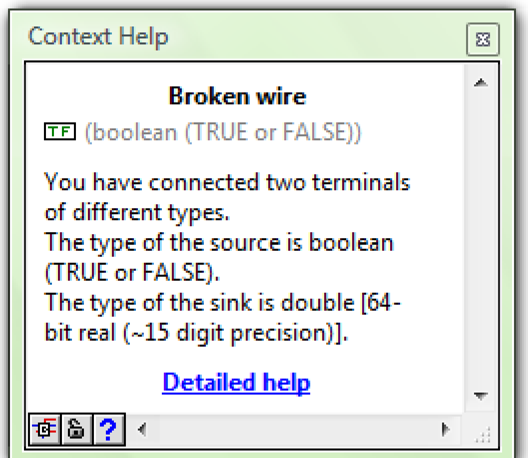
- Changes the run arrow to a broken icon (Click the broken *Run* button or select *View»Error List* to find out why a VI is broken)



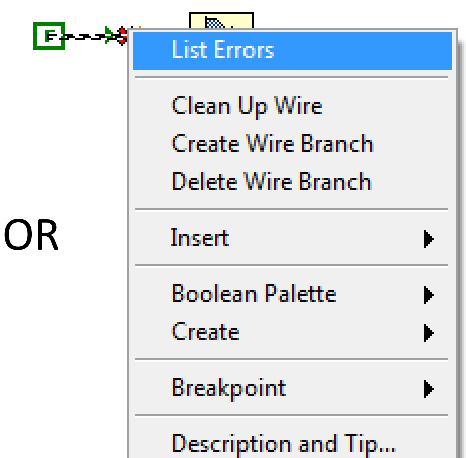
- Marks the data flow with the error

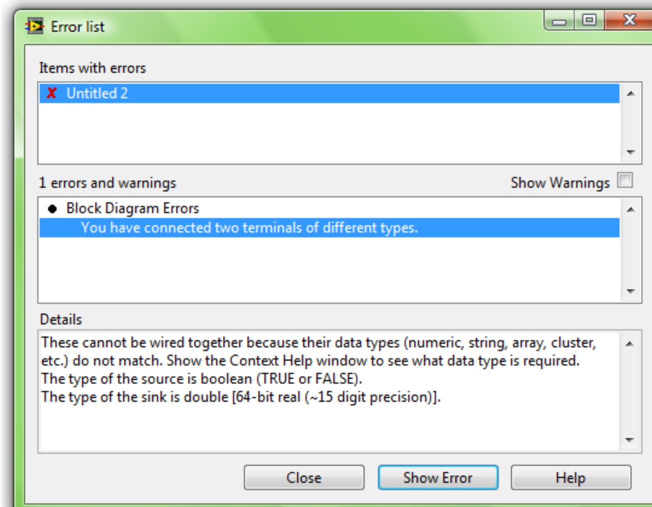


- Provides a description of the error in one of 2 ways(Context Help or right mouse click and select List Errors)



OR

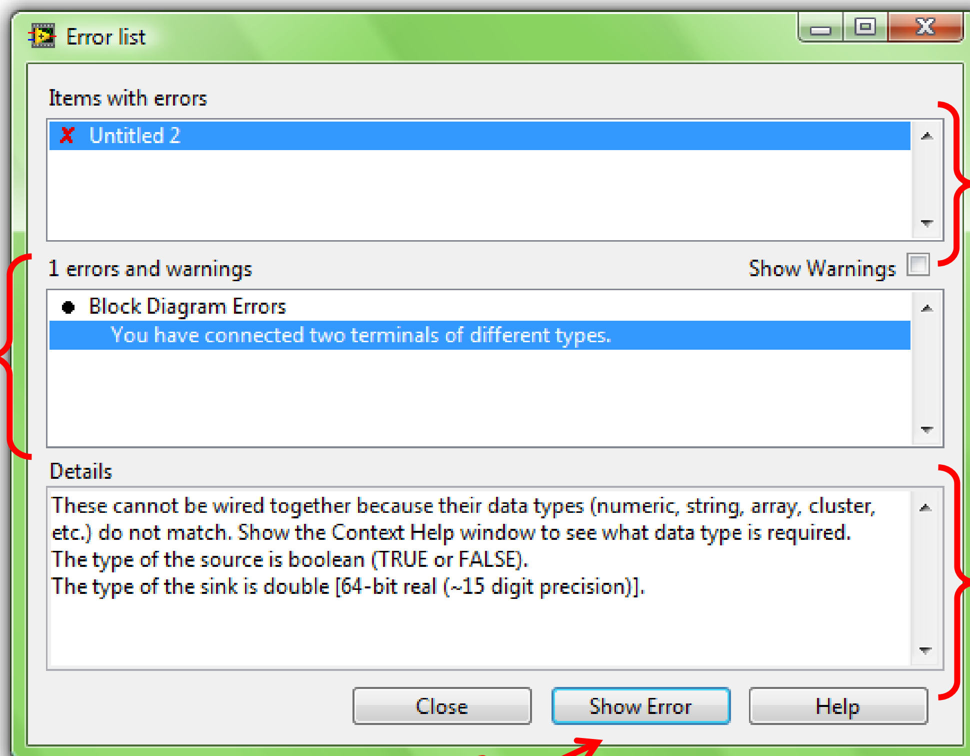




http://www.ni.com/gettingstarted/labviewbasics/debug.htm

Overview of List Errors Window

THE UNIVERSITY of NEW MEXICO



The *Items with errors* section lists the names of all files that have errors. If two or more items have the same name, this section shows the specific application instance for each item.

The *errors and warnings* section lists the errors and warnings for the VI you select in the *Items with errors* section.

The *Details* section describes the errors and in some cases recommends how to correct the errors.

Click the *Show Error* button or double-click the error description to highlight the area on the block diagram or front panel that contains the error.

Click the *Help* button to display a topic in the *LabVIEW Help* that describes the error in detail and includes step-by-step instructions for correcting the error.

http://www.ni.com/gettingstarted/labviewbasics/debug.htm

Ibero-American Science & Technology Education Consortium

GINET Global Innovation Network for Entrepreneurship and Technology

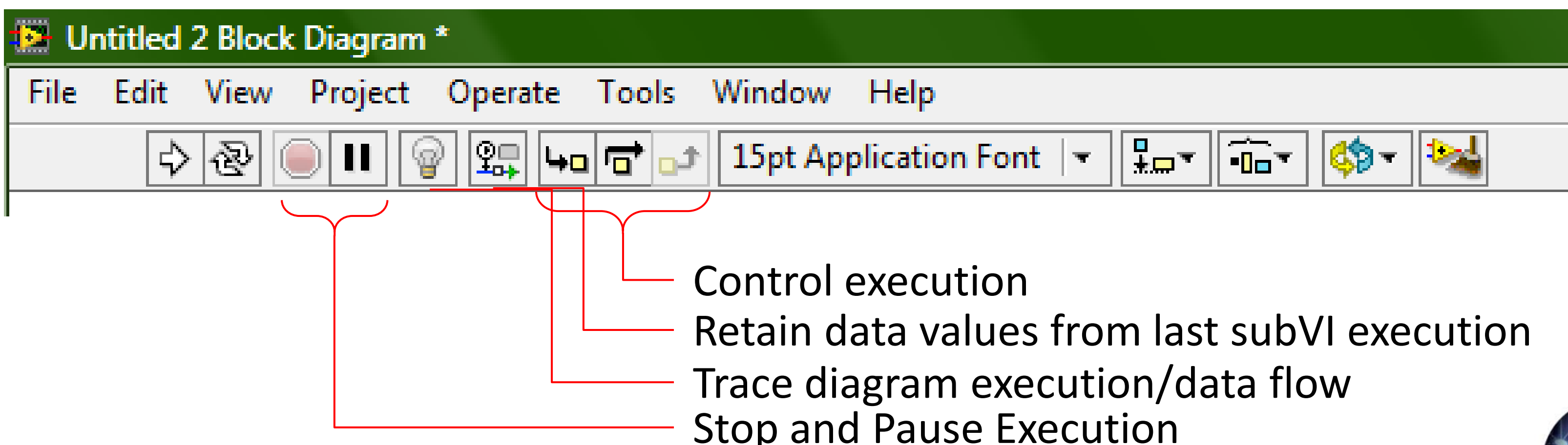
Common Causes of Errors

- The following list contains common reasons why a VI is broken while you edit it:
  - The block diagram contains a broken wire because of a mismatch of data types or a loose, unconnected end. Refer to the *Correcting Broken Wires* topic of the *LabVIEW Help* for information about correcting broken wires.
  - A required block diagram terminal is unwired. Refer to the *Using Wires to Link Block Diagram Objects* topic of the *LabVIEW Help* for information about setting required inputs and outputs.
  - A subVI is broken or you edited its connector pane after you placed its icon on the block diagram of the VI.

http://www.ni.com/gettingstarted/labviewbasics/debug.htm

Fixing Incorrect Behavior

- Next we will deal with using the debugging tools that allow you to trace the execution of a block diagram.
  - Using the trace tool to ensure there are no unintended connections.
  - Controlling the execution flow
  - Use of data probes
- These tools are accessed through the toolbar as shown below.



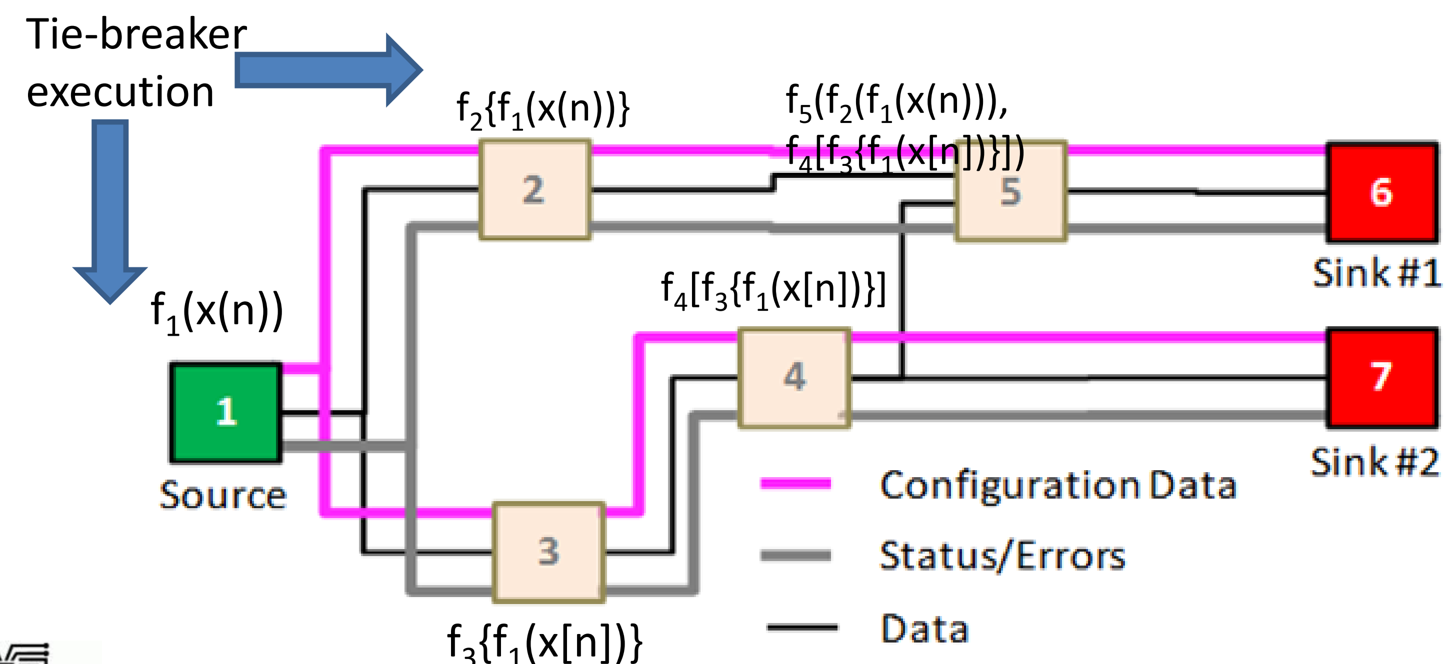
http://www.ni.com/gettingstarted/labviewbasics/debug.htm

Ibero-American Science & Technology Education Consortium

Global Innovation Network for Entrepreneurship and Technology

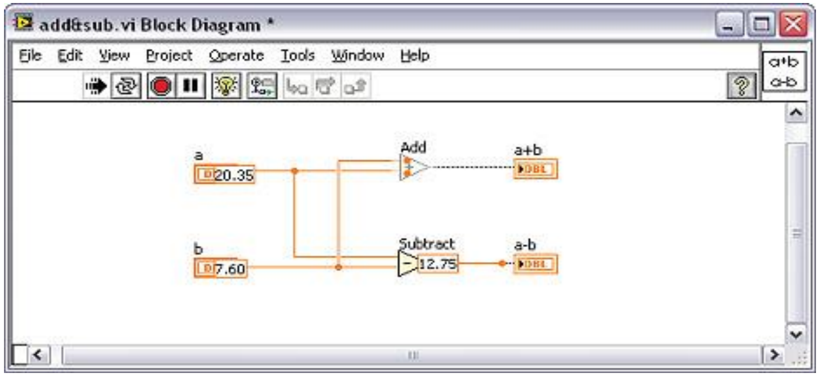
Quick Review
Diagram Execution Details

- LabVIEW the diagram is the set of instructions. LabVIEW executes at a default interval determined by the fastest rate needed for the subVIs to execute properly. (Without a looping structure the diagram executes only once.)
- You need to use a loop to get the diagram to execute repeatedly until the data collection task is complete.



Tracing Execution

- Click the Highlight Execution button to display to confirm execution sequence an animation of the movement of data on the block diagram from one node to another using bubbles that move along the wires, when you run the VI.



Red bubbles ● move along wires.

Note: Execution highlighting greatly reduces the speed at which the VI runs.

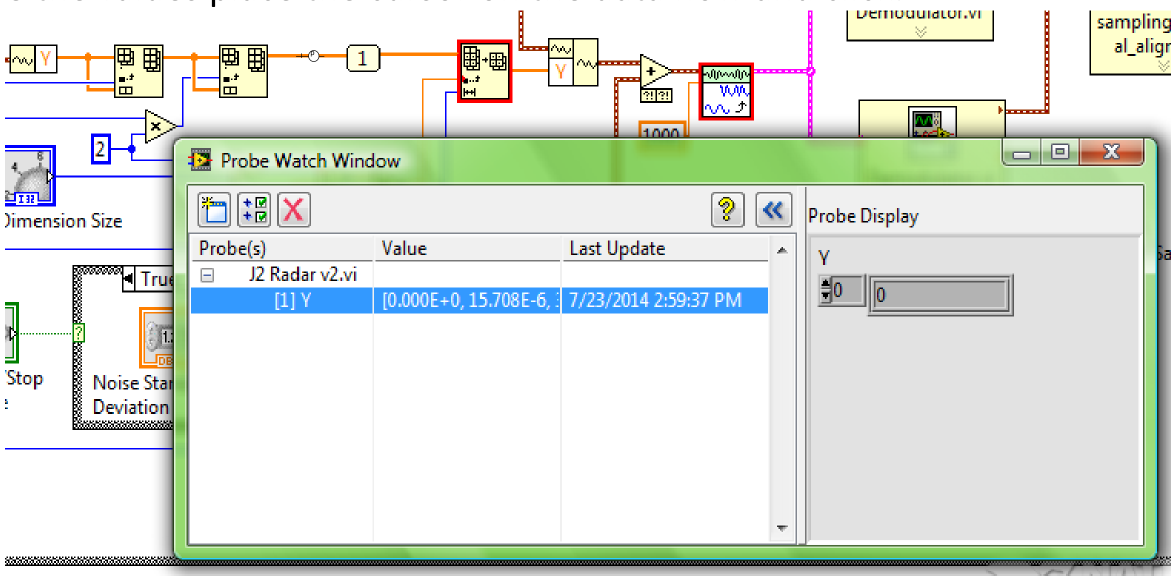
- Click the button again to disable execution highlighting.

- **TIP:** Use execution highlighting with single-stepping to see how data values move from node to node in your VI.

http://www.ni.com/gettingstarted/labviewbasics/debug.htm

Retain Wire Values

- Click the Retain Wire Values button to save the wire values at each point in the flow of execution so that when you place a probe on the wire you can immediately retain the most recent value of the data that passed through the wire.
- Please keep in mind that each data flow has a set of variables associated with it. (Even though you do not get to see them. These variables like any variable in a program get reused each time the diagram is called or executed.
- You must successfully run the VI at least once before you can retain the wire values.
- To see the values place the cursor on the data flow and click.



Click the button again to disable retaining values for probe.

http://www.ni.com/gettingstarted/labviewbasics/debug.htm

Probe Watch Window

- Use the Probe Watch Window with execution highlighting, single-stepping, and breakpoints to determine if and where data is incorrect.
- If data is available, the probe immediately updates and displays the data in the **Probe Watch Window** during execution highlighting, single-stepping, or when you pause at a breakpoint.
- When execution pauses at a node because of single-stepping or a breakpoint, you also can probe the wire that just executed to see the value that flowed through that wire.

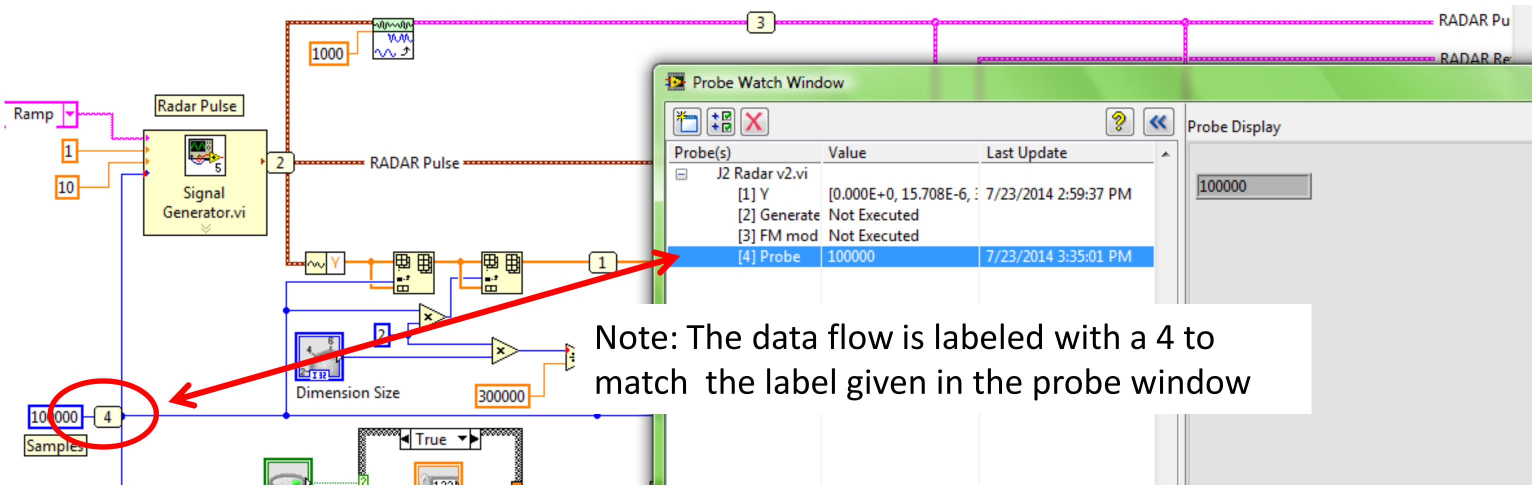
Control Execution

- You can control the execution of the diagram using the
  - Step Into button will follow the execution into a subVI and pause. When you click the Step Into button again, it executes the first action and pauses at the next action of the subVI or structure. Single-stepping through a VI steps through the VI node by node. Each node blinks to denote when it is ready to execute. *You also can press the <Ctrl> and down arrow keys.*
  - Step Over button will execute a node and pause at the next node. By stepping over the node, you execute the node **without** single-stepping through the node. *You also can press the <Ctrl> and right arrow keys.*
  - Step Out button will complete single-stepping through the node entered by stepping into it and navigate to the next node When the VI finishes executing, the Step Out button is dimmed. *You also can press the <Ctrl> and up arrow keys. By stepping out of a node, you.*

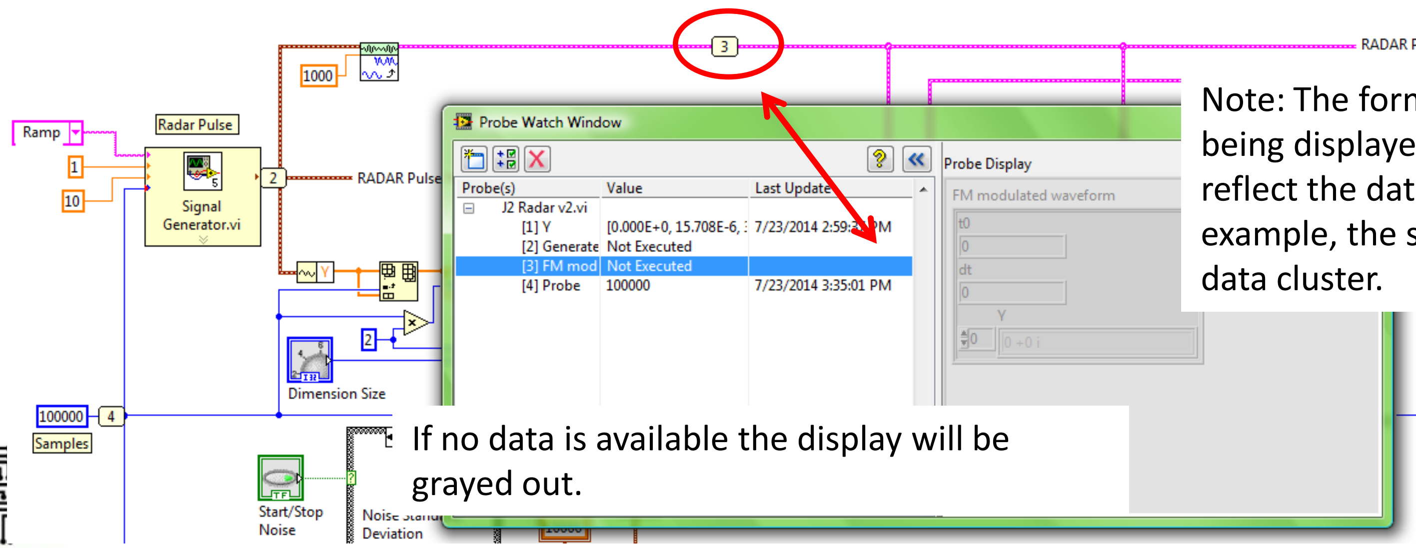
http://www.ni.com/gettingstarted/labviewbasics/debug.htm

THE UNIVERSITY of NEW MEXICO

Probe Windows (Constants & Signals)



Note: The data flow is labeled with a 4 to match the label given in the probe window



Note: The format of the data being displayed changes to reflect the data source. For example, the signal wire has a data cluster.

If no data is available the display will be grayed out.

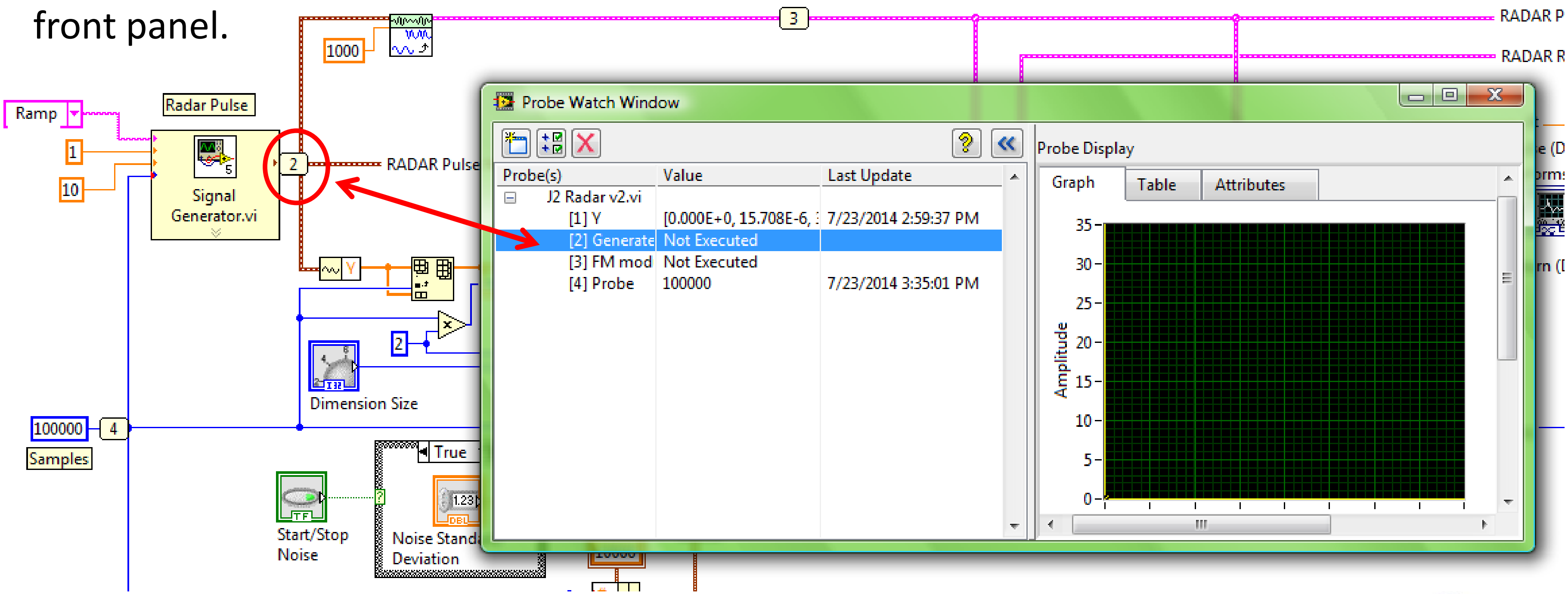
istec Ibero-American Science & Technology Education Consortium

GINET Global Innovation Network for Entrepreneurship and Technology

Probe Windows (Waveforms)

Note: Execution probing a waveform can reduce the speed at which the VI runs. And can cause memory issues. This should be done sparingly.

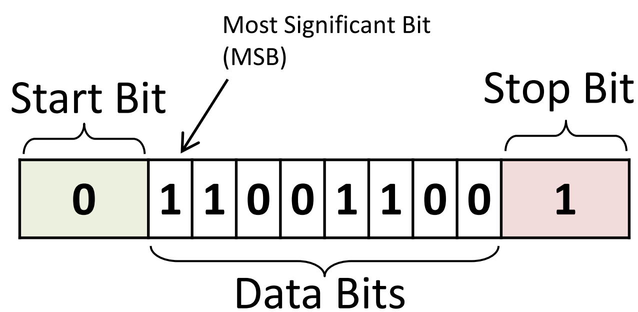
Better approach is to create a temporary waveform chart or graph indicator on the front panel.



Debugging Example

Debugging Example (Background Information)

A digital communication packet structure consists of 10 bits- 1 START bit, 8 DATA bits (one byte), and 1 STOP bit.

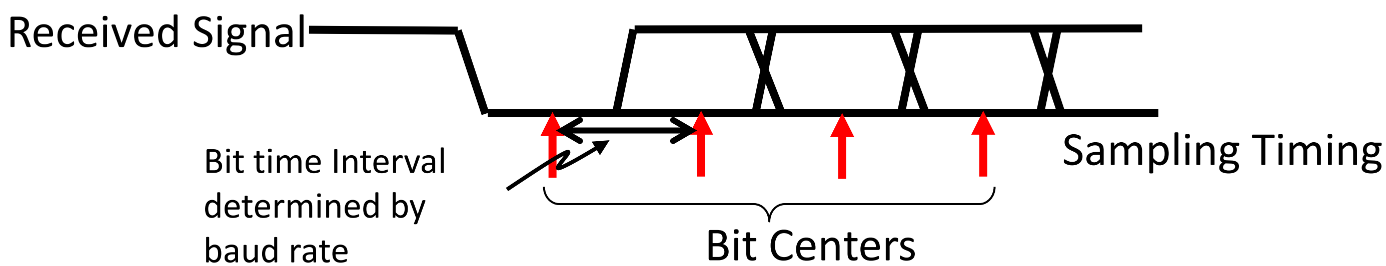


Commonly referred to as 8N1 (8 data bits, no parity, 1 stop bit).

Debugging Example (Serial Communications)

Serial communications follows this general pattern:

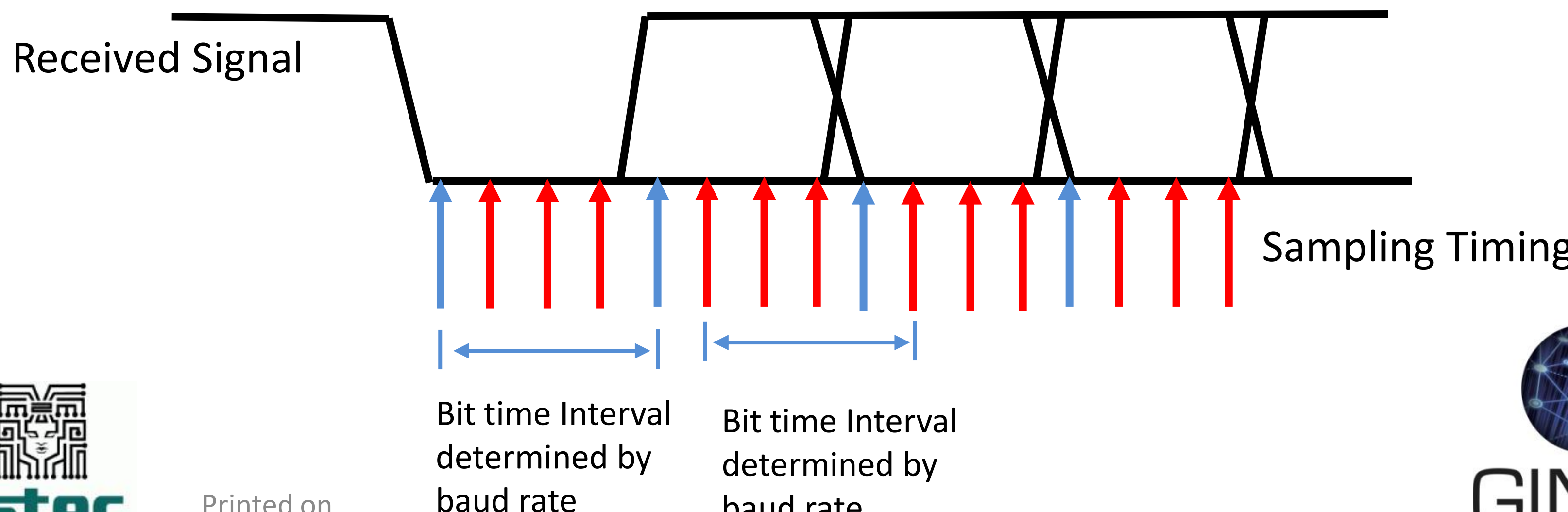
- In order to achieve synchronization with an incoming packet, the communication wire idles in the HIGH (1) state in between packets. Since the START bit is always a LOW, we know a packet has begun when this transition occurs.
- After we synchronize to the start of a packet, we use the known baud rate to estimate the center of each data bit, and sample the voltage of the signal at this point.



- After the receiver decodes the entire data packet, the bit order is reversed (to get the original MSB->LSB) byte
- The stop bit simply returns the communications wire to the original IDLE (HIGH) state, and the receiver begins waiting for the next START bit which signals the beginning of the next packet.

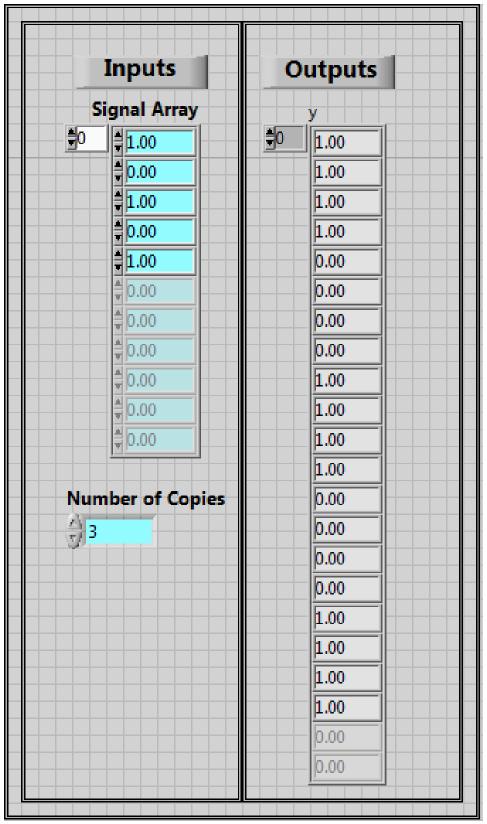
Debugging Example (Bit Stretching)

- Suppose that the signal is transmitted through a serial communications link that causes every fourth bit to flip from 1 to 0 or vice versa.
- If the signal wave form is sampled only once, there is not enough information to determine if the data received is the data sent.
- That is why the parity bit is used in the standard. However, the parity bit only tells us the data is corrupted.
- To make things more robust we may want to send more than one copy of the sampled value. The number depends on the error correction scheme being used. In this example each bit is oversampled by a factor of 4.

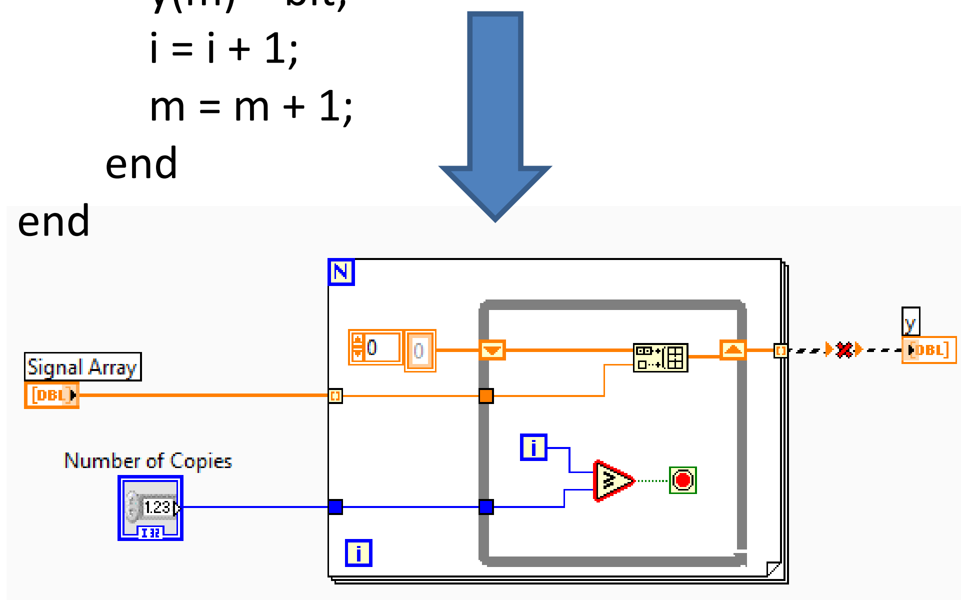


Debugging Example (Bit Stretching)

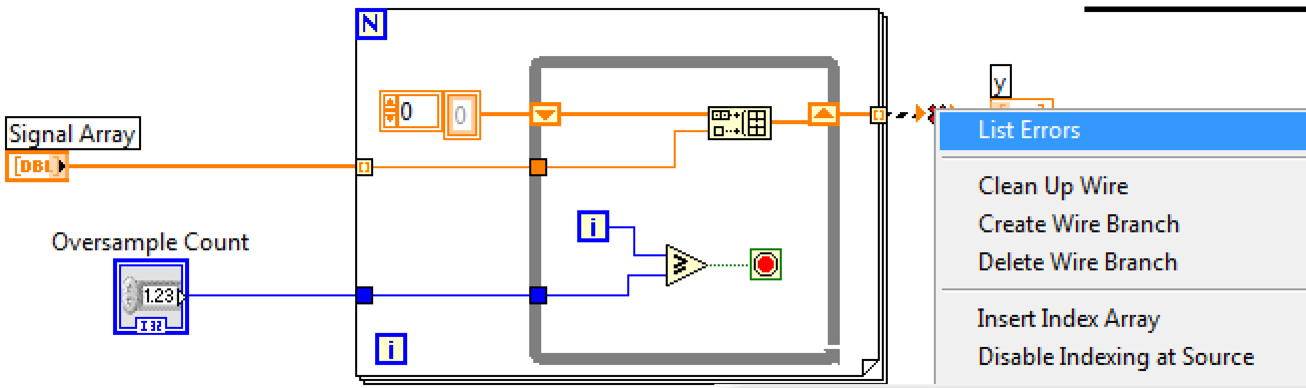
To this end we have developed the following VI to stretch the bits by making the number of copies specified in for each bit in the signal array.



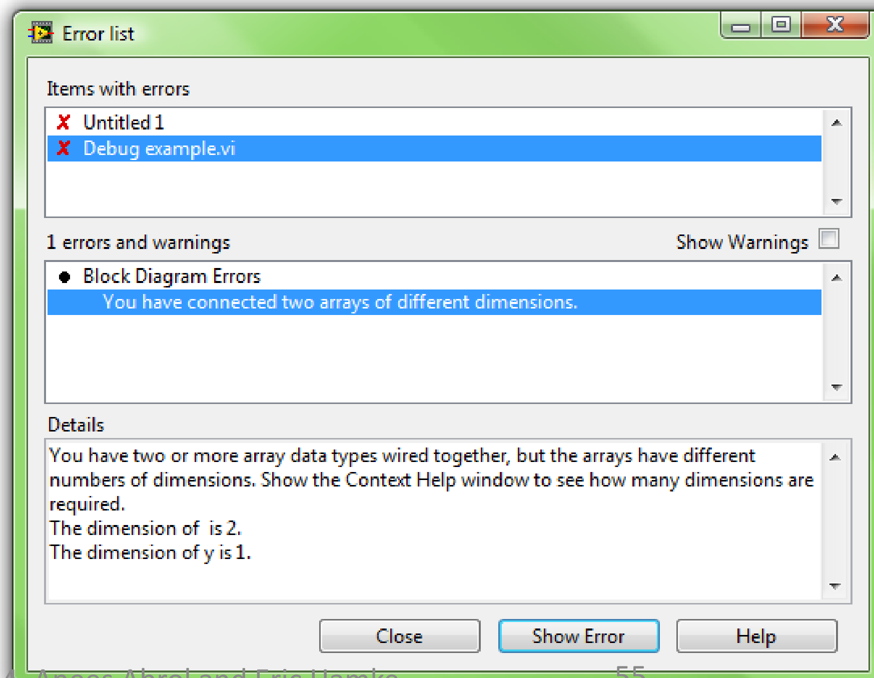

```
m = 0;
for k = 1 to length of the array
      bit = signalArray(k);
      i = 0;
      while not(i ≥ NumberOfCopies)
          y(m) = bit;
          i = i + 1;
          m = m + 1;
       end
end
```



Debugging Example (Recovering Error Information)



Right mouse click on the [broken wire icon] and select list errors



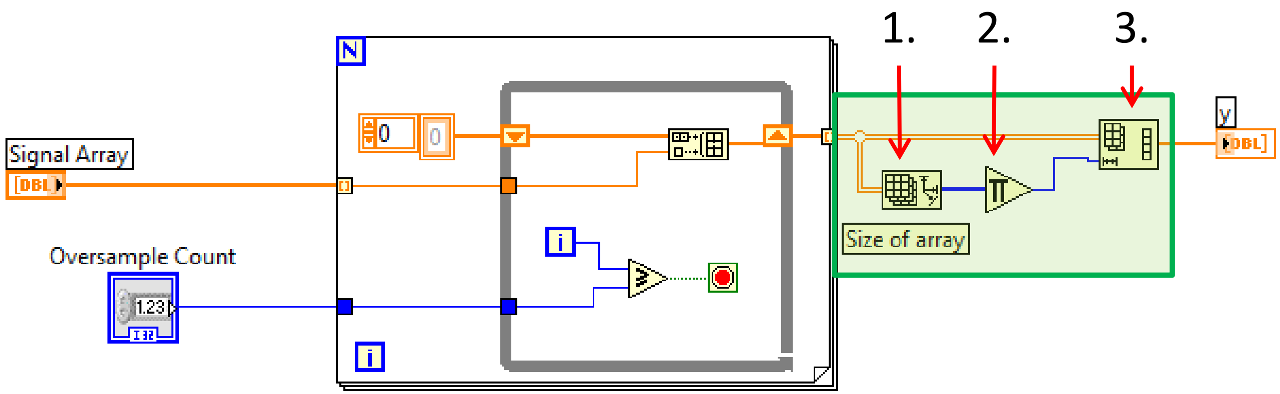
The error is stating that we have an array

$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

and it is expecting

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

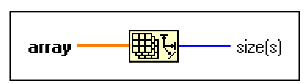
Debugging Example (Fixing Error)



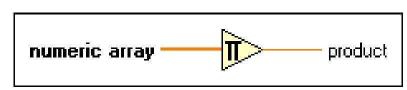
The fix: Reshape the matrix into a vector

Step 1: We need to know the dimensions of the matrix. We can find this using the

Array Size Function

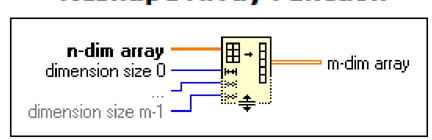


Step 2: The resulting vector will have a length of the number of rows times with number of columns. This found using an **Multiply Array Elements Function** on the output of **Step 1**.

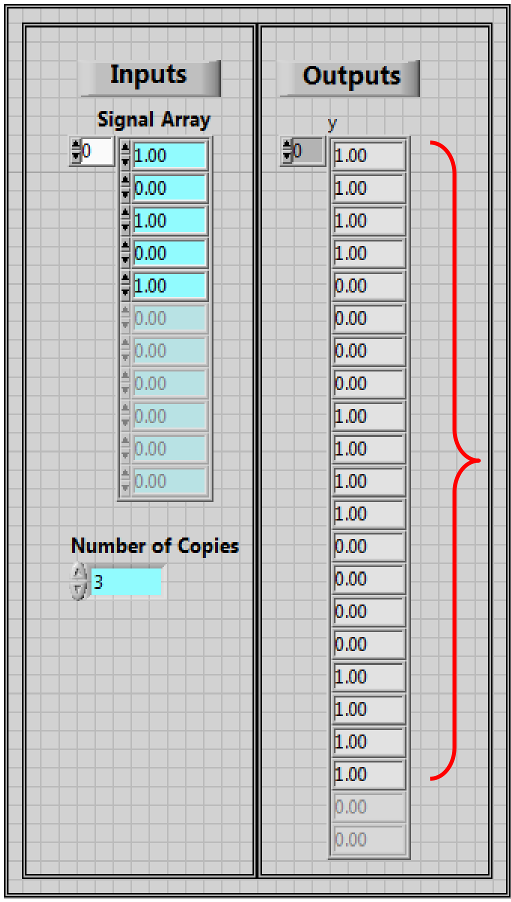


Step 3: We now have what we need to reshape the matrix using the

Reshape Array Function



Debugging Example (Unexpected Behavior)



For the bit string $\begin{bmatrix}1 & 0 & 1 & 0 & 1\end{bmatrix}$ with three copies we should get

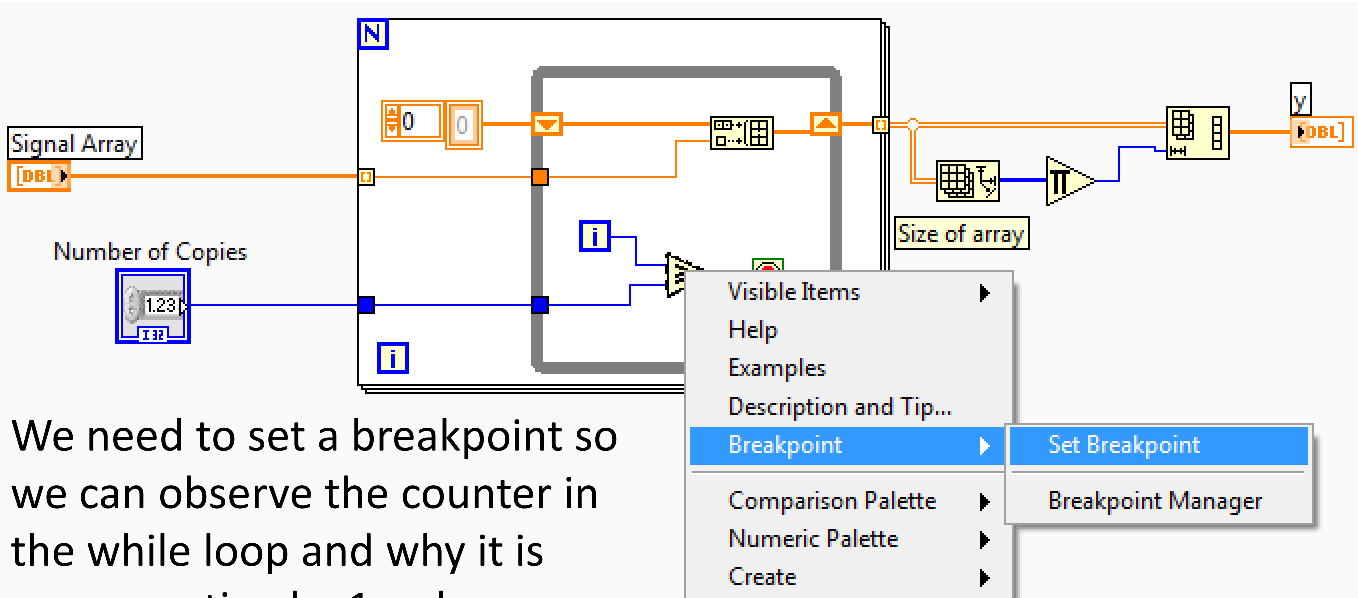
$$\begin{bmatrix}1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1\end{bmatrix}$$

However, the VI outputs

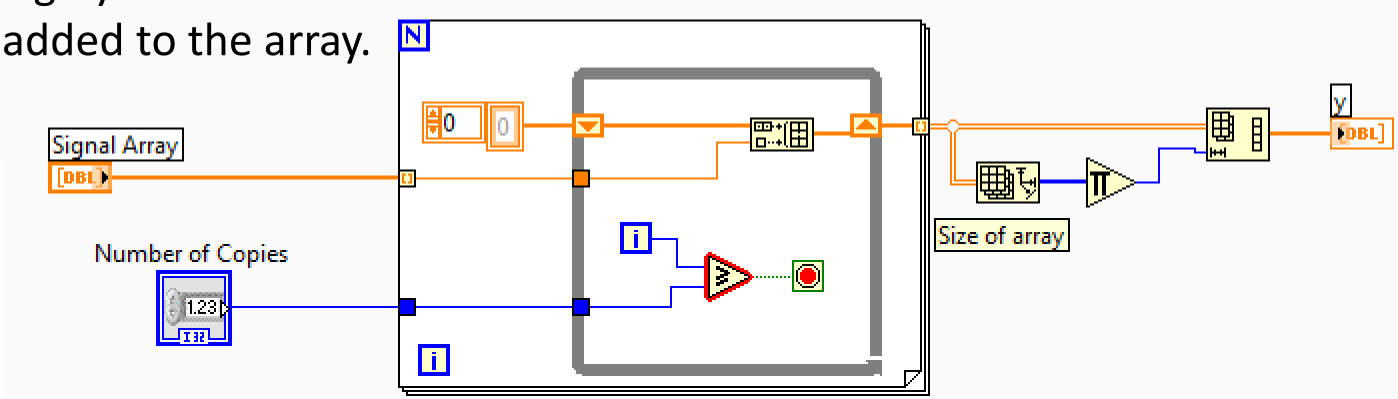
$$\begin{bmatrix}1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1\end{bmatrix}$$

Each bit is being copied 4 instead of 3 times.

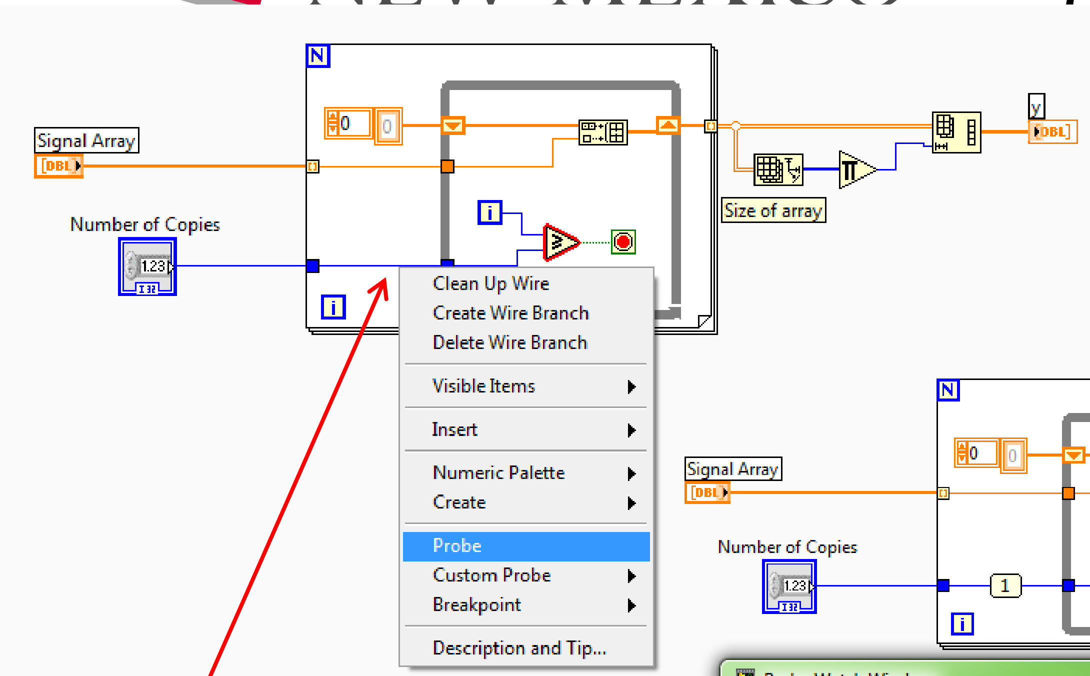
Debugging Example (Observing The Behavior)

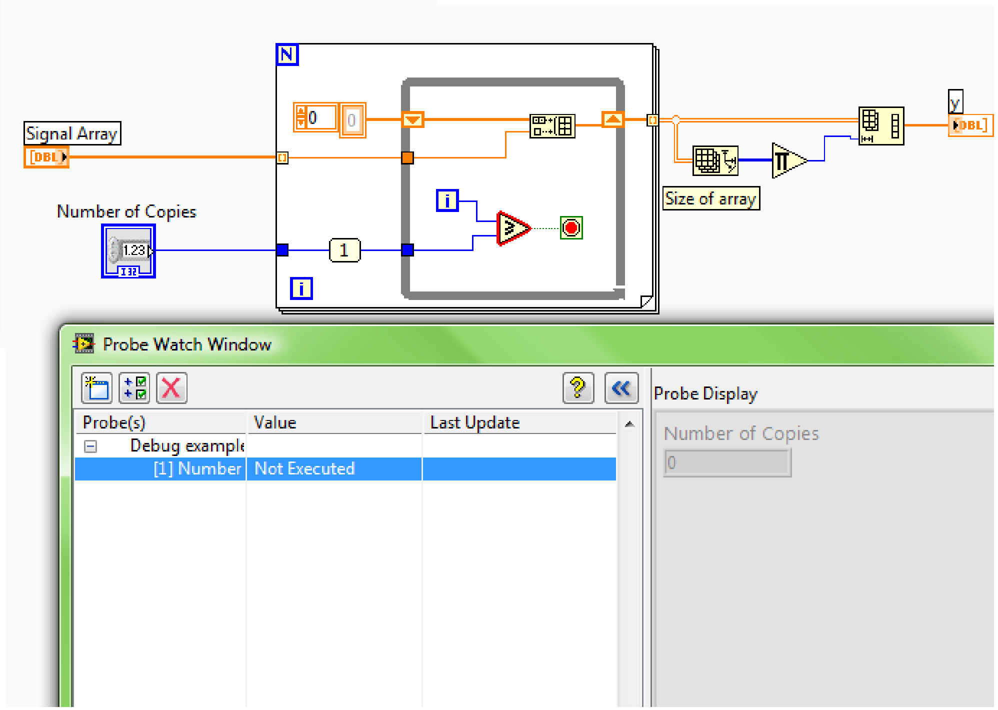


We need to set a breakpoint so we can observe the counter in the while loop and why it is over counting by 1 or how an extra bit is added to the array.



Debugging Example (Placing 1st Probe)

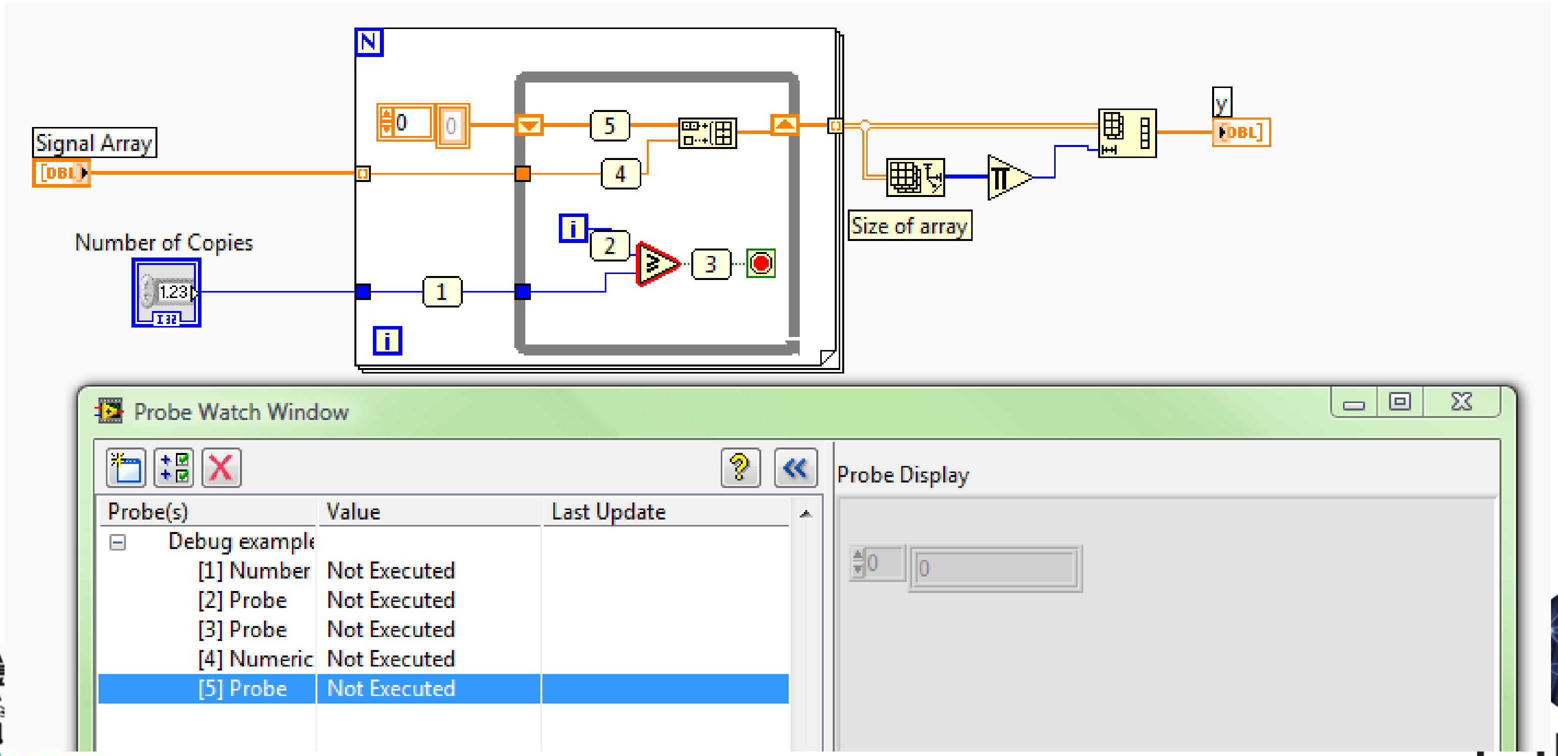




Right clicking on the wire we wish to observe and selecting probe from the menu will result the wire receiving a label and the probe watch window will appear.

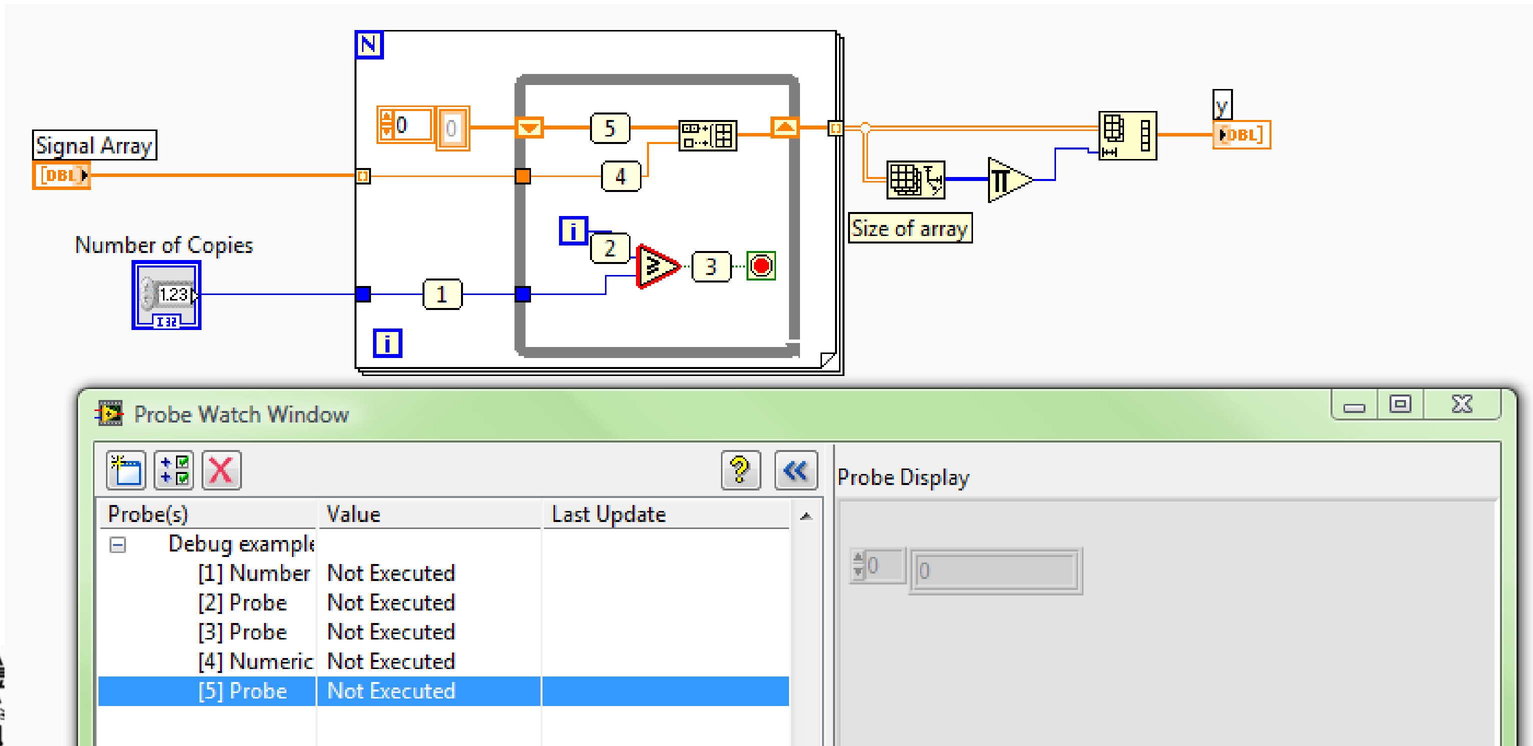
Debugging Example (Placing Remaining Probe)

We not only wish to see the number of copies input but would like to observe the counter (probe 2) and the results of the comparison (probe 3). The result of the comparison will determine if the loop executes another time (FALSE) or stops (TRUE).



Debugging Example (Observing the Matrix)

In addition we do not know if the problem is the way the matrix is being build up. So have placed a probe at labels 4 (current element) and 5 (final matrix from the last pass).



Debugging Example (First Pass Through Diagram)

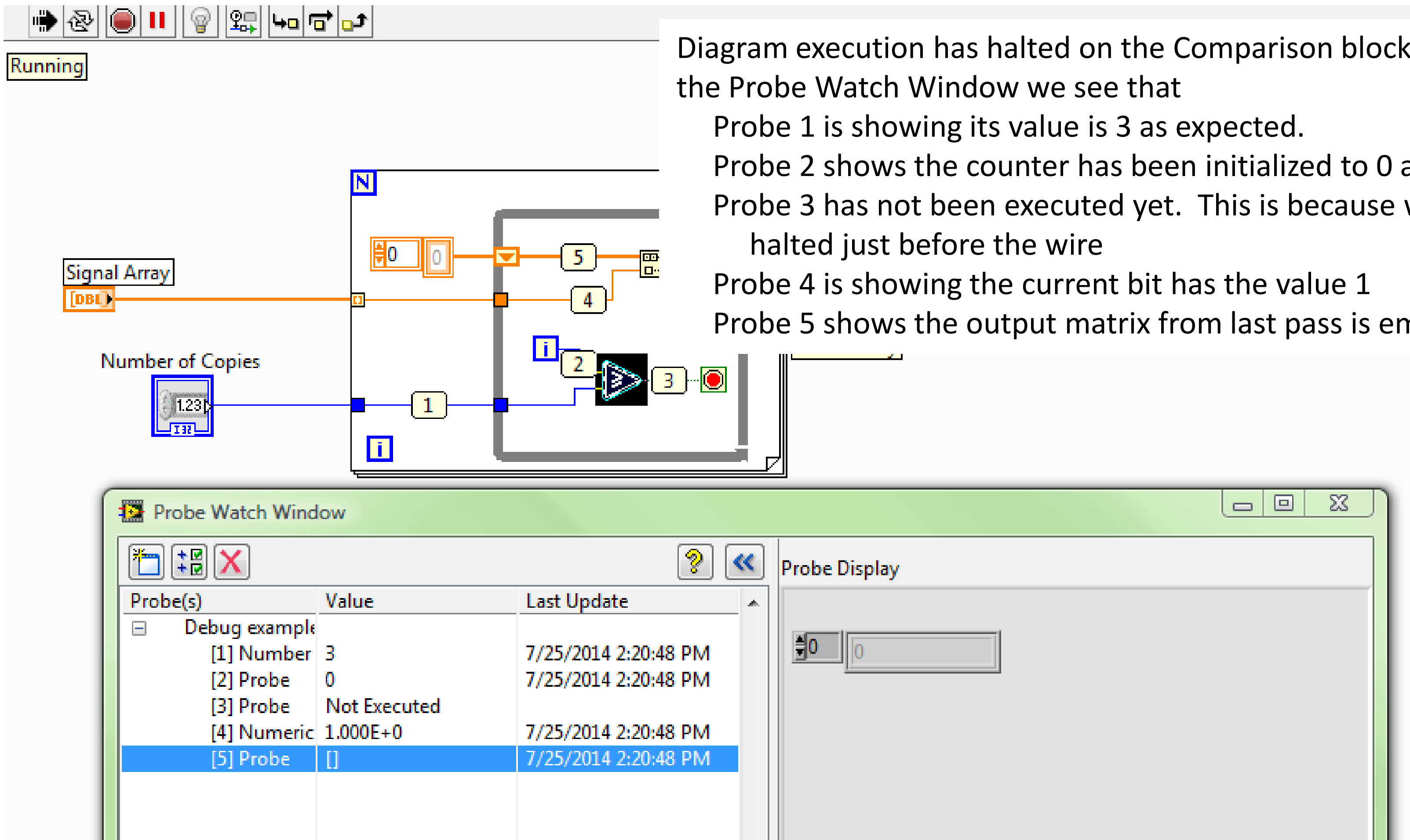
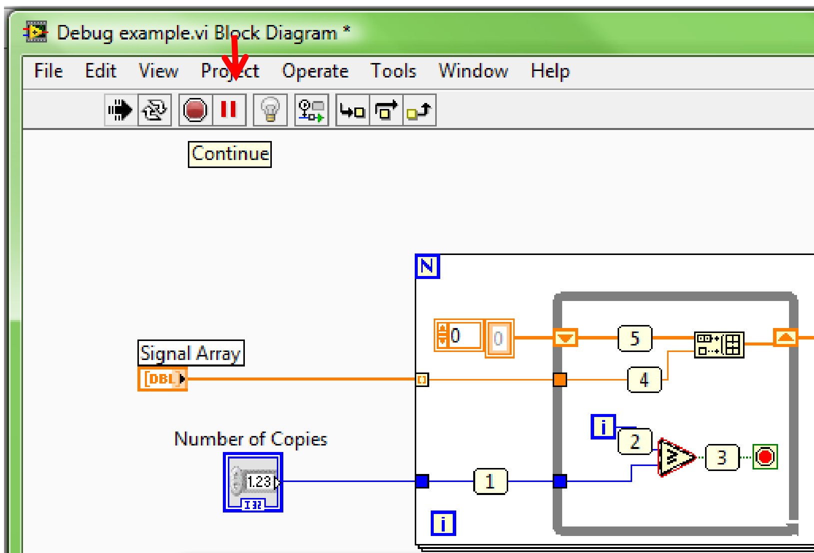


Diagram execution has halted on the Comparison block. Observing the Probe Watch Window we see that

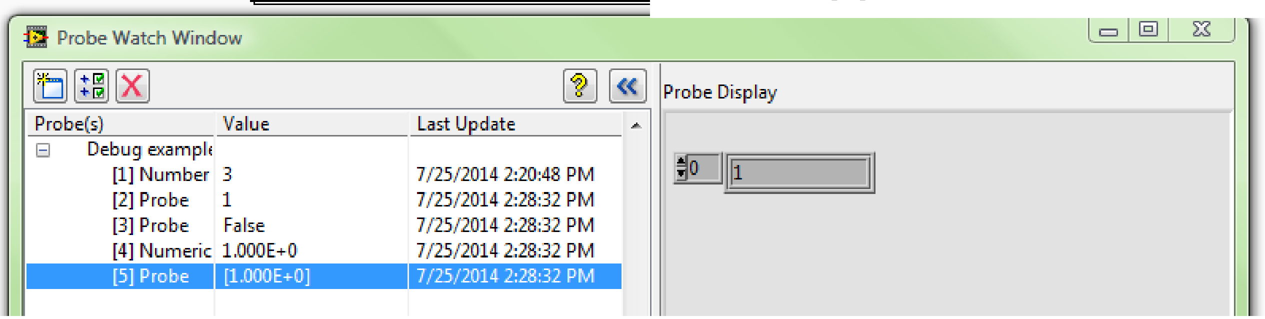
- Probe 1 is showing its value is 3 as expected.
- Probe 2 shows the counter has been initialized to 0 as expected.
- Probe 3 has not been executed yet. This is because we are halted just before the wire
- Probe 4 is showing the current bit has the value 1
- Probe 5 shows the output matrix from last pass is empty.

Debugging Example (Second Pass Through Diagram)

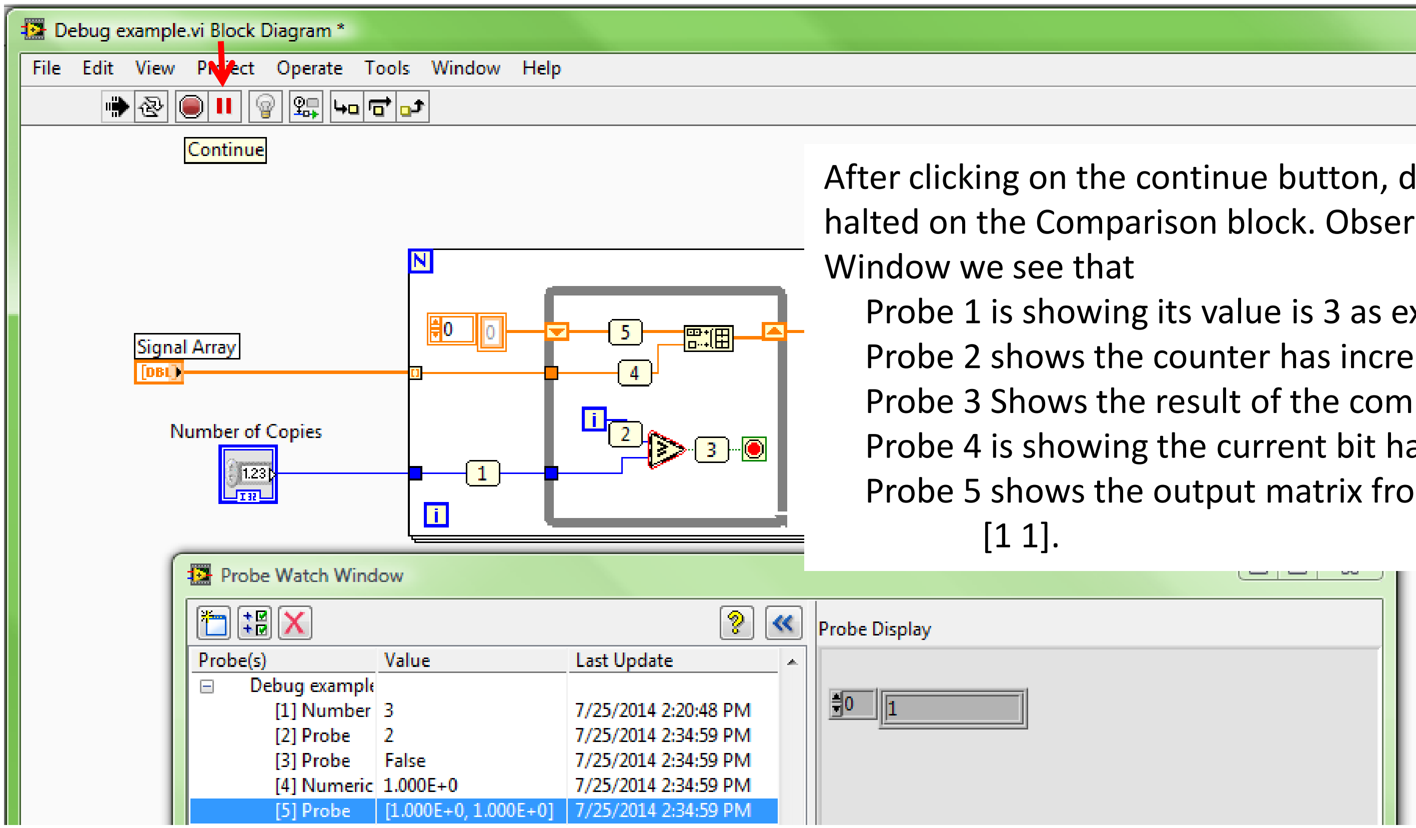


After clicking on the continue button, diagram execution has again halted on the Comparison block. Observing the Probe Watch Window we see that

- Probe 1 is showing its value is 3 as expected.
- Probe 2 shows the counter has incremented by 1.
- Probe 3 Shows the result of the comparison from the last pass
- Probe 4 is showing the current bit has the value 1
- Probe 5 shows the output matrix from last pass has elements [1].



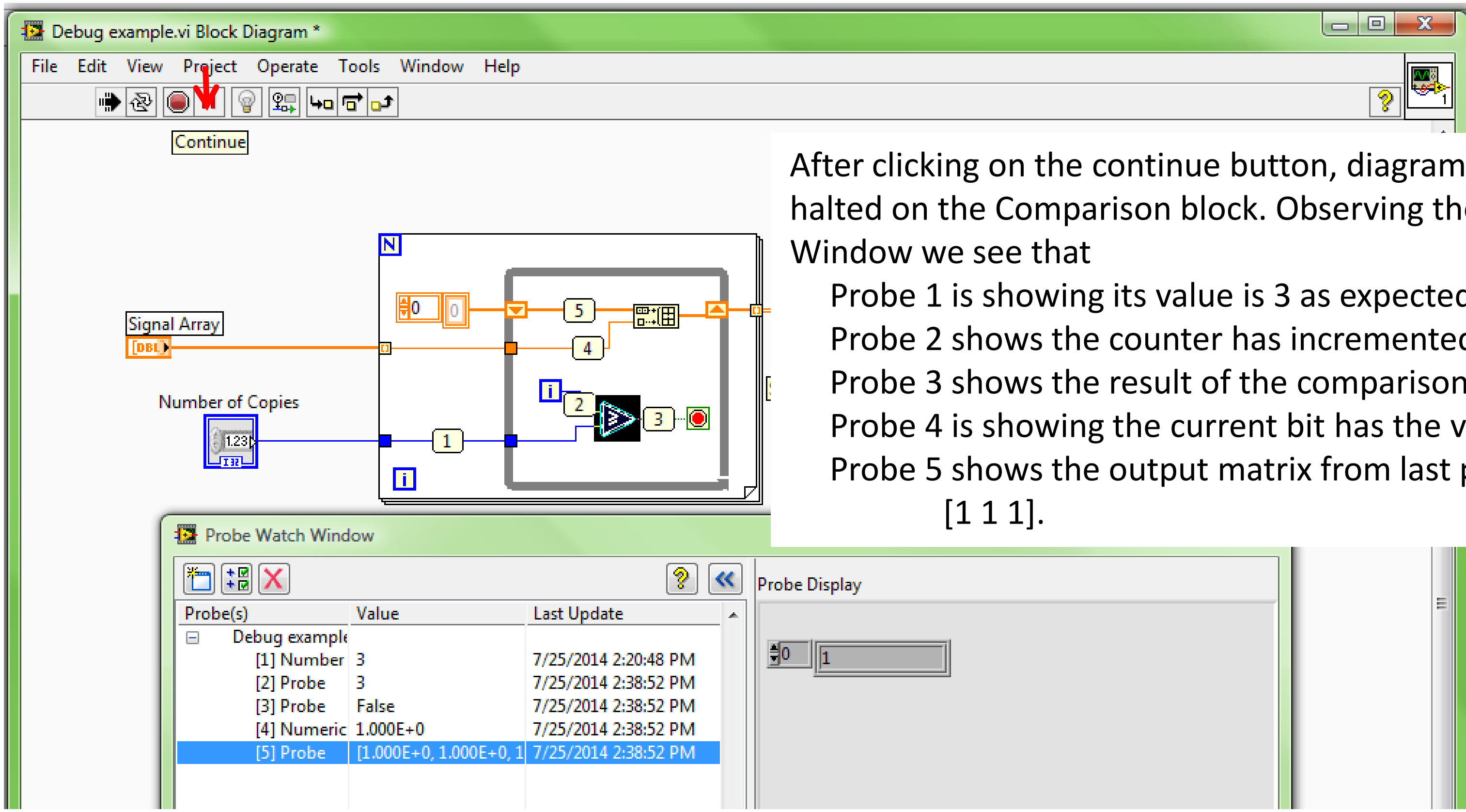
Debugging Example (Third Pass Through Diagram)



After clicking on the continue button, diagram execution has again halted on the Comparison block. Observing the Probe Watch Window we see that

- Probe 1 is showing its value is 3 as expected.
- Probe 2 shows the counter has incremented by 1 to 2.
- Probe 3 Shows the result of the comparison from the last pass
- Probe 4 is showing the current bit has the value 1
- Probe 5 shows the output matrix from last pass has elements [1 1].

Debugging Example (Fourth Pass Through Diagram)



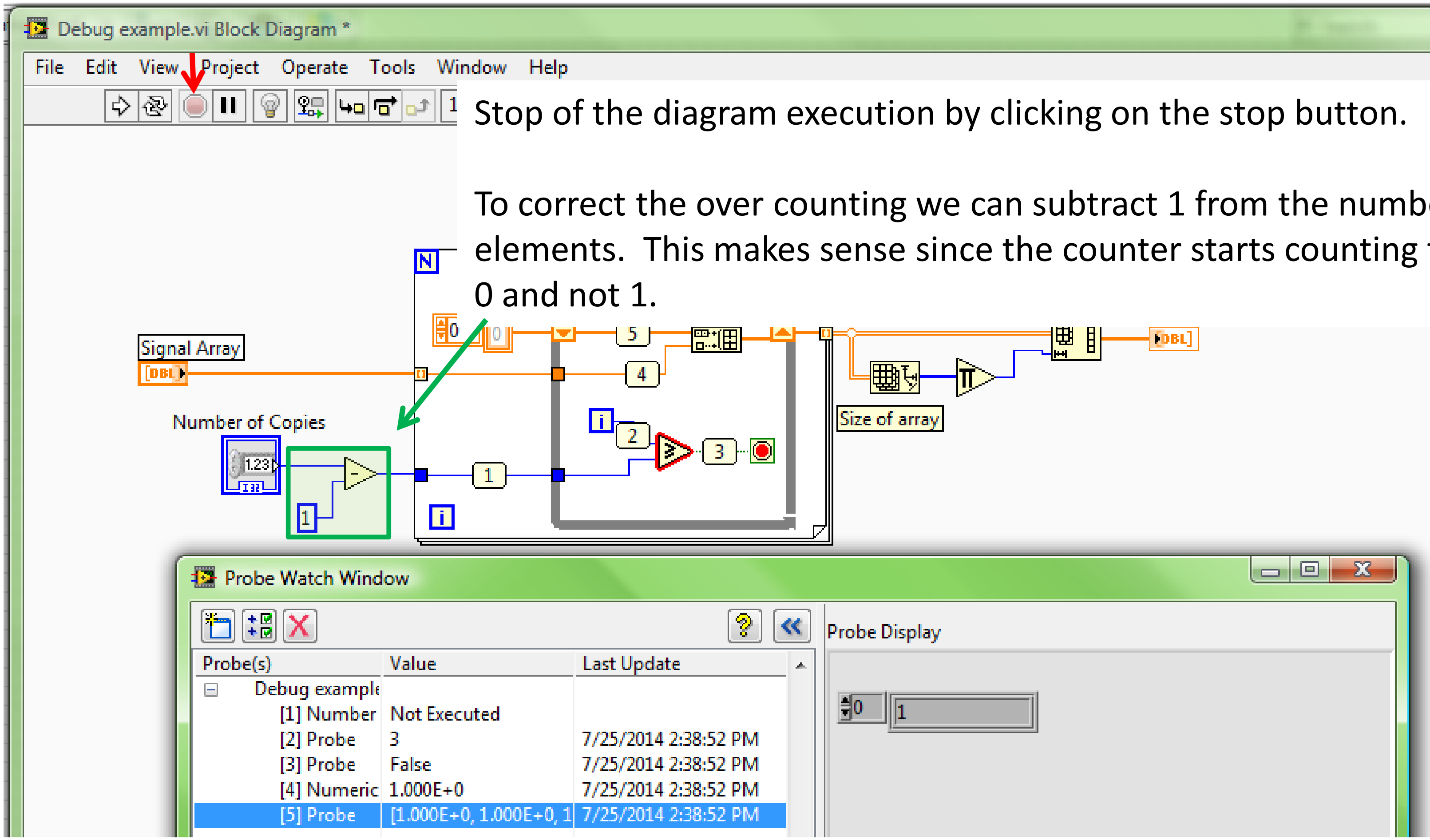
After clicking on the continue button, diagram execution has again halted on the Comparison block. Observing the Probe Watch Window we see that

- Probe 1 is showing its value is 3 as expected.
- Probe 2 shows the counter has incremented by 1 to 3.
- Probe 3 shows the result of the comparison from the last pass
- Probe 4 is showing the current bit has the value 1
- Probe 5 shows the output matrix from last pass has elements [1 1 1].

At this point we know that the comparison will result in a TRUE ending the loop and the matrix will receive an additional element because of this.

THE UNIVERSITY *of* NEW MEXICO

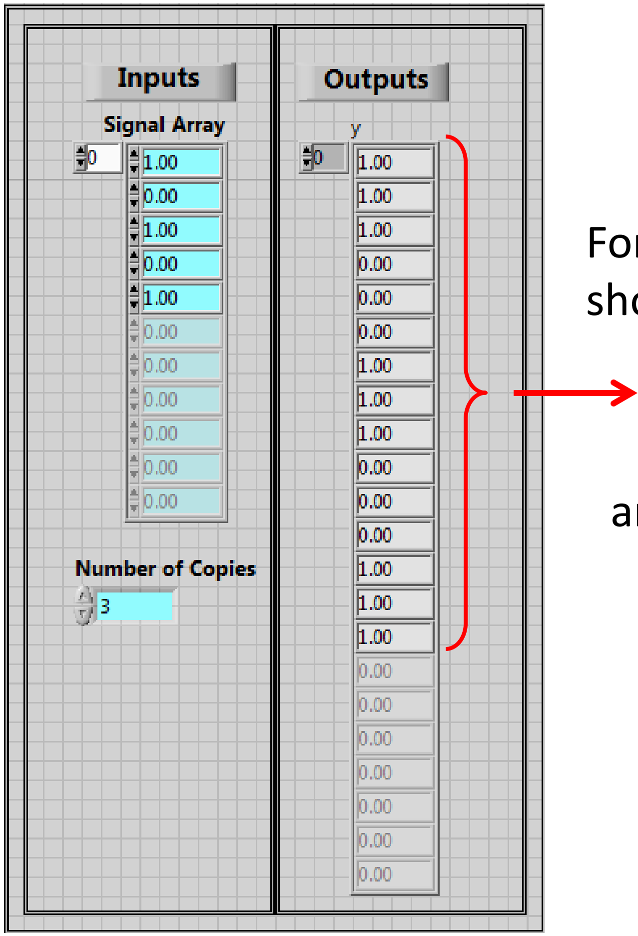
Debugging Example (The Correction)



Stop of the diagram execution by clicking on the stop button.

To correct the over counting we can subtract 1 from the number of elements. This makes sense since the counter starts counting from 0 and not 1.

Debugging Example (Confirming the Fix)



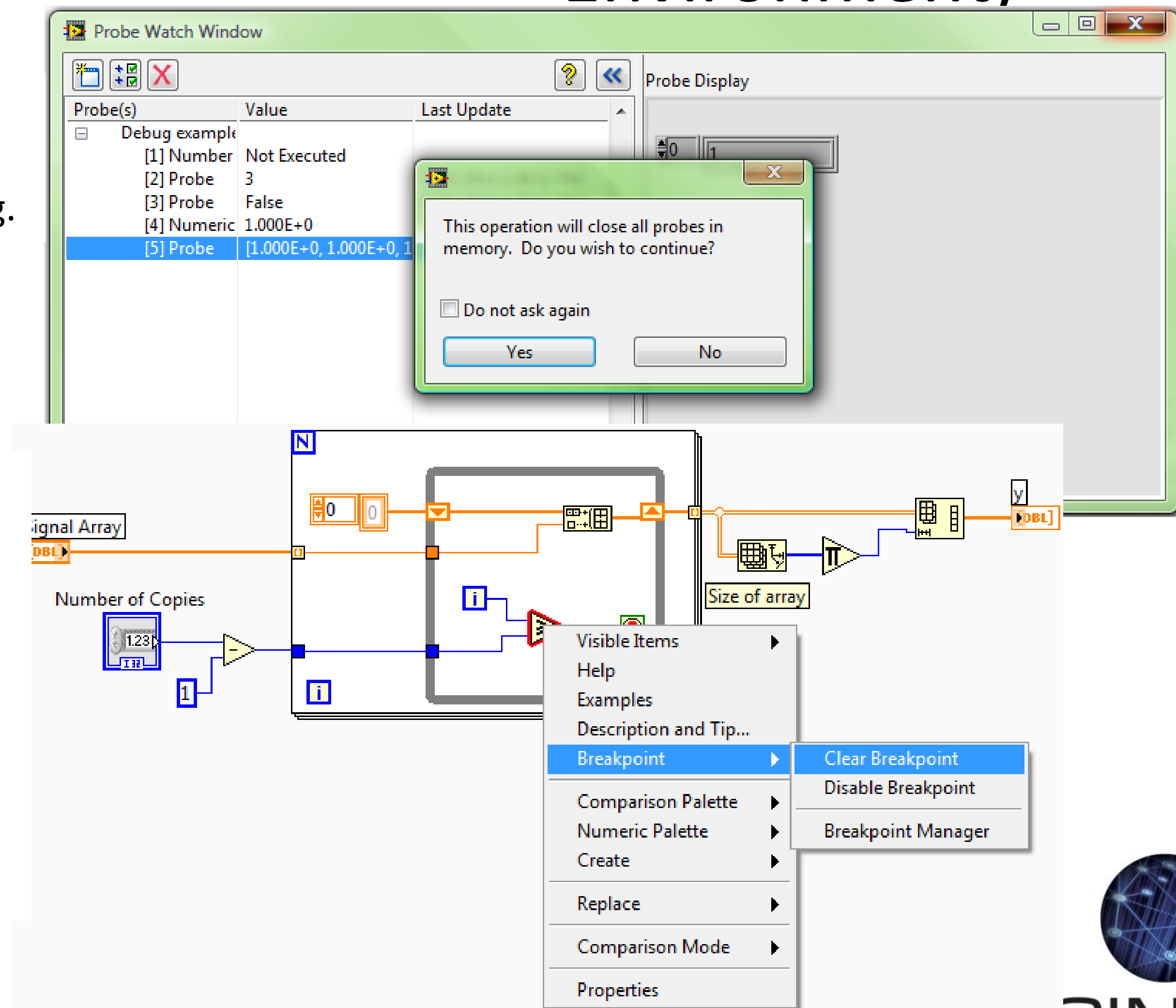
For the bit string $\begin{bmatrix}1 & 0 & 1 & 0 & 1\end{bmatrix}$ with three copies we should get

$$\begin{bmatrix}1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1\end{bmatrix}$$

and as can be seen each bit is being copied just 3 times.

Debugging Example (Leaving the Debugging Environment)

Closing the probe window will remove all the probes and labels from the drawing.



You also will need to remove or clear the breakpoint on the comparison block.

You MUST do this otherwise it will be saved with the corrected file and will be come a nuisance in the future.

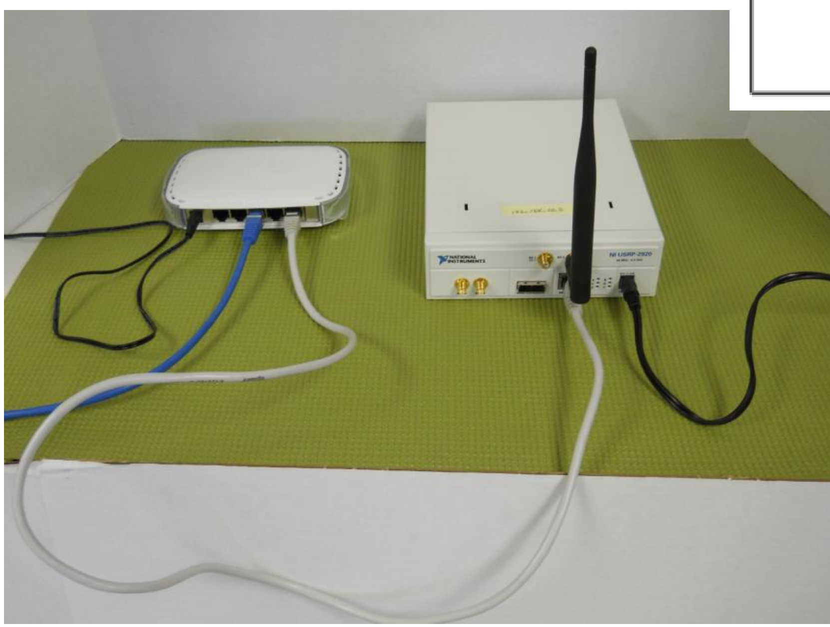
Introduction to USRP

The USRP

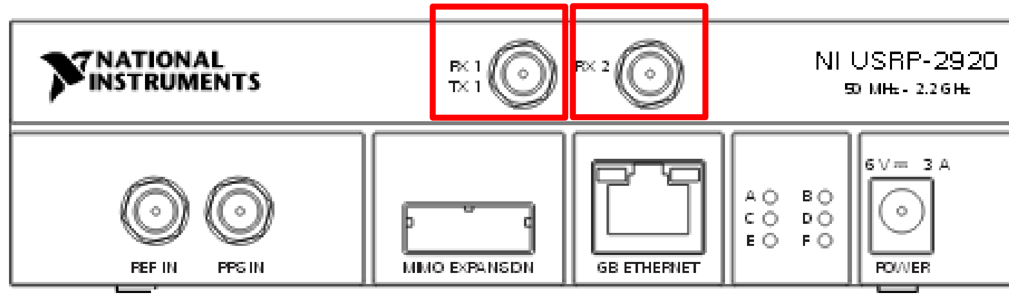
Universal Software Radio Peripheral (USRP) is a software-programmable radio transceiver and a secondary receiver .
• Programmable with NI LabVIEW software,
• Physical layer communication and spectrum monitoring

USRP Antennas

All of the labs will be using a carrier frequency in the MHz ranges. So you should be using the VERT400.



TX1/RX1 RX2



VERT400 Antenna

Tri-Band Vertical Antenna

(144 MHz, 400 MHz, 1200 MHz)

VERT2450 Antenna

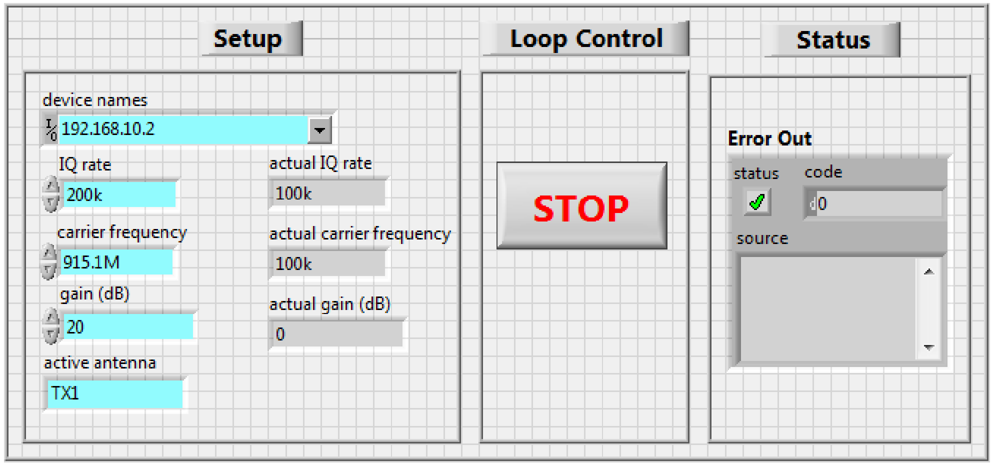
Dual-Band Vertical Antenna

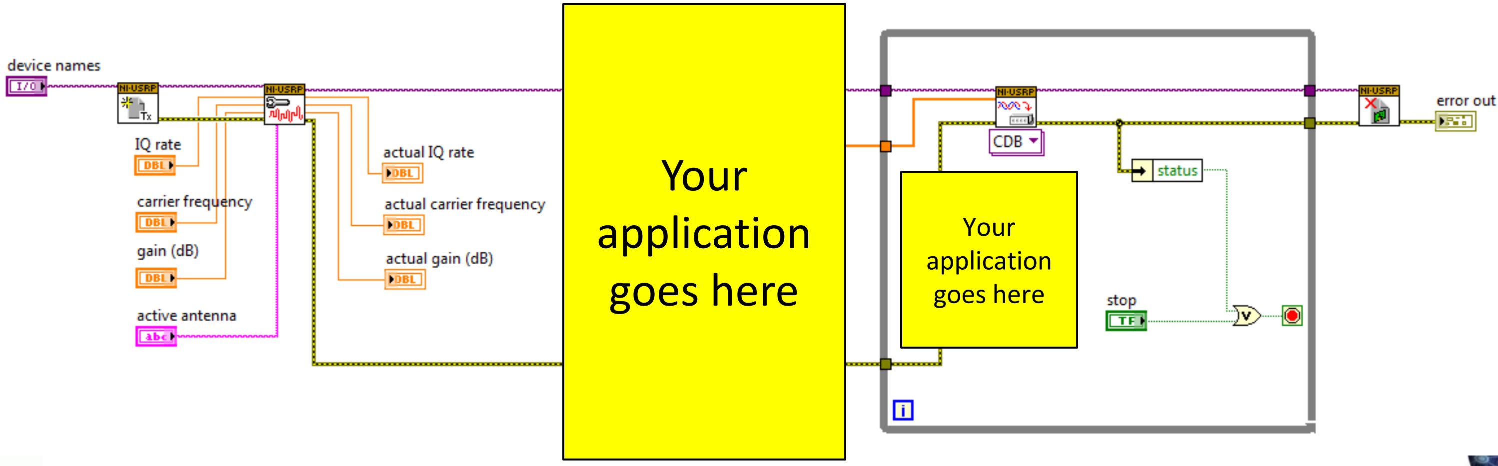
(2.4 GHz, 5 GHz)

USRP Transmitter (Transmitter Template)

The transmitter template consists of 4 elements:

- USRP Transmitter Configuration
- While-loop to control execution of lab.
- Write to the transmitter buffer
- USRP shutdown & status reporting

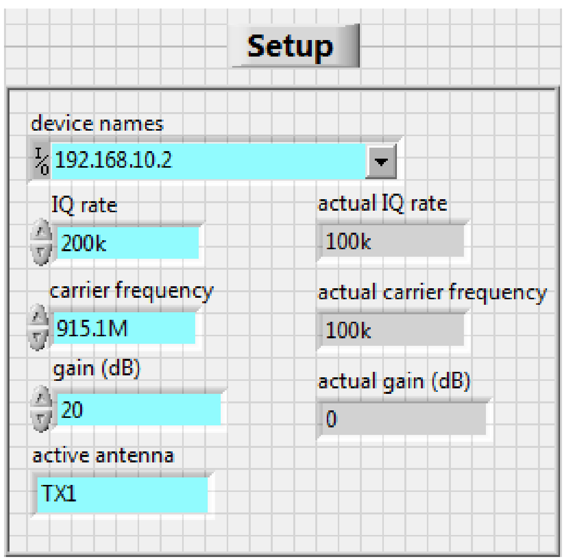


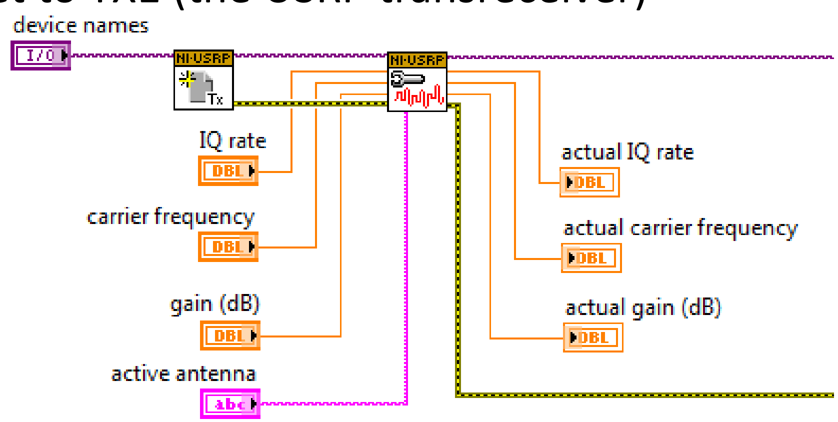


USRP Transmitter (USRP Transmitter Config.)

The front panel for each application will have an USRP configuration panel. The panel supports entering the following radio parameters:

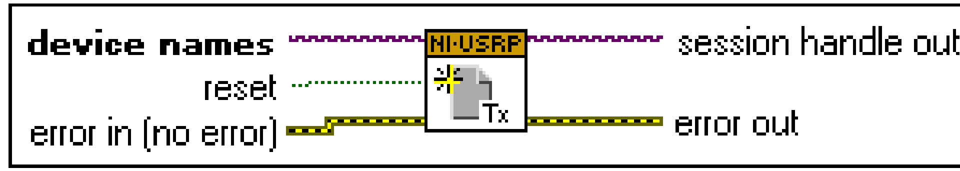
- Device names – this configures the LabView interface to talk with the radio.
- IQ Rate - Specifies the sample rate of the baseband I/Q data for Tx or Rx in samples per second (Samples/second).
- Carrier frequency – The passband frequency to be used by the radios for modulation
- Gain – Amplification of the transmitted signal.
- Active antenna – Should always be set to TX1 (the USRP transreceiver)





USRP Transmitter (Open Tx Session)

This sub-VI initiates the transmitter session and generates a session handle and an error cluster that are propagated through all VIs.



device names specifies the name(s) or IP address(es) of the device(s).

reset specifies whether to reset the device(s) to a known initialization state.

> **Note** This parameter has no effect in NI-USRP. All properties are set to their default values when a session is created.

error in describes error conditions that occur before this node runs. This input provides standard error in functionality.

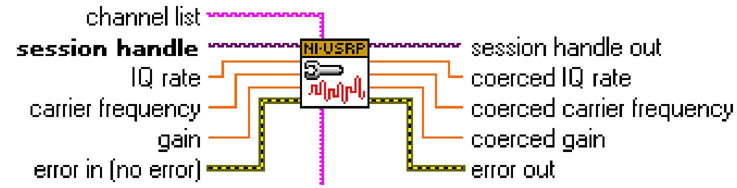
session handle out passes a reference to your instrument session to the next VI.

session handle out is obtained from this VI and identifies this Tx session.

error out contains error information. This output provides standard error out functionality.

Ibero-American Science & Technology Education Consortium
Global Innovation Network for Entrepreneurship and Technology

USRP Transmitter (Configure Signal)



session handle identifies your instrument session.

session handle is obtained from the niUSRP Open Tx Session VI or the niUSRP Open Rx Session VI and identifies a particular Tx or Rx session.

channel list specifies the channel(s) to configure.

Refer to Using Properties for more information about using the channel list parameter.

IQ rate specifies the rate of the baseband I/Q data in samples per second (S/s).

carrier frequency specifies the carrier frequency, in Hz, of the RF signal.

active antenna specifies the antenna port to use for this channel.

Refer to NI USRP-2920, NI USRP-2921, or NI USRP-2922 for a list of antenna names that this parameter accepts.

gain specifies the aggregate gain, in dB, applied to the RF signal.

error in describes error conditions that occur before this node runs. This input provides standard error in functionality.

session handle out passes a reference to your instrument session to the next VI.

session handle out is obtained from the niUSRP Open Tx Session VI or the niUSRP Open Rx Session VI and identifies a particular Tx or Rx session.

coerced IQ rate returns the actual I/Q rate, in samples per second (S/s), for this session, coerced to a value supported by the device.

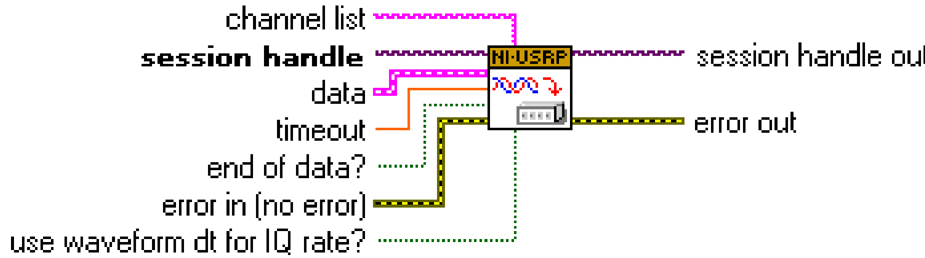
coerced carrier frequency returns the actual carrier frequency, in Hz, for this session, coerced to a value supported by the device.

coerced gain returns the actual gain, in dB, for this session, coerced to a value supported by the device.

error out contains error information. This output provides standard error out functionality.

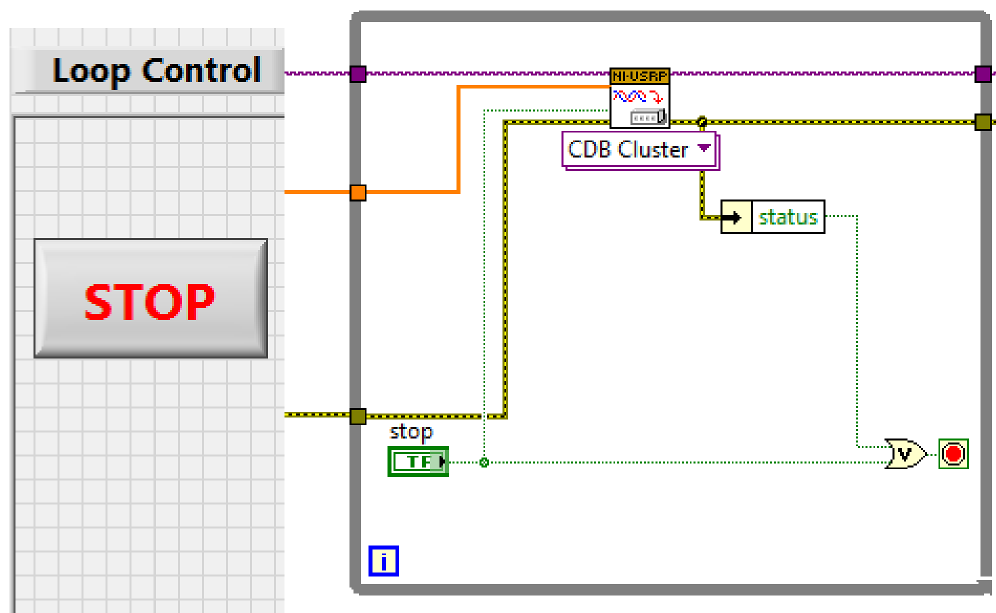
USRP Transmitter (Write to USRP TX Buffer)

The signal to be transmitted will consist of an array of data, sampling period, and an initial time for the time vector.



In some of the labs, you will generate this array and repeatedly send the same signal. In this case, your application will be inserted outside the loop.

In others, the signal will change dynamically with the controls on the front panel. In this situation, your application will be inside the loop.



All templates will come with a stop button on the front panel. Use this to stop execution of your application – it will ensure the radio shuts down properly.

USRP Transmitter (Writing to Transmit Buffer)

I/O **session handle** identifies your instrument session.

session handle is obtained from the niUSRP Open Tx Session VI and identifies a particular Tx session.

data specifies the baseband samples to transmit as complex, double-precision floating-point data in a cluster, which also includes sampling information.

data accepts complex, double-precision floating-point values whose real and imaginary components range from 1.0 to −1.0. The maximum complex magnitude is 1.0. Use the following equation to determine the complex magnitude of the signal:

$$\textit{complex magnitude} = \sqrt{Real^2 + Imaginary^2}$$

- DBL **t0** NI-USRP ignores this value.
- DBL **dt** specifies the time between values in the **Y** array.
- CDB **Y** specifies the complex-valued baseband waveform. The real and imaginary parts of this complex data array correspond to the in-phase (I) and quadrature-phase (Q) data, respectively.

DBL **timeout** specifies the time to wait, in seconds, before returning an error if the requested number of samples have not been generated.

A negative value indicates to the driver to wait indefinitely.

TF **end of data?** specifies whether this is the last call to the niUSRP Write Tx Data VI for the current contiguous transmit operation. The default value is **FALSE**.

TRUE	Specifies that the **data** input contains the end of the data transmission. The transmission aborts when the last data sample generates.
FALSE	Specifies that you will provide more data.

abc **channel list** specifies the channel(s) to which to write the data.

Refer to Using Properties for more information about using the channel list parameter.

TF **use waveform dt for IQ rate?** specifies whether the **dt** subparameter of the **data** waveform overrides the I/Q rate. The default value is **FALSE**.

TRUE	Specifies that the waveform **dt** overrides the I/Q rate.
FALSE	Specifies that the waveform **dt** does not override the I/Q rate.

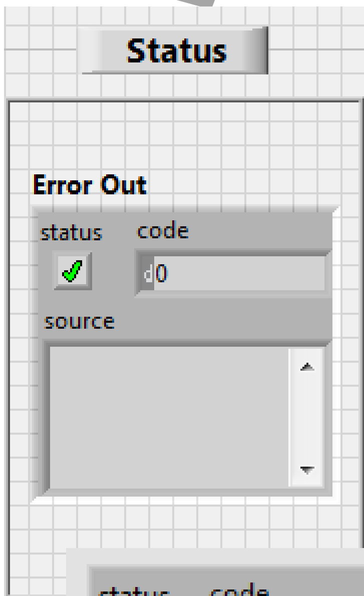
error in describes error conditions that occur before this node runs. This input provides standard error in functionality.

I/O **session handle out** passes a reference to your instrument session to the next VI.

session handle out is obtained from the niUSRP Open Tx Session VI and identifies a particular Tx session.

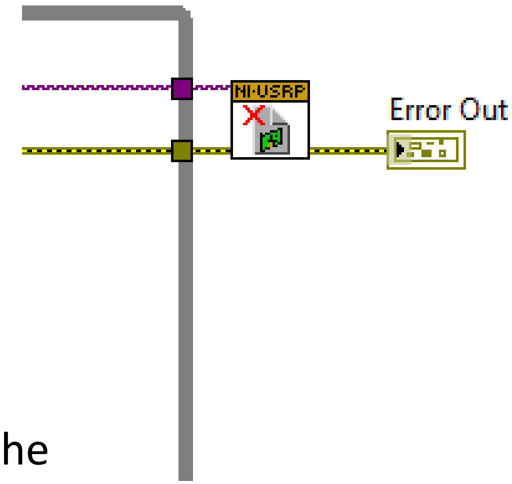
error out contains error information. This output provides standard error out functionality.

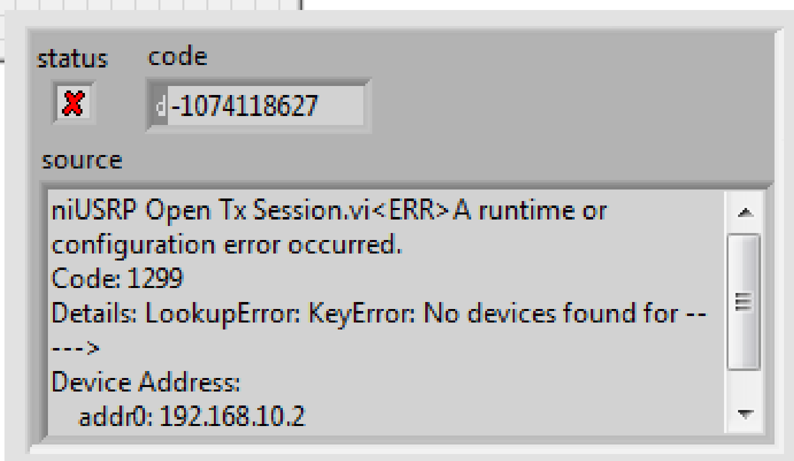
USRP Transmitter (Transmitter Status)



Each transmitter template has a status window.

If there are no errors in the transmission of the data, you should have a status display with a green check mark.





If there is an error in the transmission of the data, you should have a status display with a red x mark with an error code and an error message.

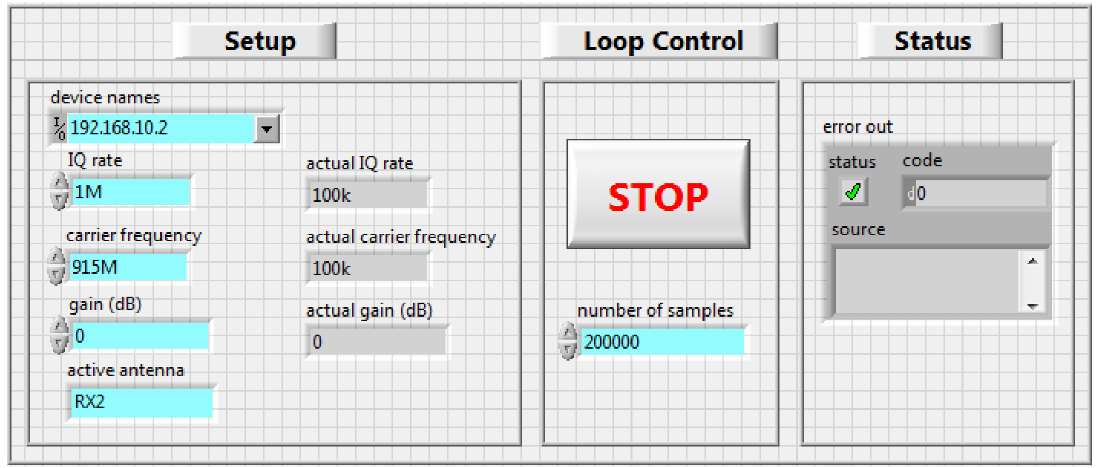
In this case, the message indicates you are not connected to the radio through the ethernet interface.

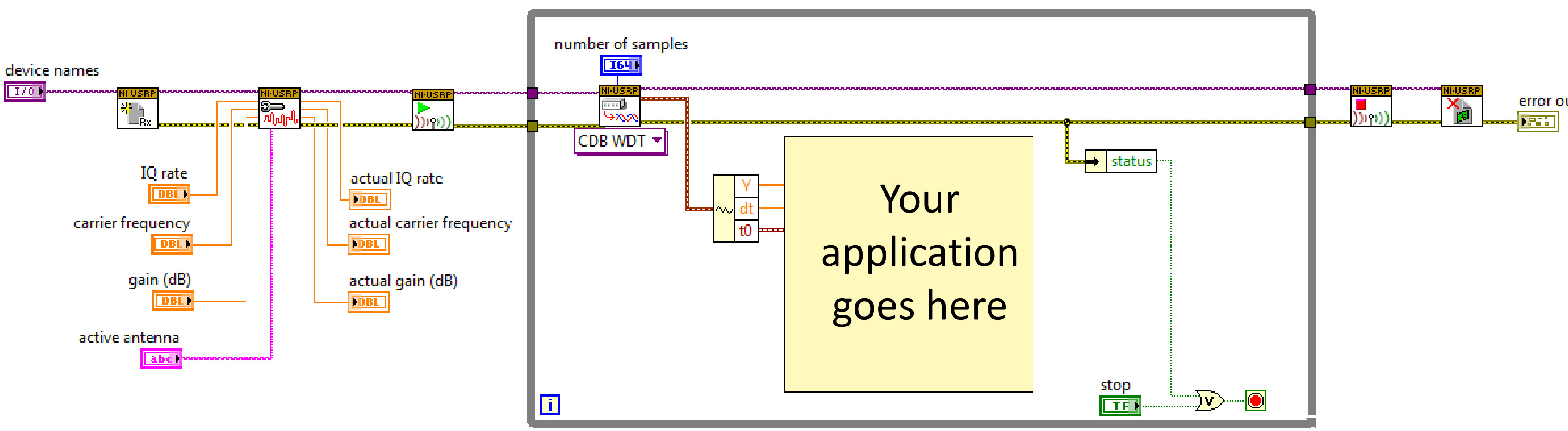
THE UNIVERSITY of NEW MEXICO

USRP Receiver (Receiver Template)

The Receiver template consists of 4 elements:

- USRP Receiver Configuration
- While-loop to control execution of lab.
- Read from the receiver buffer
- USRP shutdown & status reporting





istec Ibero-American Science & Technology Education Consortium

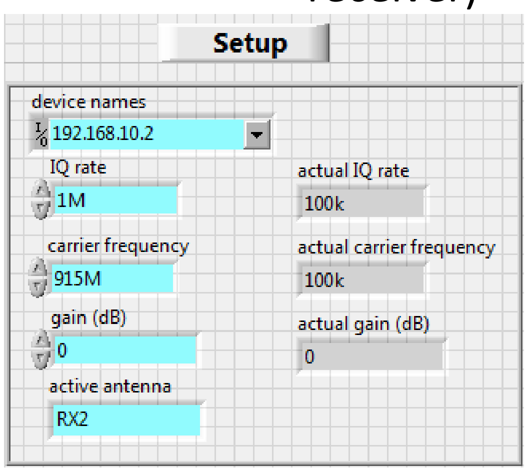
GINET Global Innovation Network for Entrepreneurship and Technology

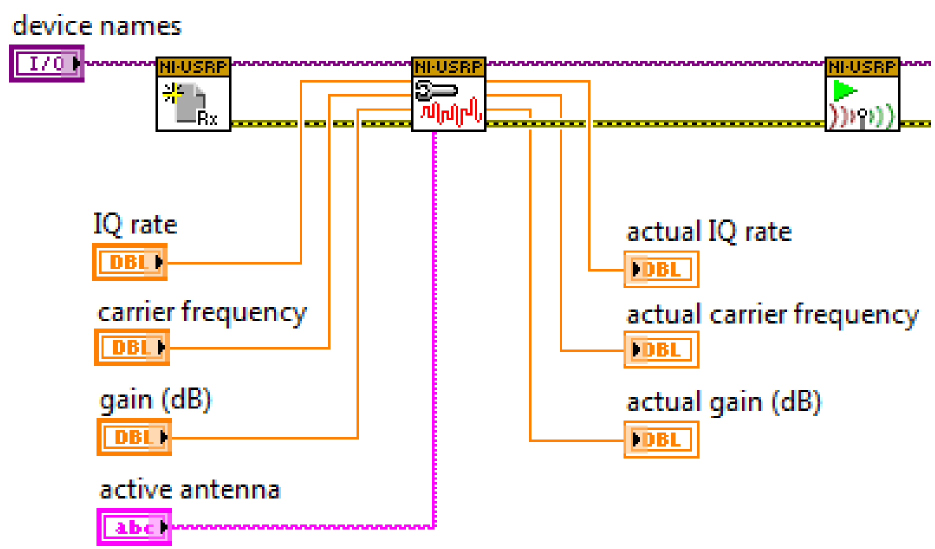
USRP Receiver (Configuration)

THE UNIVERSITY of NEW MEXICO

The front panel for each application will have an USRP configuration panel. The panel supports entering the following radio parameters:

- Device names – this configures the LabView interface to talk with the radio.
- IQ Rate - Specifies the sample rate of the baseband I/Q data for Tx or Rx in samples per second (Samples/second).
- Carrier frequency – The passband frequency to be used by the radios for modulation
- Gain – Amplification of the received signal.
- Active antenna – Should be set to RX1 or RX2 (the USRP transceiver or secondary receiver)





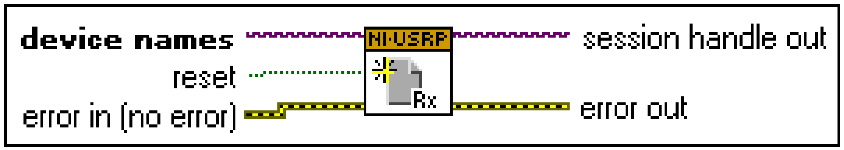
istec
Ibero-American Science & Technology Education Consortium

Printed on 9/1/2014

GINET
Global Innovation Network for Entrepreneurship and Technology

USRP Reciever (Open Rx Session)

This sub-VI initiates the receiver session and generates a session handle and an error cluster that are propagated through all VIs.



device names specifies the name(s) or IP address(es) of the device(s).

reset specifies whether to reset the device(s) to a known initialization state.

Note This parameter has no effect in NI-USRP. All properties are set to their default values when a session is created.

error in describes error conditions that occur before this node runs. This input provides standard error in functionality.

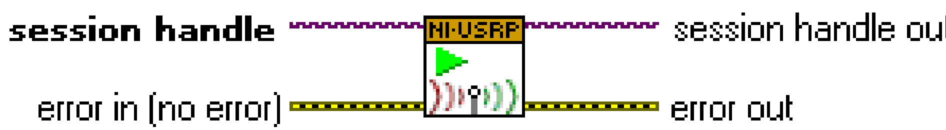
session handle out passes a reference to your instrument session to the next VI.

session handle out is obtained from this VI and identifies this Rx session.

error out contains error information. This output provides standard error out functionality.

USRP Receiver (Initiate Reception of Data)

The niUSRP Initiate VI starts the waveform acquisition in a Rx session. You must initiate the Rx session before you use a Fetch Rx Data (poly) VI to retrieve waveform data. You do not need to call the niUSRP Initiate VI for Tx sessions; you initiate waveform generation when you provide data using the Write Tx Data (poly) VI.



session handle identifies your instrument session.

session handle is obtained from the niUSRP Open Rx Session VI and identifies a particular Rx session.

error in describes error conditions that occur before this node runs. This input provides standard error in functionality.

session handle out passes a reference to your instrument session to the next VI.

session handle out is obtained from the niUSRP Open Rx Session VI and identifies a particular Rx session.

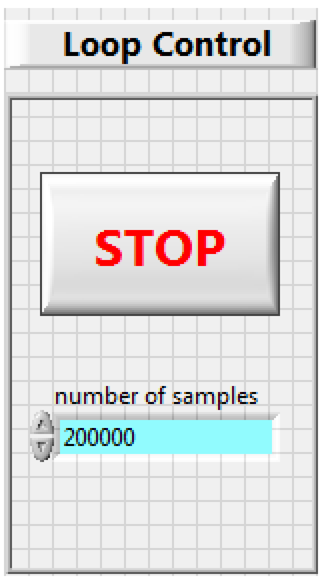
error out contains error information. This output provides standard error out functionality.

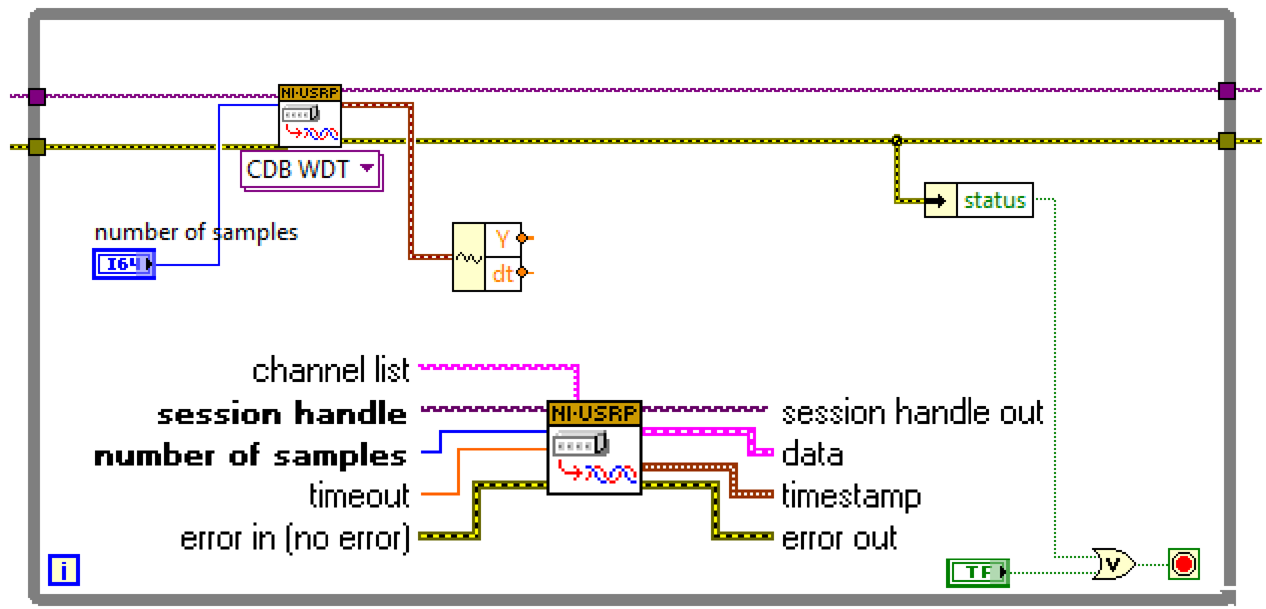
THE UNIVERSITY of NEW MEXICO

USRP Receiver (Read From USRP Buffer)

The signal of received data will consist of an array of data, sampling period, and an initial time for the time vector.

All templates will come with a stop button on the front panel. Use this to stop execution of your application – it will ensure the radio shuts down properly.





istec
Ibero-American Science & Technology Education Consortium

GINET
Global Innovation Network for Entrepreneurship and Technology

USRP Receiver (Reading From Receive Buffer)

I/O **session handle** identifies your instrument session.

session handle is obtained from the niUSRP Open Rx Session VI and identifies a particular Rx session.

I64 **number of samples** specifies the number of samples to fetch from the acquisition channel.

DBL **timeout** specifies the time to wait, in seconds, before returning an error if the requested number of samples have not been acquired.

A negative value indicates to the driver to wait indefinitely.

abc **channel list** specifies the channel(s) from which to fetch the data.

Refer to Using Properties for more information about using the channel list parameter.

error in describes error conditions that occur before this node runs. This input provides standard error in functionality.

I/O **session handle out** passes a reference to your instrument session to the next VI.

session handle out is obtained from the niUSRP Open Rx Session VI and identifies a particular Rx session.

data returns the received baseband samples as complex, double-precision floating-point data in a cluster, which also includes sampling information.

- DBL **t0** specifies the trigger (start) time of the acquired **Y** array.
- DBL **dt** specifies the time between values in the **Y** array.
- CDB **Y** specifies the complex-valued baseband waveform. The real and imaginary parts of this complex data array correspond to the in-phase (I) and quadrature-phase (Q) data, respectively.

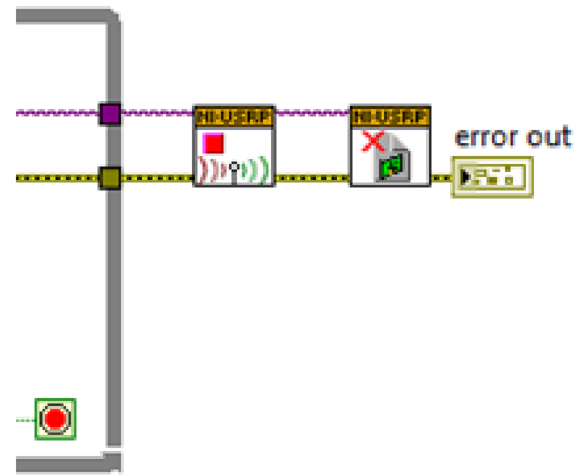
timestamp returns the timestamp of the first Rx sample returned and indicates the time associated with the first sample of the waveform, according to the onboard device timer.

timestamp is the time of the clock in seconds, interpreted as **whole seconds.fractional seconds**.

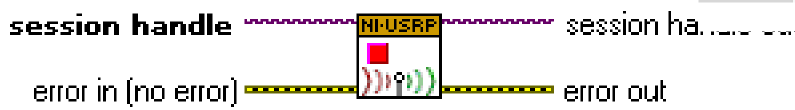
- U64 **whole seconds** is the integer number of seconds for the time associated with the first sample of the waveform, according to the onboard device timer.
- DBL **fractional seconds** is the double-precision, floating-point value representing the remaining fraction of a second for the time associated with the first sample of the waveform, according to the onboard device timer.

error out contains error information. This output provides standard error out functionality.

USRP Receiver (Receiver Shutdown & Status)



Stops an acquisition previously started.
For finite acquisitions, calling this VI is optional unless you want to stop the acquisition before it is complete. If the acquisition aborts successfully, the driver transitions to the Done state.



session handle identifies your instrument session.

session handle is obtained from the niUSRP Open Rx Session VI and identifies a particular Rx session.

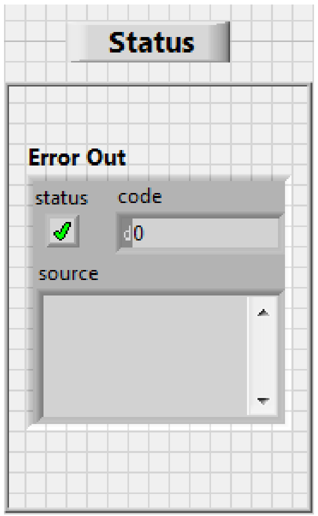
error in describes error conditions that occur before this node runs. This input provides standard error in functionality.

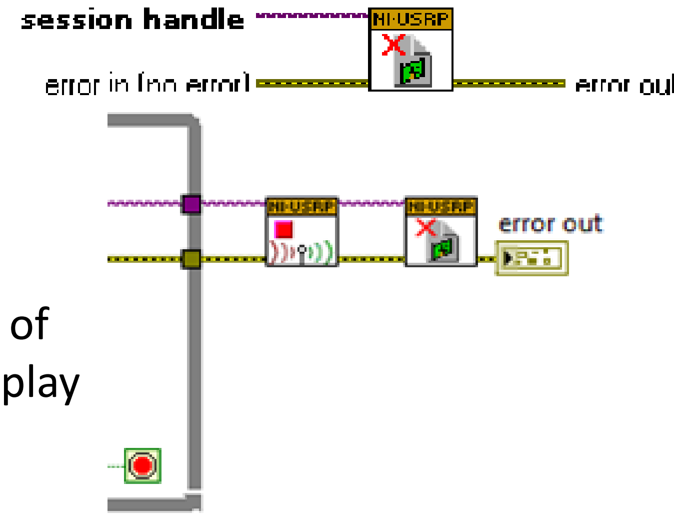
session handle out passes a reference to your instrument session to the next VI.

session handle out is obtained from the niUSRP Open Rx Session VI and identifies a particular Rx session.

error out contains error information. This output provides standard error out functionality.

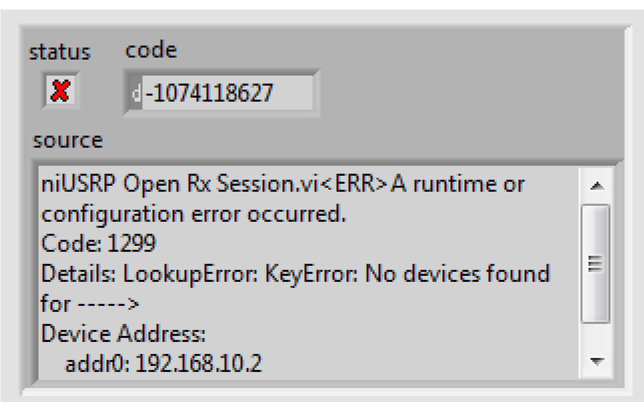
USRP Receiver (Receiver Status)





Each Receiver template has a status window.

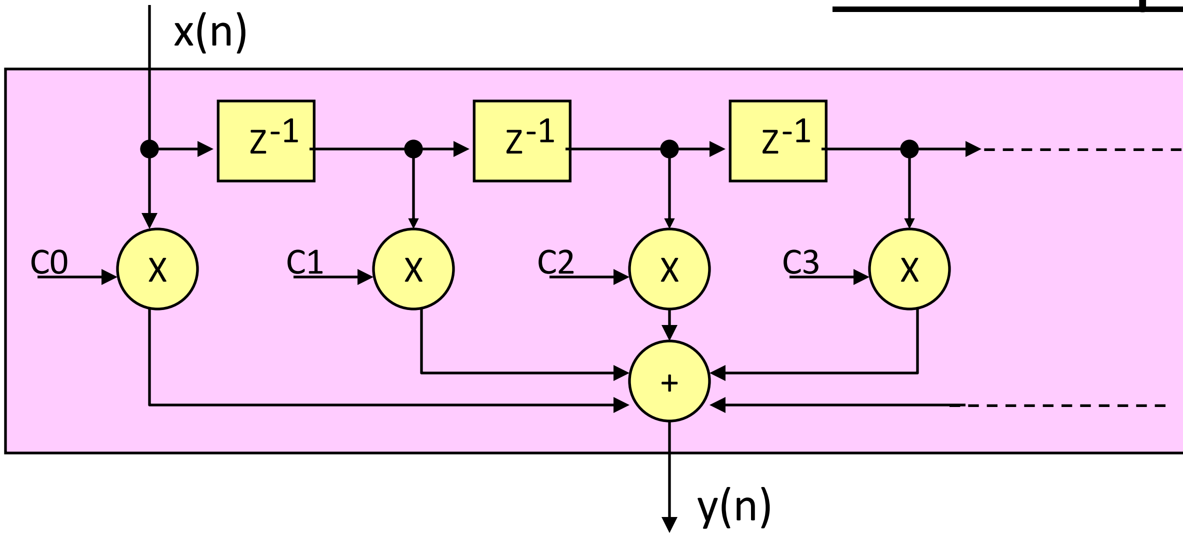
If there are no errors in the reception of the data, you should have a status display with a green check mark.



If there is an error in the reception of the data, you should have a status display with a red x mark with an error code and an error message.

In this case, the message indicates you are not connected to the radio through the ethernet interface.

FIR digital filters: *Finite* Impulse Response



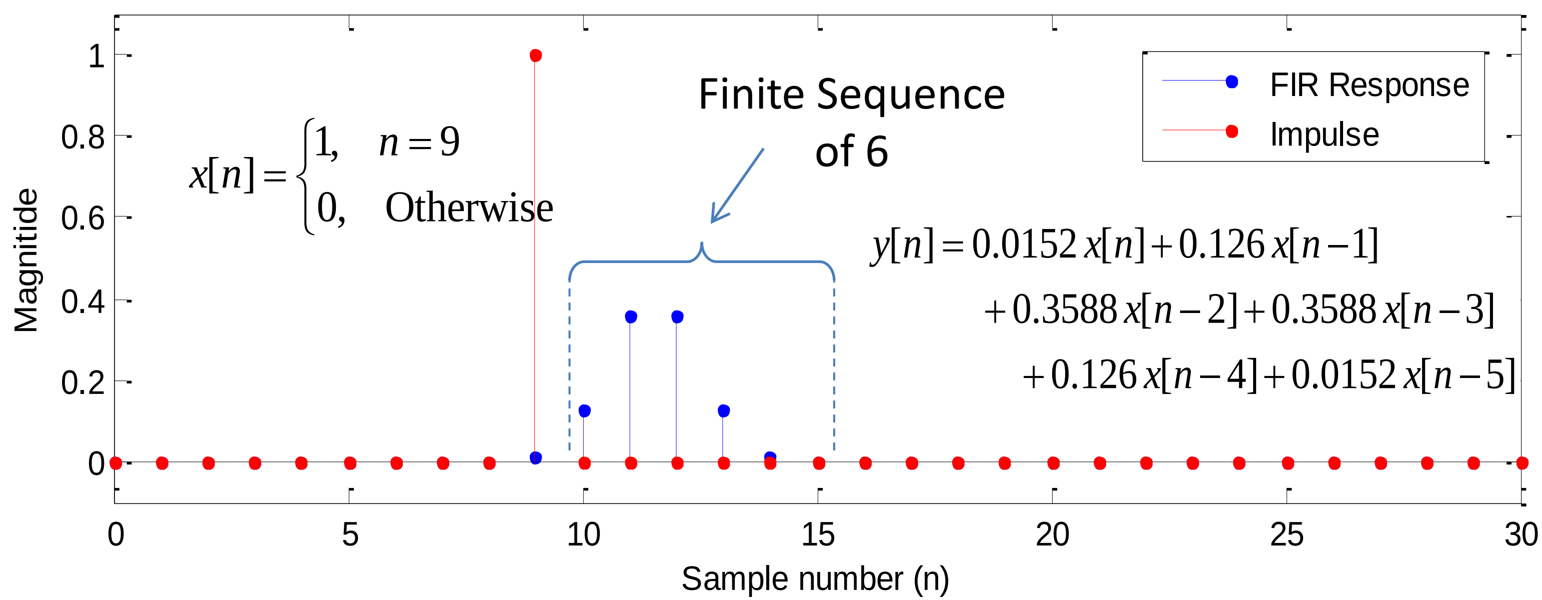
$$y(n) = \sum_{i=0}^{N} C[i]\, x(n-i)$$

- FIR filters: stands for *Finite Impulse Response*, is the simplest type of digital filter, it is inherently estable, and always realizable.
- The C(k) coefficients of an FIR are actually the sampled values of the filter's impulse response.
- Given an FIR with "n" taps, the effect of an input vanishes in the output after "n" delays. Thus its *finite response.*
- Can be designed to have a linear phase response
- Usually non recursive

Recursive FIR example? Think of an average!!

FIR digital filters:
FIR Example

The output signal *y[n]* of the filter in response to an impulse is limited only the last *N* values of *x[n]*, so after *N+1* samples the response returns to zero. For example, the response of a fifth order filter consists of a finite sequence of six (*N+1*) samples



Digital filters: FIR filters

- An FIR filter performs the *convolution* between the filter's impulse response (C[..] coefficients) and the samples of the input signal (x[..]), thus, the coefficients C[..] of an FIR are the sampled values of the filter's impulse response

$$y(n) = \sum_{i=0}^{N} C[i]\, x(n-i)$$

Time response

- The filter's transfer functions, in Z, is:

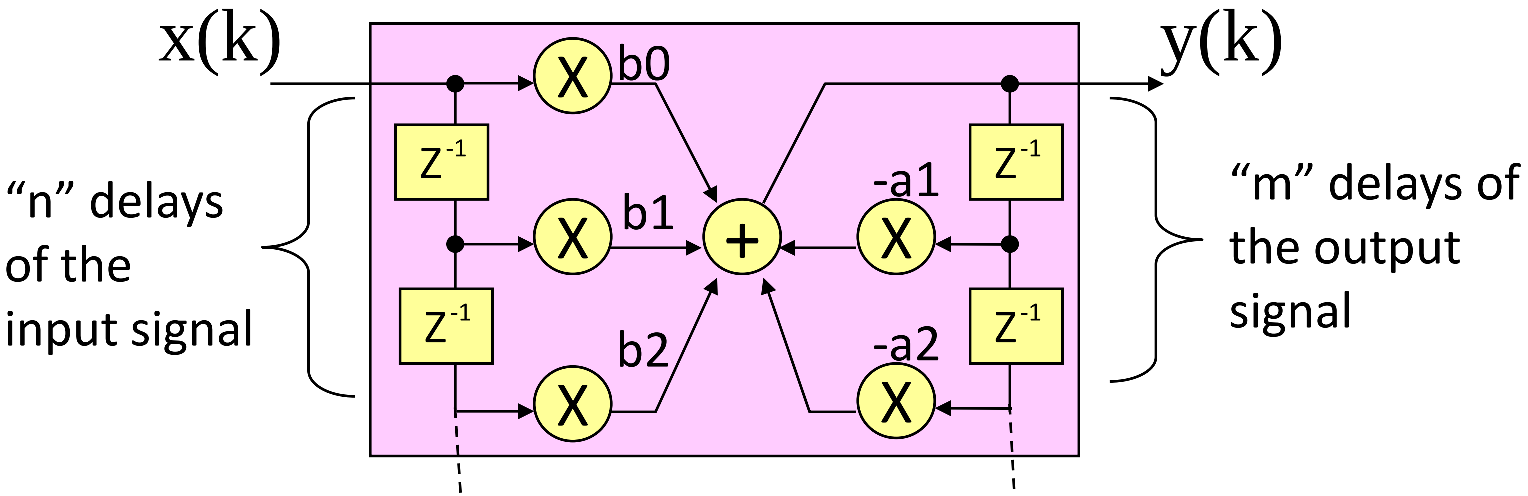
$$H(z) = \frac{Y(z)}{X(z)} = \sum_{i=0}^{N} h[i]\, z^{-i}$$

H(z): filter's transfer function

- This is a polynomial equation of order N, and the N roots of this polynomial are the N zeros of the filter

IIR digital Filters : *Infinite* Impulse Response

$$y(k) = \sum_{i=0}^{N} b_i \; x(k-i) + \sum_{j=1}^{M} a_j \; y(k-j)$$



- The IIR is a more complex type of filter, where the output feedback enables its response to extend infinitely in time
- They are usually more efficient (requiring less storage, lower complexity, lower cost) than the FIR, although with more problems, namely stabilty and numerical error propagation
- They can be desgined starting from analogies with existing analog filters

Digital filters: IIR Filters

- The IIR (*Infinite Impulse Response*) filters are a more complex type of filter, with an output at time k, given by:

$$O(k) = \sum_{i=0}^{N} b_i . I(k-i) + \sum_{j=1}^{M} a_j . O(k-j)$$

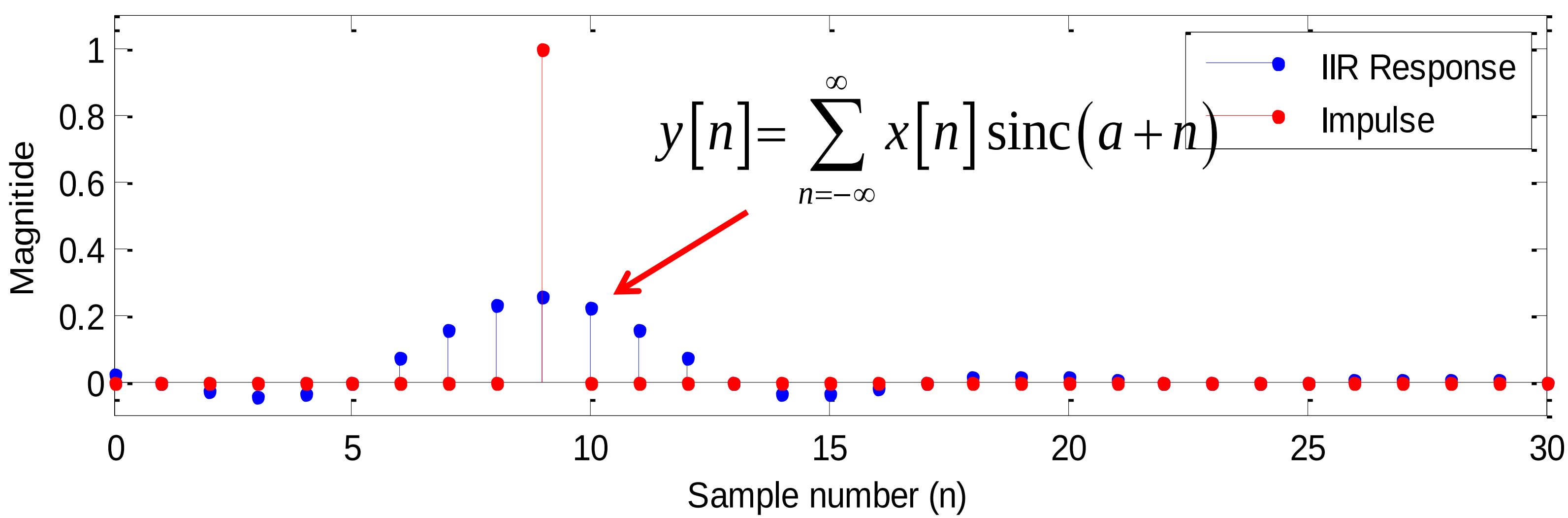
- The output is a linear combination of the current input I(k), N previous inputs, but now, also of the previous M outputs, and its corresponding transfer function is:

$$H(z) = \frac{\sum_{i=0}^{N} b_i . z^{-i}}{1 - \sum_{j=1}^{M} a_j . z^{-j}} = \frac{N(z)}{D(z)}$$

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

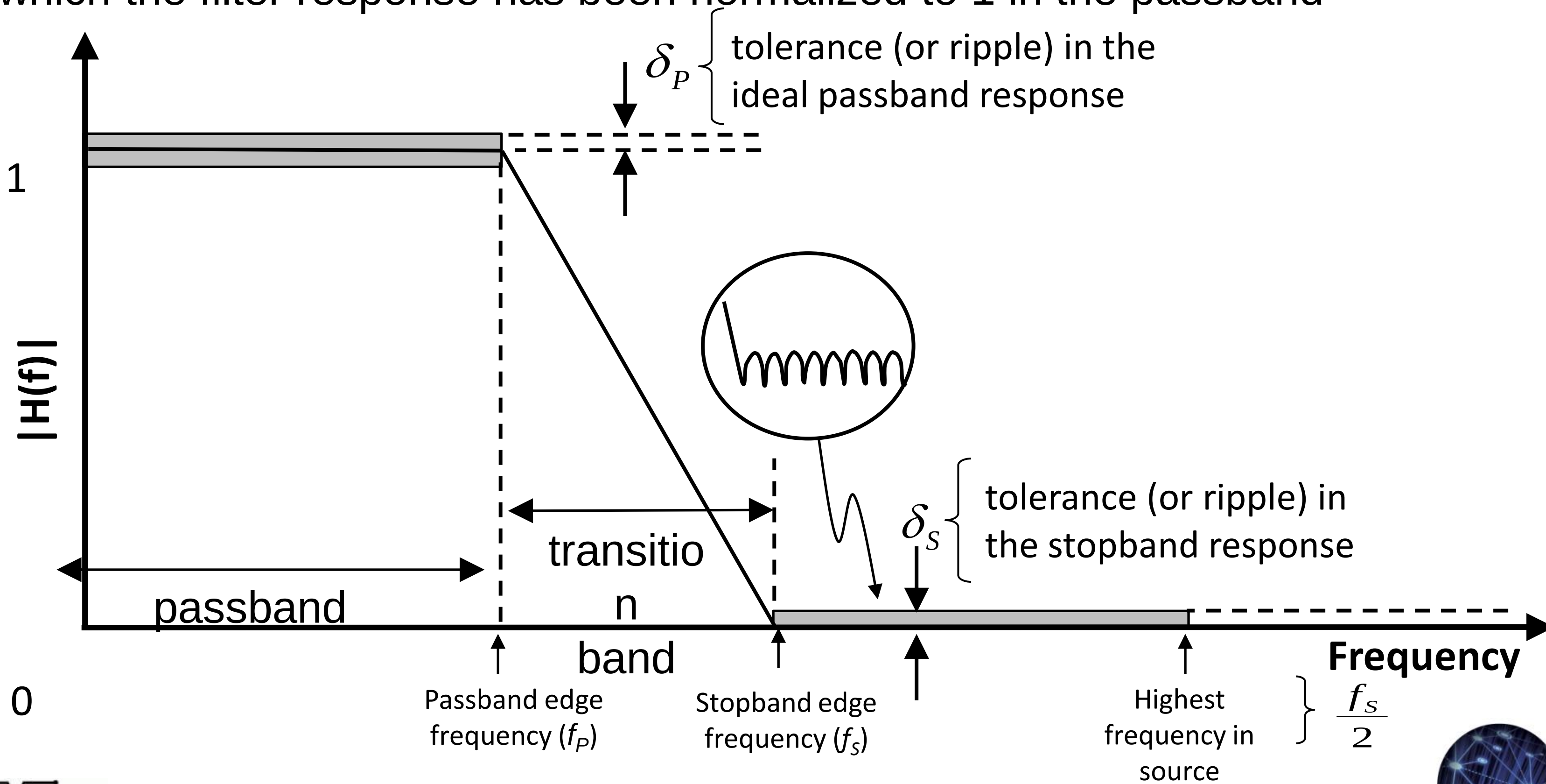
FIR digital filters: *IIR Example*

The output signal *y[n]* of the filter in response to an impulse The output signal of the filter can be non-zero infinitely, even when the input signal has a value of zero. In theory, when a recursive filter is excited by an impulse, the output will persist forever.



Preliminaries (Absolute)

A typical absolute specification of a lowpass filter is shown below, in which the filter response has been normalized to 1 in the passband

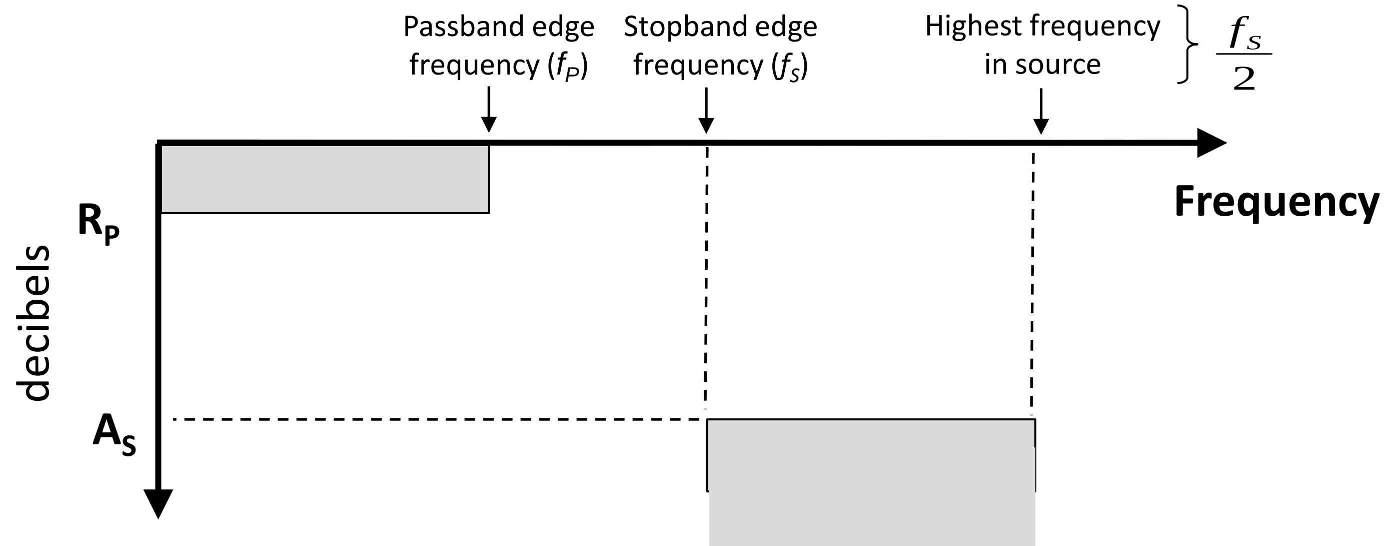


Digital Signal Processing Using MATLAB® ,Third Edition, Vinay K. Ingle John G. Proakis

Preliminaries (dB Relative Specifications)

A typical relative specification of a lowpass filter is shown below, in which

- *Rp is the passband ripple in dB, and*
- *As is the stopband attenuation in dB.*



The parameters given in these two specifications are obviously related. Since |*H(f)*| *in absolute specifications is equal to (1 +* δ_P*), we have*

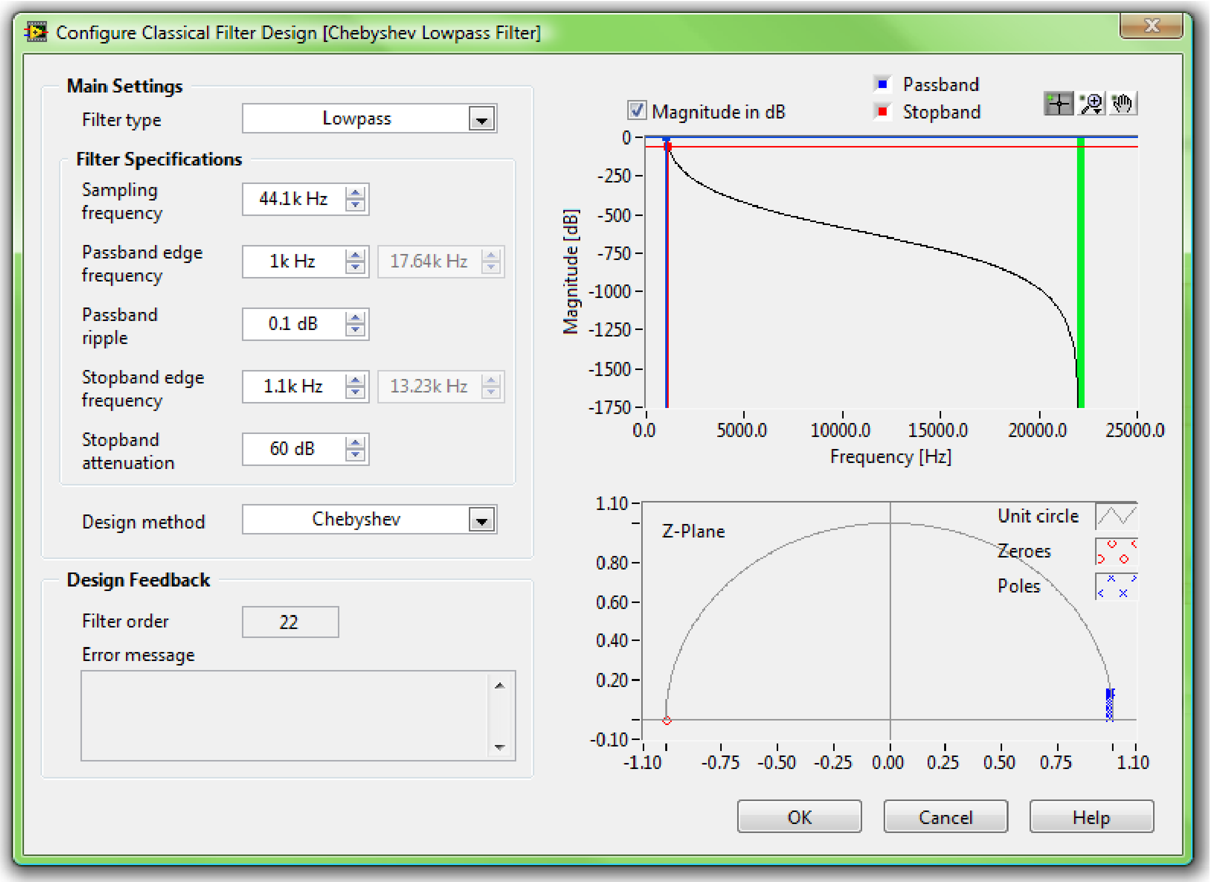
$$R_P = -20\log_{10}\left(\frac{1-\delta_P}{1+\delta_P}\right)$$

$$A_S = -20\log_{10}\left(\frac{\delta_S}{1+\delta_P}\right)$$

Digital Signal Processing Using MATLAB® ,Third Edition, Vinay K. Ingle John G. Proakis

Preliminaries (Absolute vs. Relative)

THE UNIVERSITY of NEW MEXICO



LabVIEW is looking for relative specifications

Amplitude Modulation

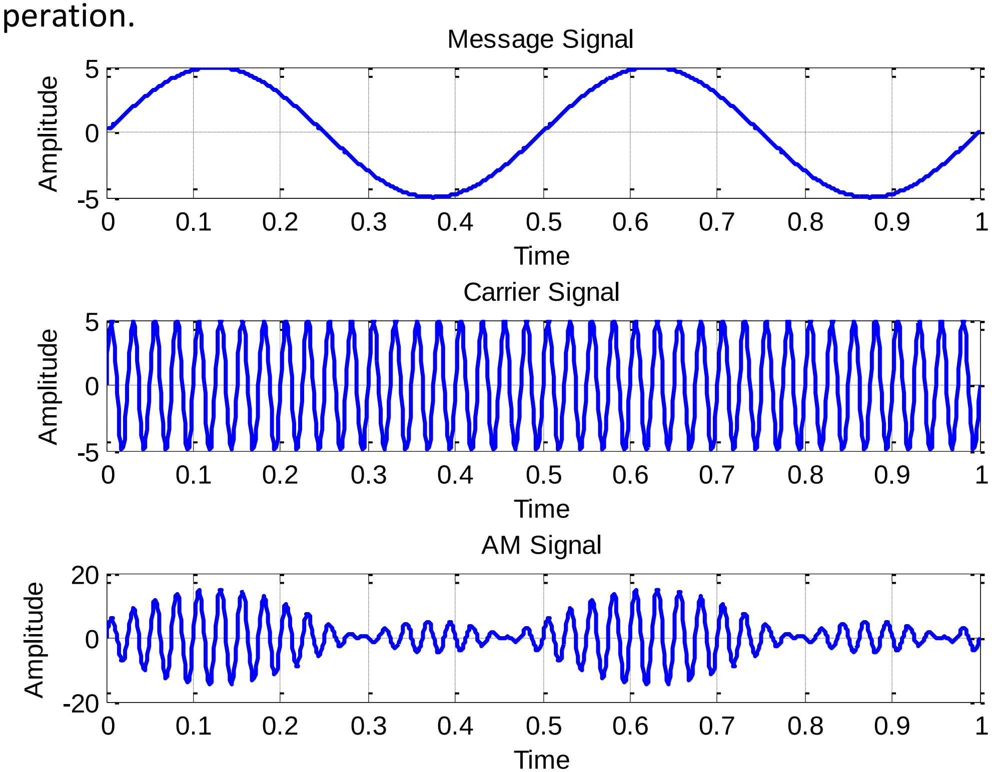
What you need to know to do the Lab…

AM Overview

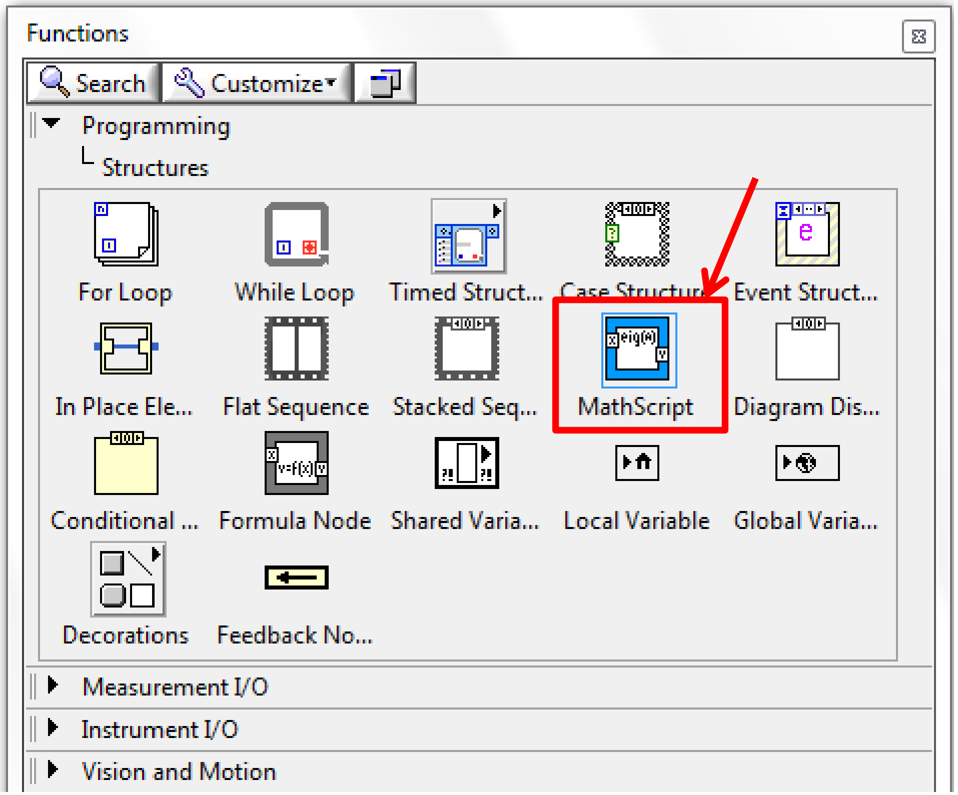
If $m(t)$ is a baseband "message" signal with a peak value m_p, and $A_c \cos(2\pi f_c t)$ is a "carrier" signal at carrier frequency, f_c, then we can write the AM signal $g(t)$ as

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

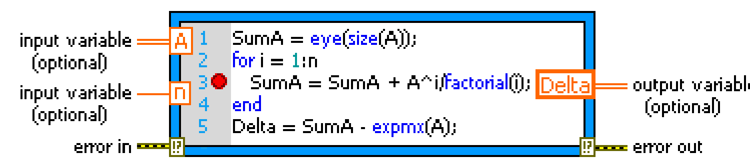
where the parameter μ is called the "modulation index" and takes values in the range $0 < \mu \leq 1$ (0 to 100%) in normal operation.



Modulation: MathScript Node



MathScript Node



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

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

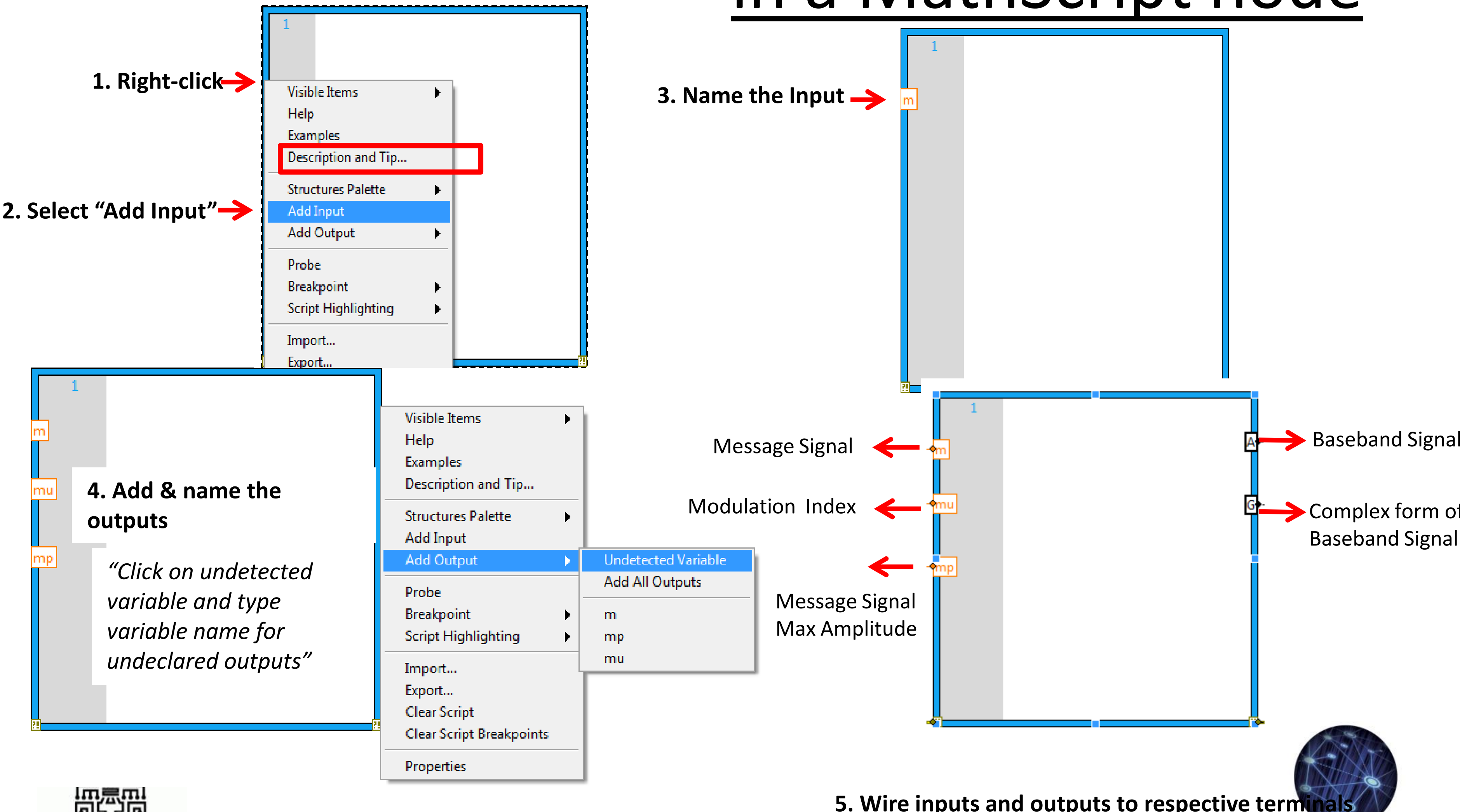
"Equations"

$$
\begin{aligned}
a &= 2b - \max(d) \\
p &= a \log(a) \\
s &= a\, e^{2\pi p j}
\end{aligned}
$$

"Text-based scripts"

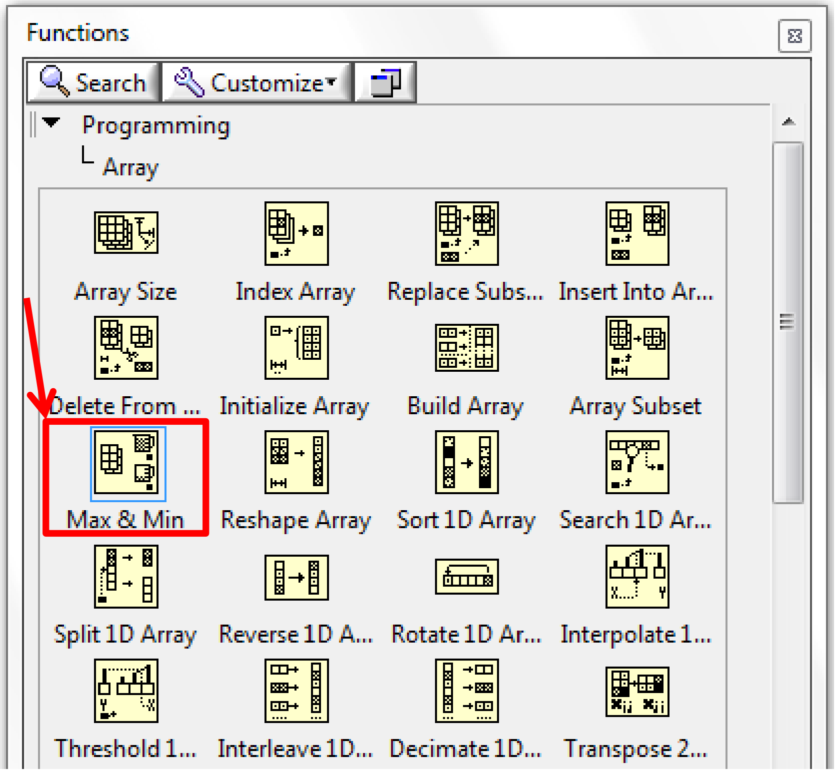
```
a = 2*b – max (d);
p = log(a)*a;
s = a*exp(2*pi*p*j);
```

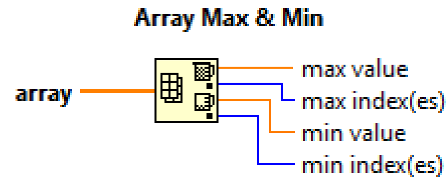
Setting up I/Ps & O/Ps in a MathScript node



THE UNIVERSITY of NEW MEXICO

Array Max & Min VI



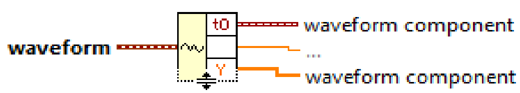


Returns the maximum and minimum values found in **array**, along with the indexes for each value.

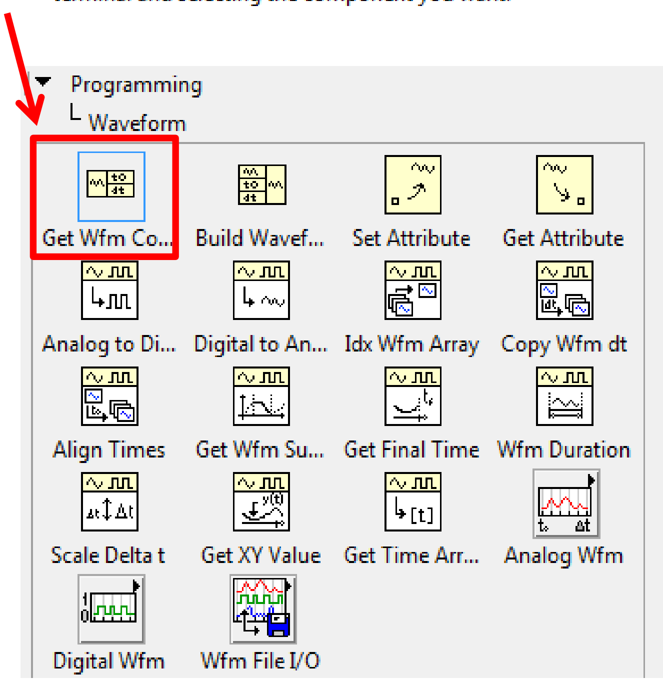
THE UNIVERSITY of NEW MEXICO

Get Waveform Components VI

Get Waveform Components

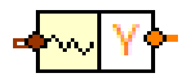


Returns the analog waveform you specify. You specify components by clicking on the center of the output terminal and selecting the component you want.

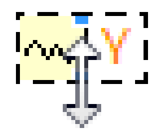


"Waveform attribute selection"

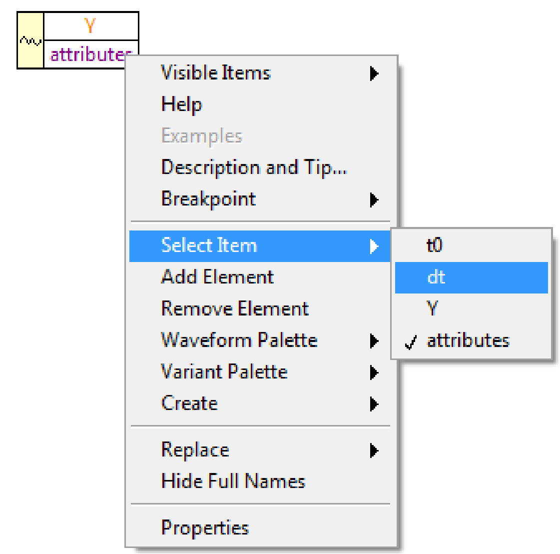
1. Select, hold and drop VI



2. Click on bottom line, hold and extend

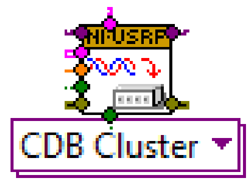


3. Right-click on attributes, scroll to "Select Item" and pick the attribute.

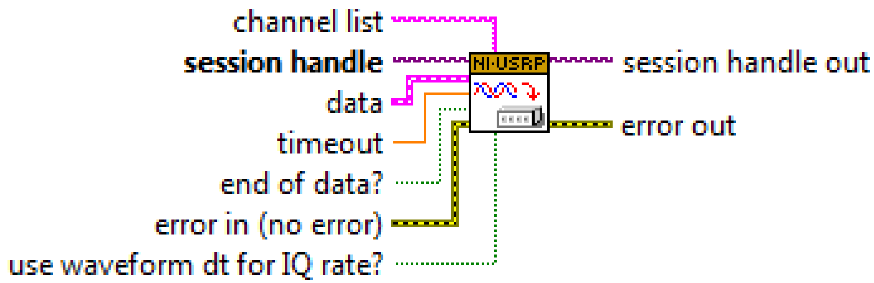


niUSRP Write Tx Data VI

"Buffer to transmit data to receiver"



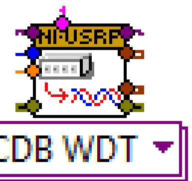
niUSRP Write Tx Data (poly).vi



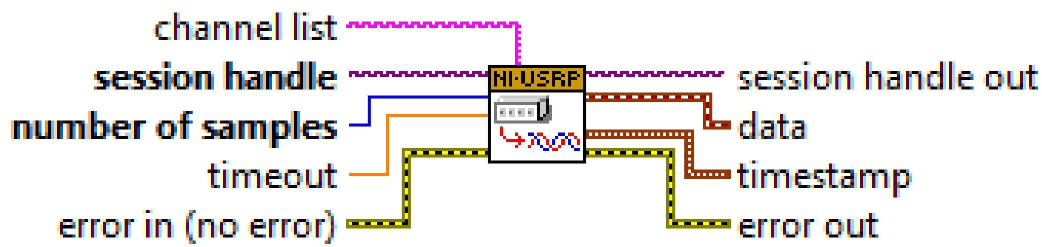
Writes data to the specified channel list.

niUSRP Fetch Rx Data VI

"Buffer to receive data from transmitter"

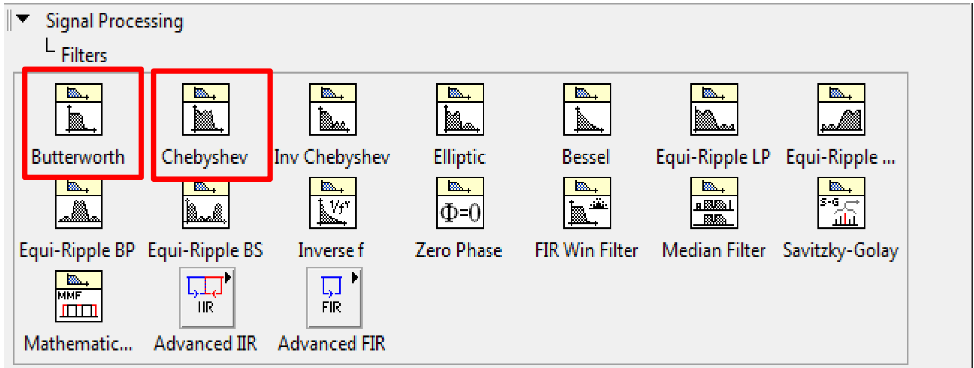


niUSRP Fetch Rx Data (poly).vi



Fetches data from the specified channel list.

Demodulation: Filters

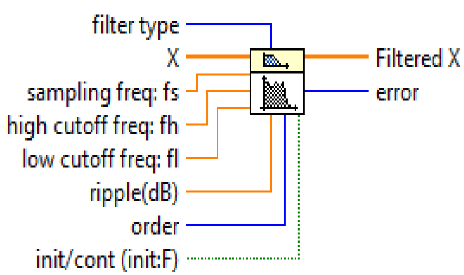


"Set filter parameters as constants"

"Chebyshev clears noise around carrier frequency"

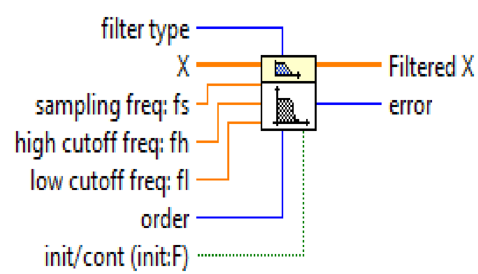
"Butterworth implemented after full wave rectification to complete envelope detection"

Chebyshev Filter.vi



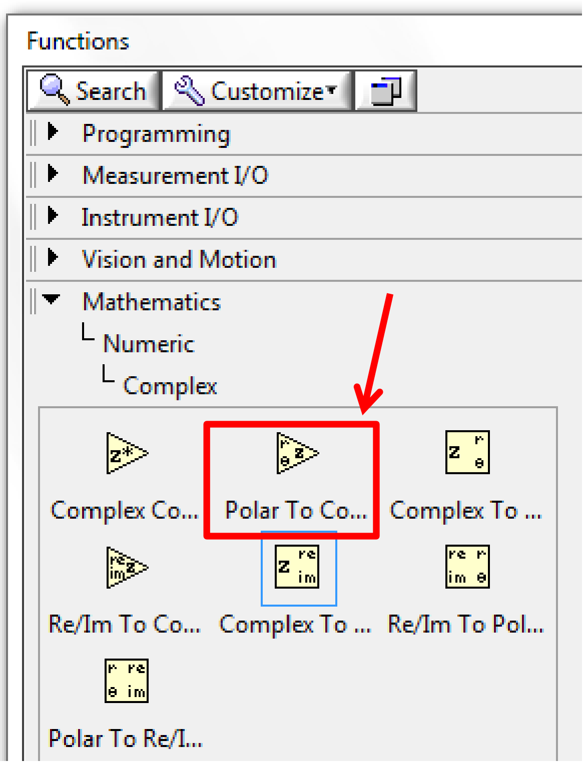
Generates a digital Chebyshev filter by calling the Chebyshev Coefficients VI. Wire data to the **X** input to determine the polymorphic instance to use or manually select the instance.

Butterworth Filter.vi

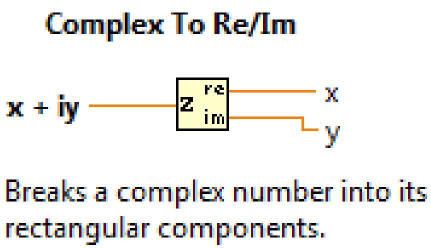


Generates a digital Butterworth filter by calling the Butterworth Coefficients VI. Wire data to the **X** input to determine the polymorphic instance to use or manually select the instance.

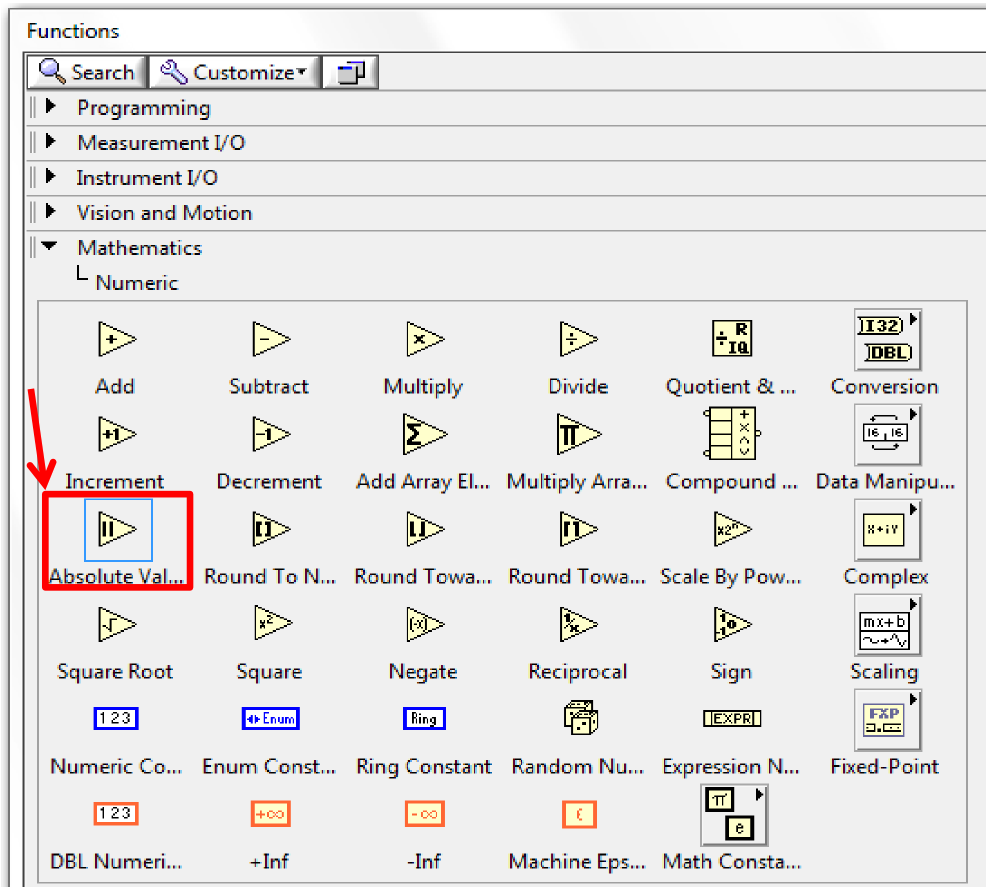
Complex to Real/Imaginary



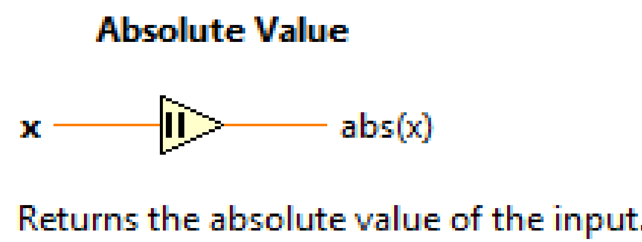
"Extract real part from complex data values"



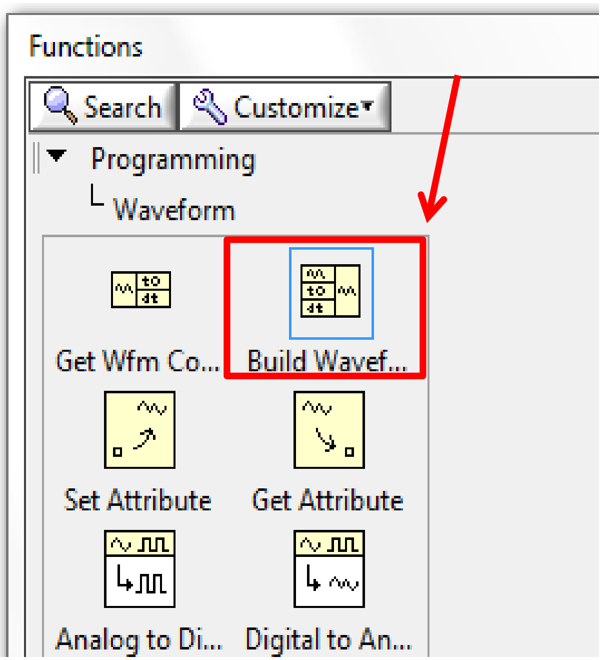
Absolute Value VI

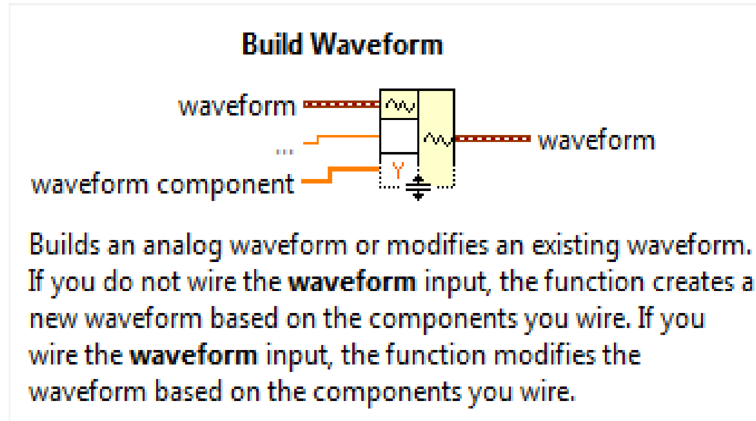


"Full-wave Rectifier"



Build Waveform VI



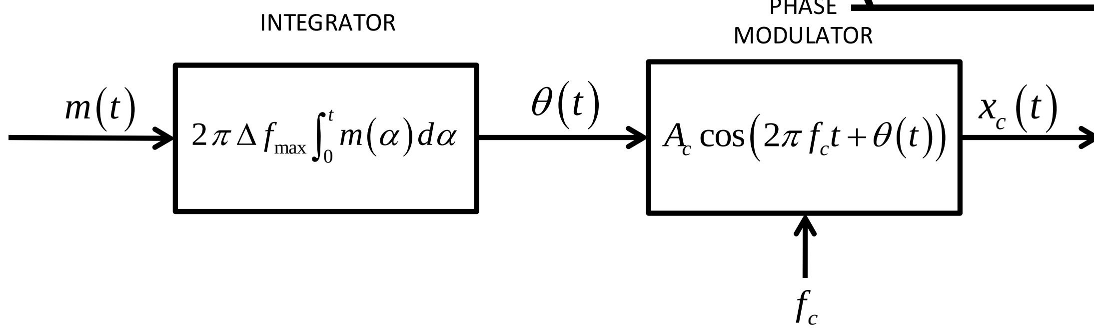


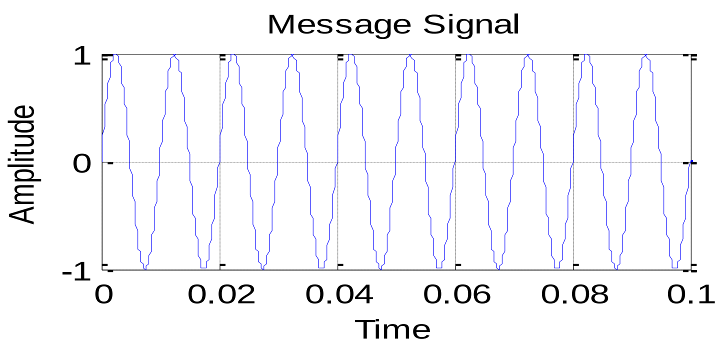
"Waveform attribute selection"
Same as "Get Waveform Components"

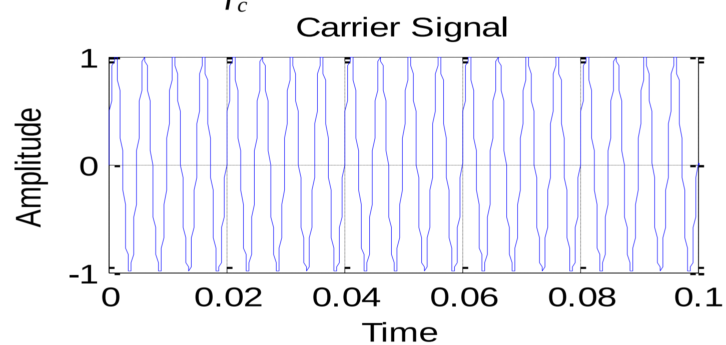
Frequency Modulation

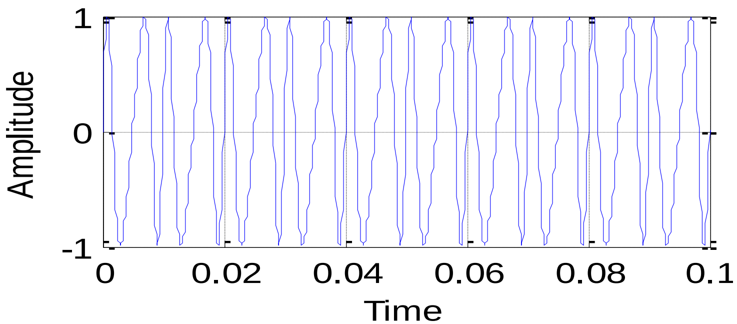
What you need to know to do the Lab…

FM Overview (Modulation)

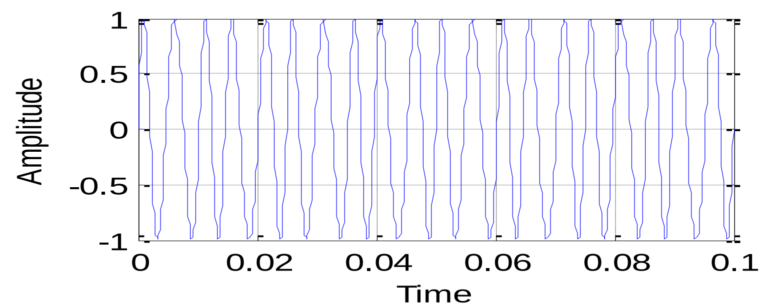








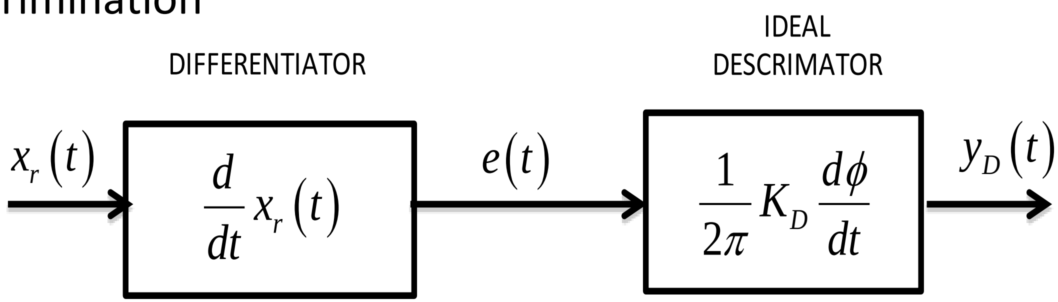
$k_f = 1$



$k_f = 0.5$

FM Overview (Demodulation)

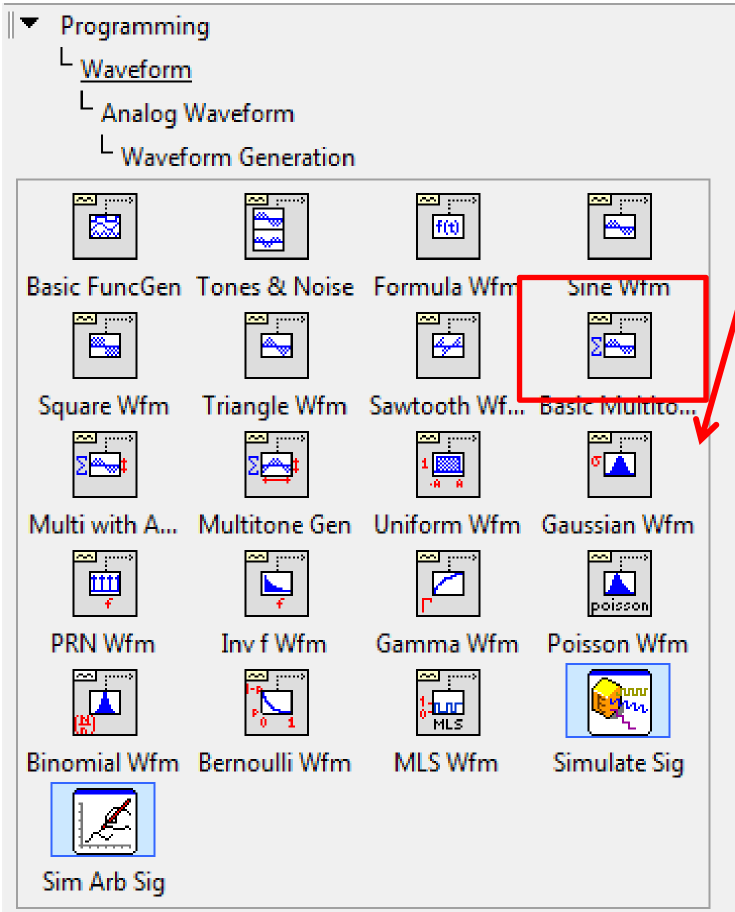
FM demodulation can be divided into three broad categories: Frequency discrimination, Phase-shift discrimination, and Phase-locked loop (PLL). This lab focuses solely on frequency discrimination



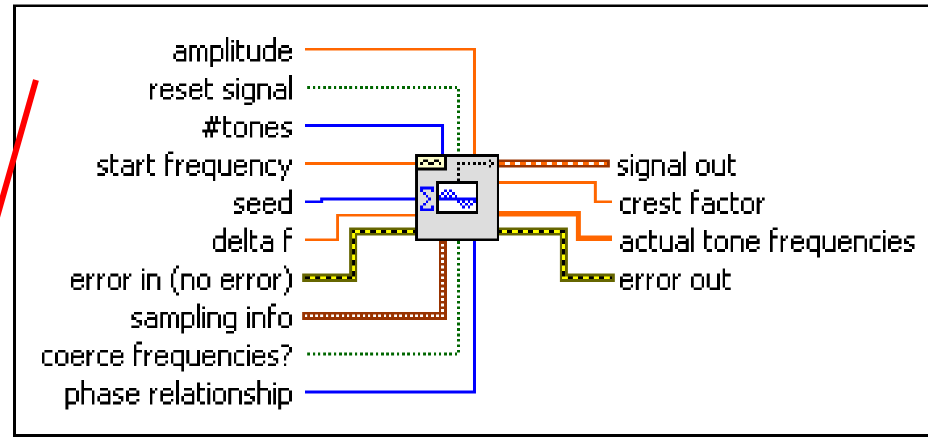
$$e(t) = -K_D\left[2\pi f_c + \frac{d\theta}{dt}\right]\sin\left(2\pi f_c t + \theta(t)\right)$$

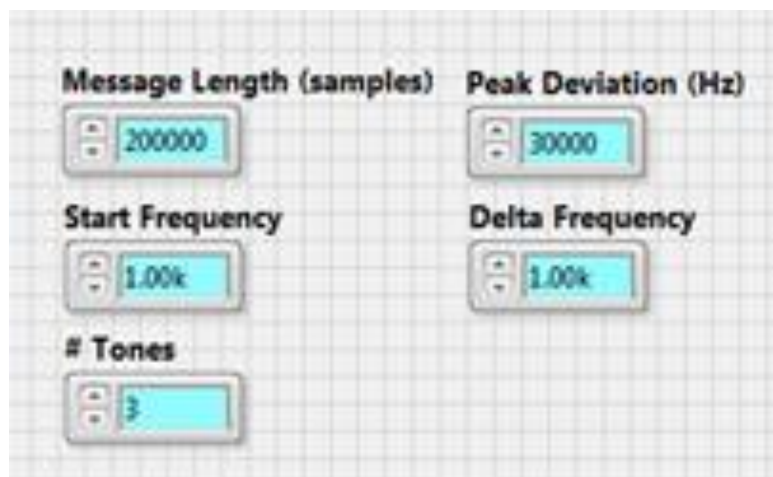
$$y_D(t) = 2\pi f_c + K_D \Delta f_{max} m_n(t)$$

Multi-Tone Message Generator

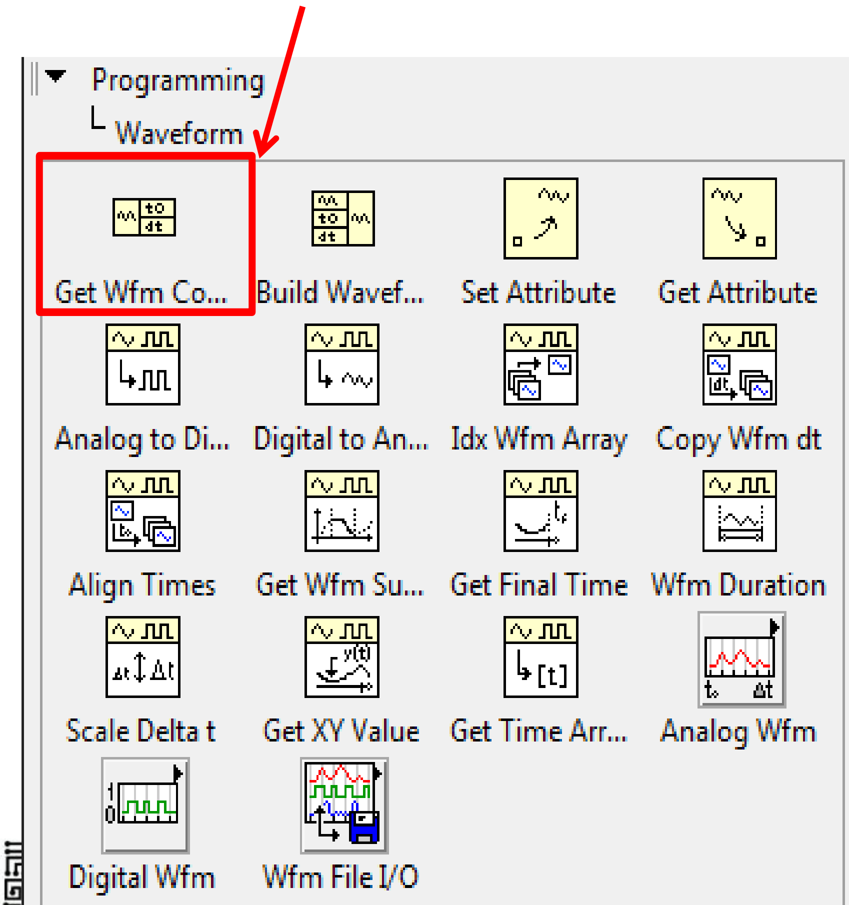


Basic Multitone VI

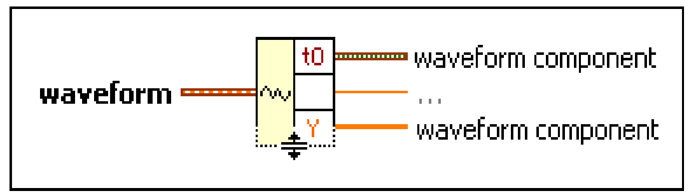




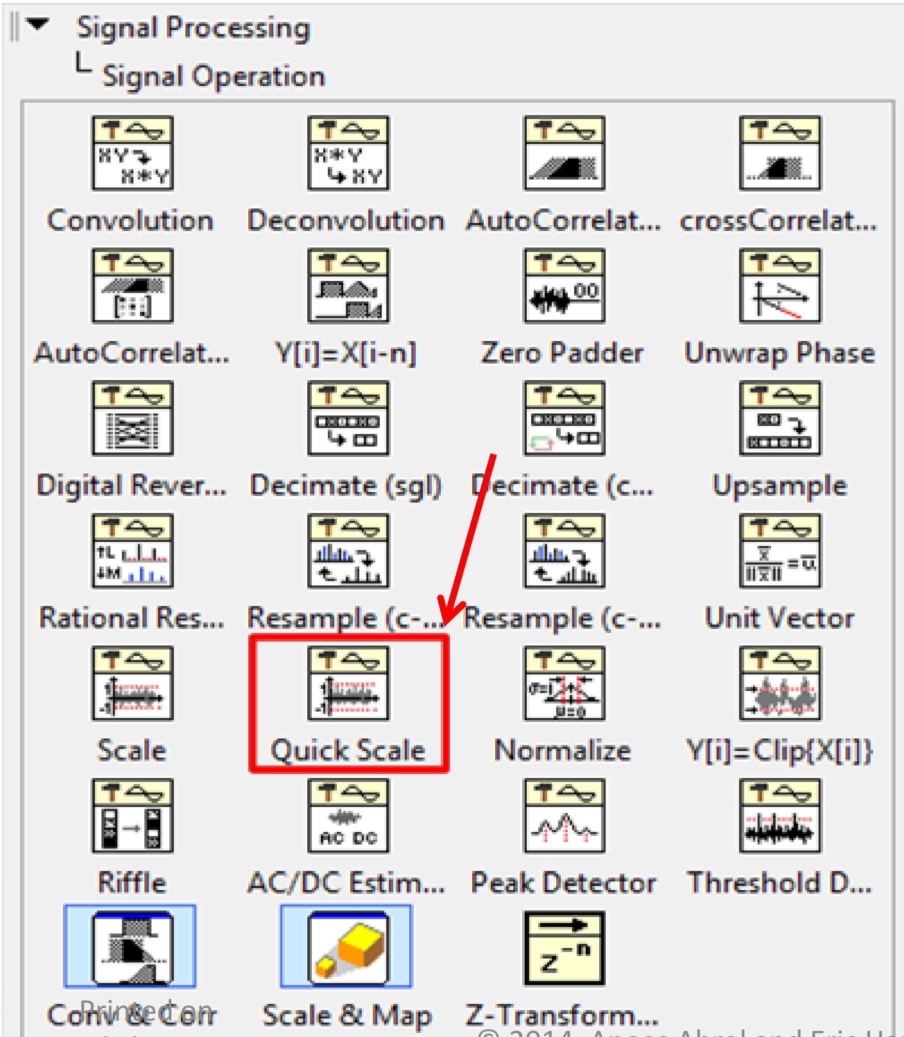
Get Waveform Components

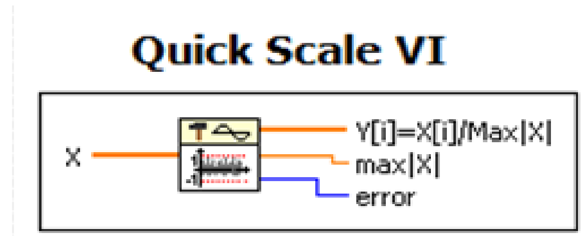


Get Waveform Components (Analog Waveform) Function

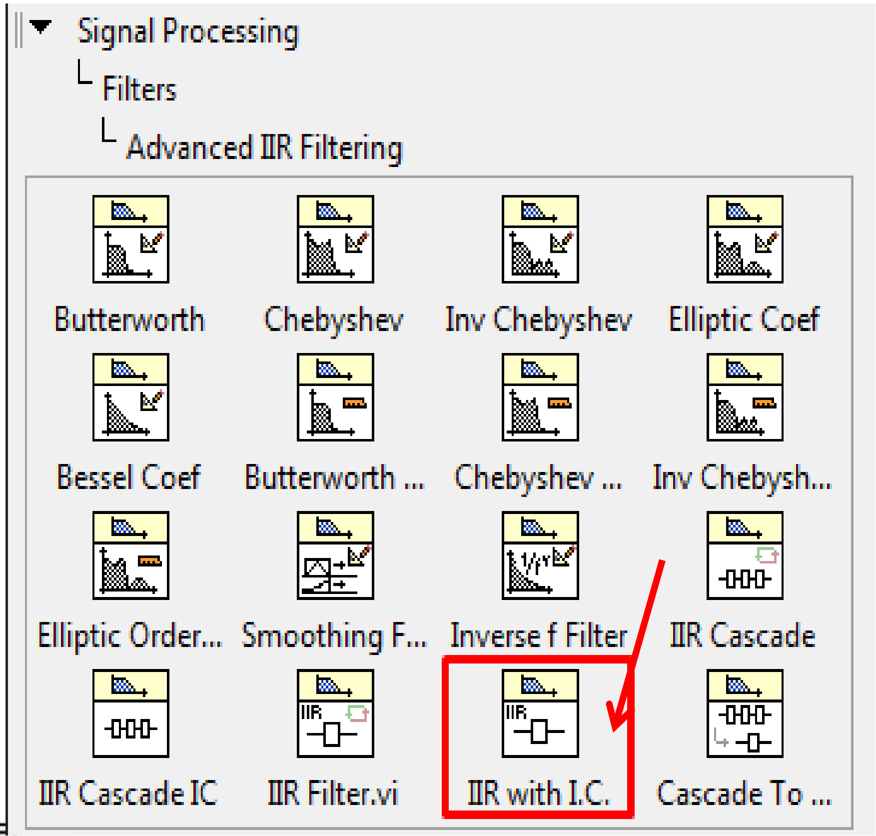


Normalize Message Sequence

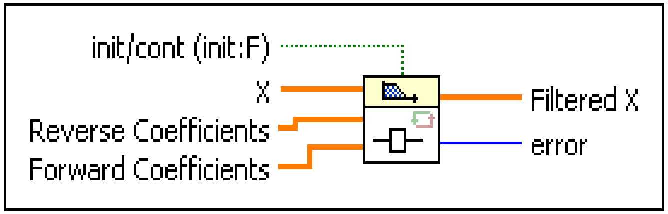




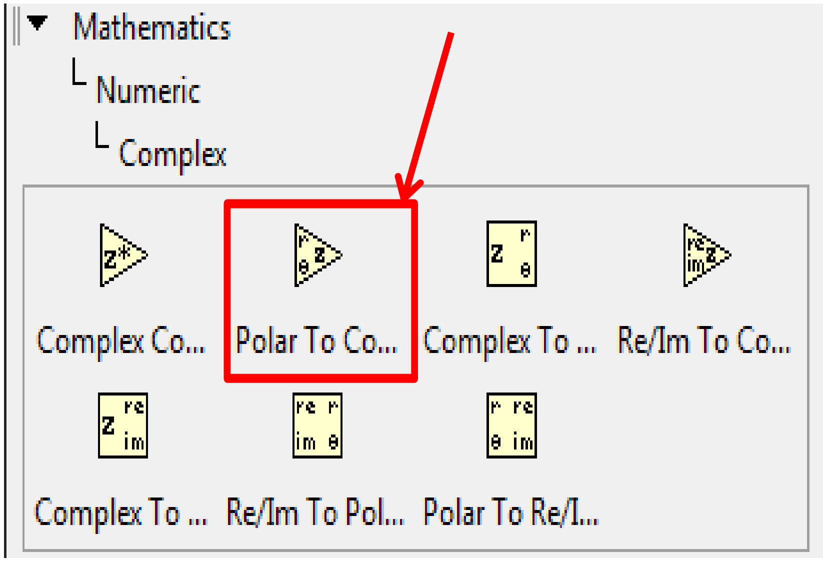
Implement IIR Filter



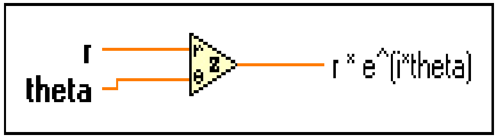
IIR Filter VI



Convert from Polar to Complex form

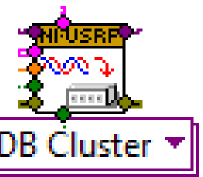


Polar To Complex Function

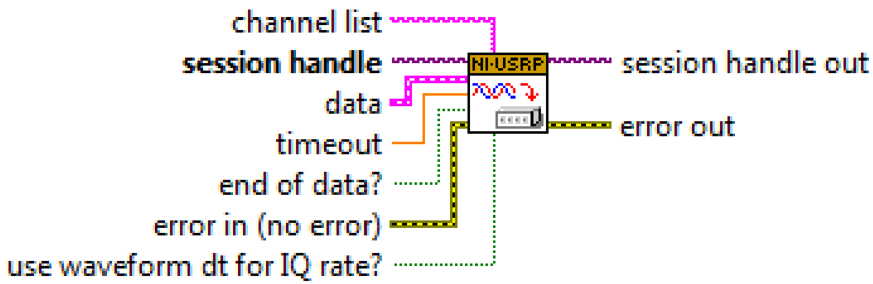


niUSRP Write Tx Data VI

"Buffer to transmit data to receiver"



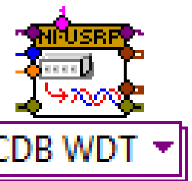
niUSRP Write Tx Data (poly).vi



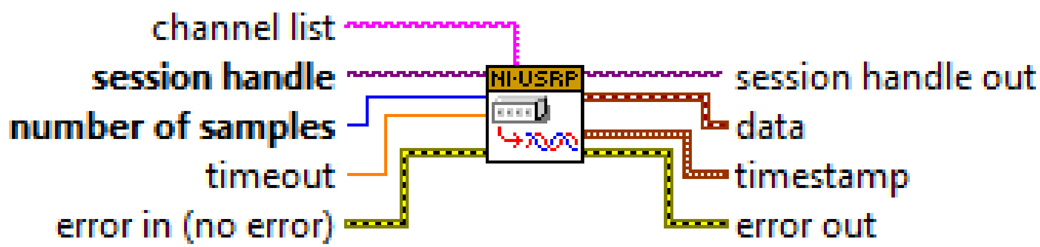
Writes data to the specified channel list.

niUSRP Fetch Rx Data VI

"Buffer to receive data from transmitter"



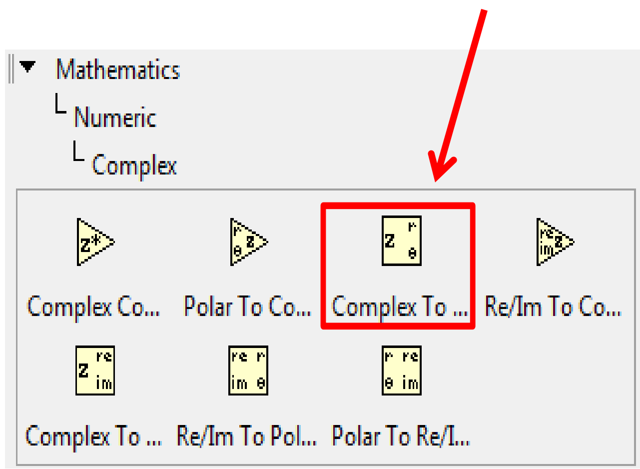
niUSRP Fetch Rx Data (poly).vi



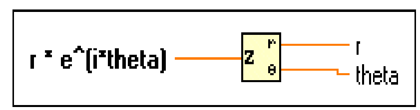
Fetches data from the specified channel list.

Finding the Phase

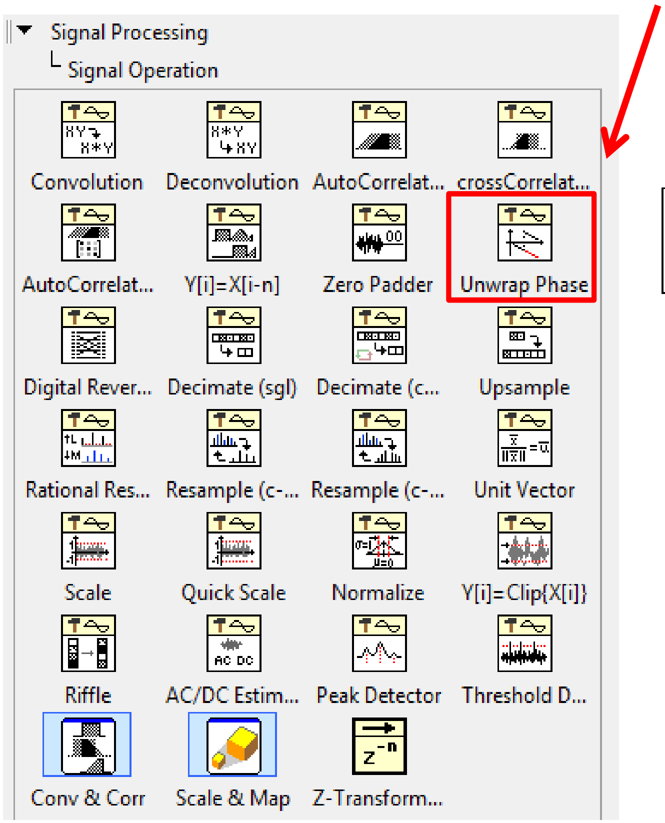
Get Angle (Phase) component by converting from Complex to Polar form

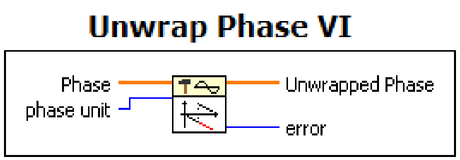


Complex To Polar Function

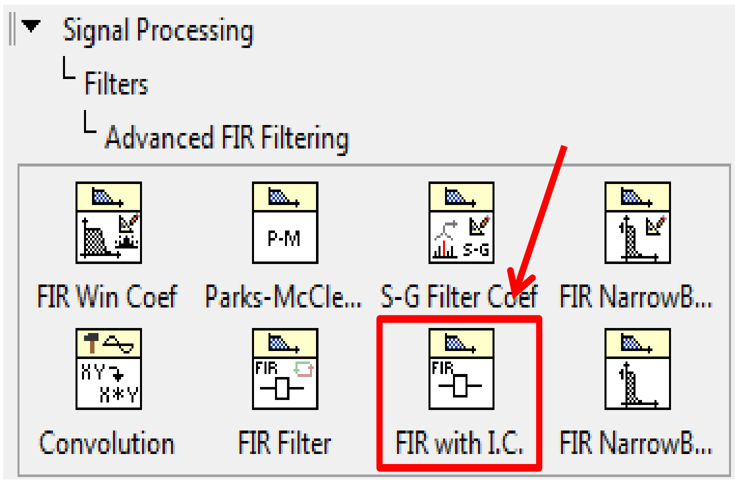


Unwrap the Phase Angle

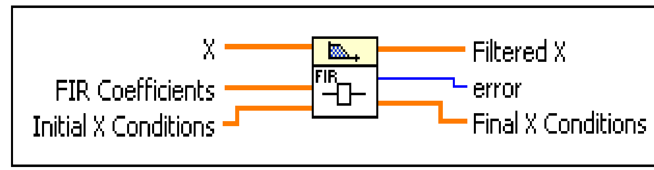




Implement Difference Equation



FIR Filter with I.C. VI



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

FIR Co-efficients

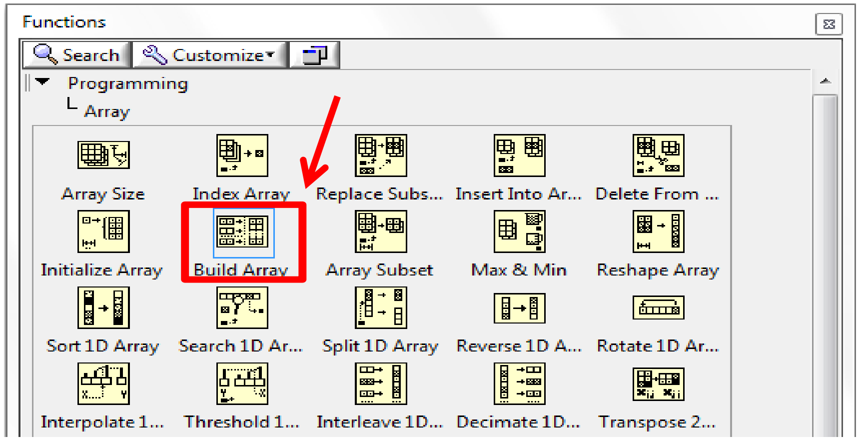
$$\left(\frac{1}{T}, -\frac{1}{T}\right).$$

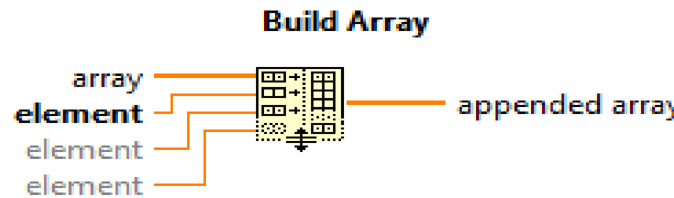
Ibero-American Science & Technology Education Consortium

Global Innovation Network for Entrepreneurship and Technology

THE UNIVERSITY of NEW MEXICO

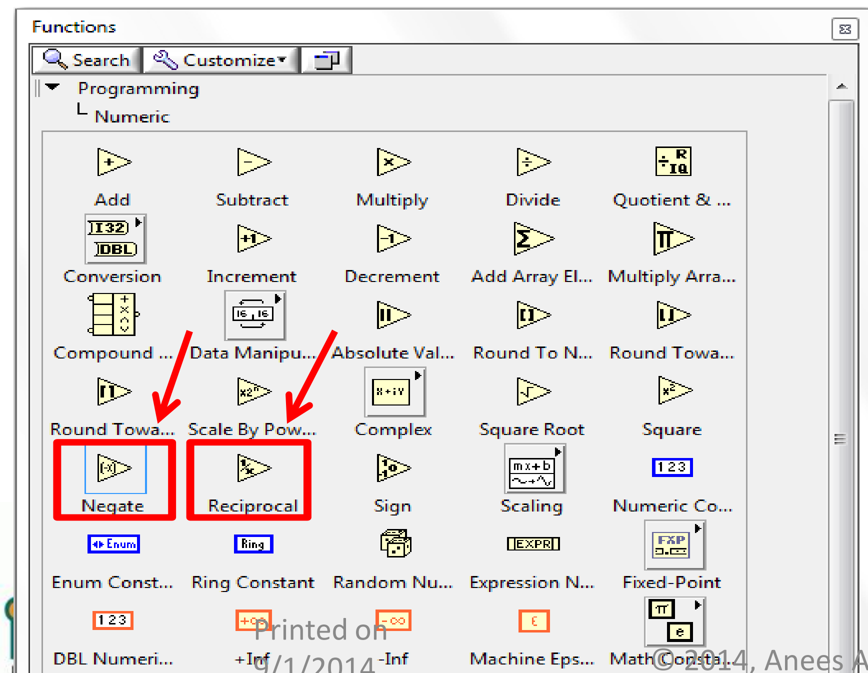
FIR Coefficients Array

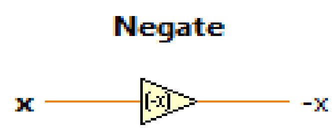




$$\left(\frac{1}{T}, -\frac{1}{T}\right).$$

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

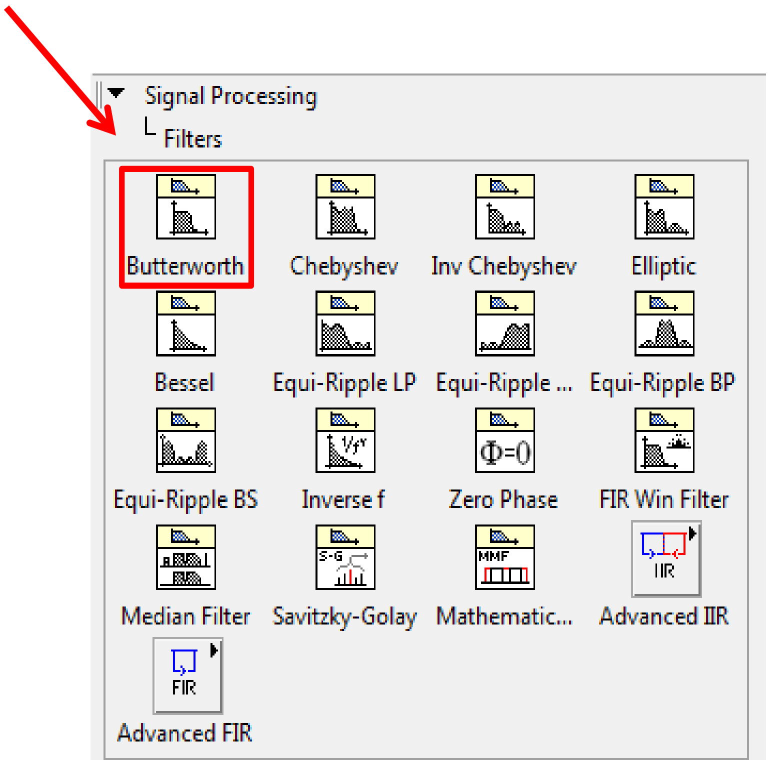




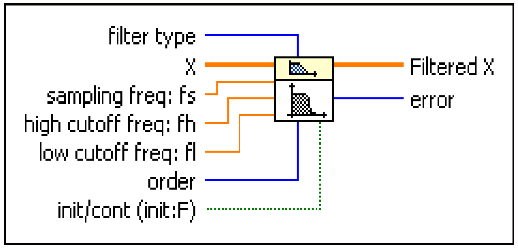
Negates the input value.

Divides 1 by the input value.

Envelope Detector Implementation



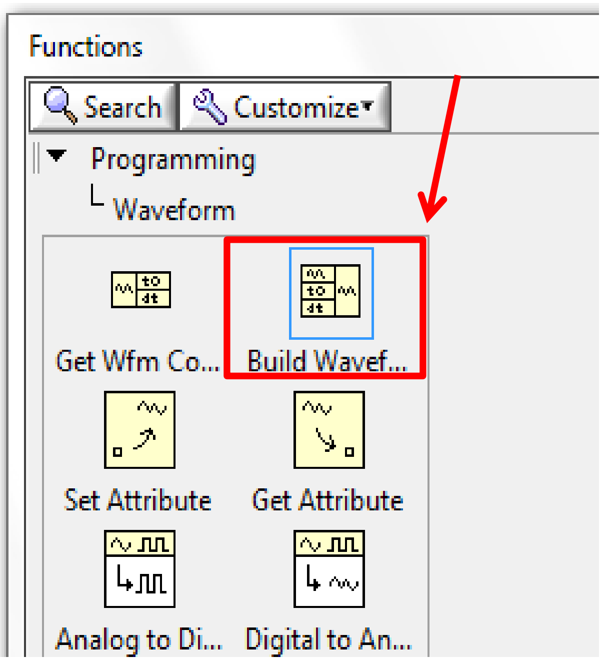
Butterworth Filter VI

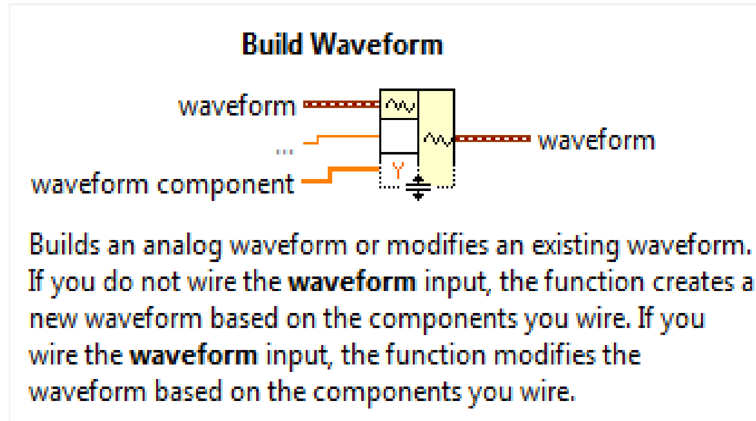


Low-pass Butterworth Filter

THE UNIVERSITY of NEW MEXICO

Build Waveform VI



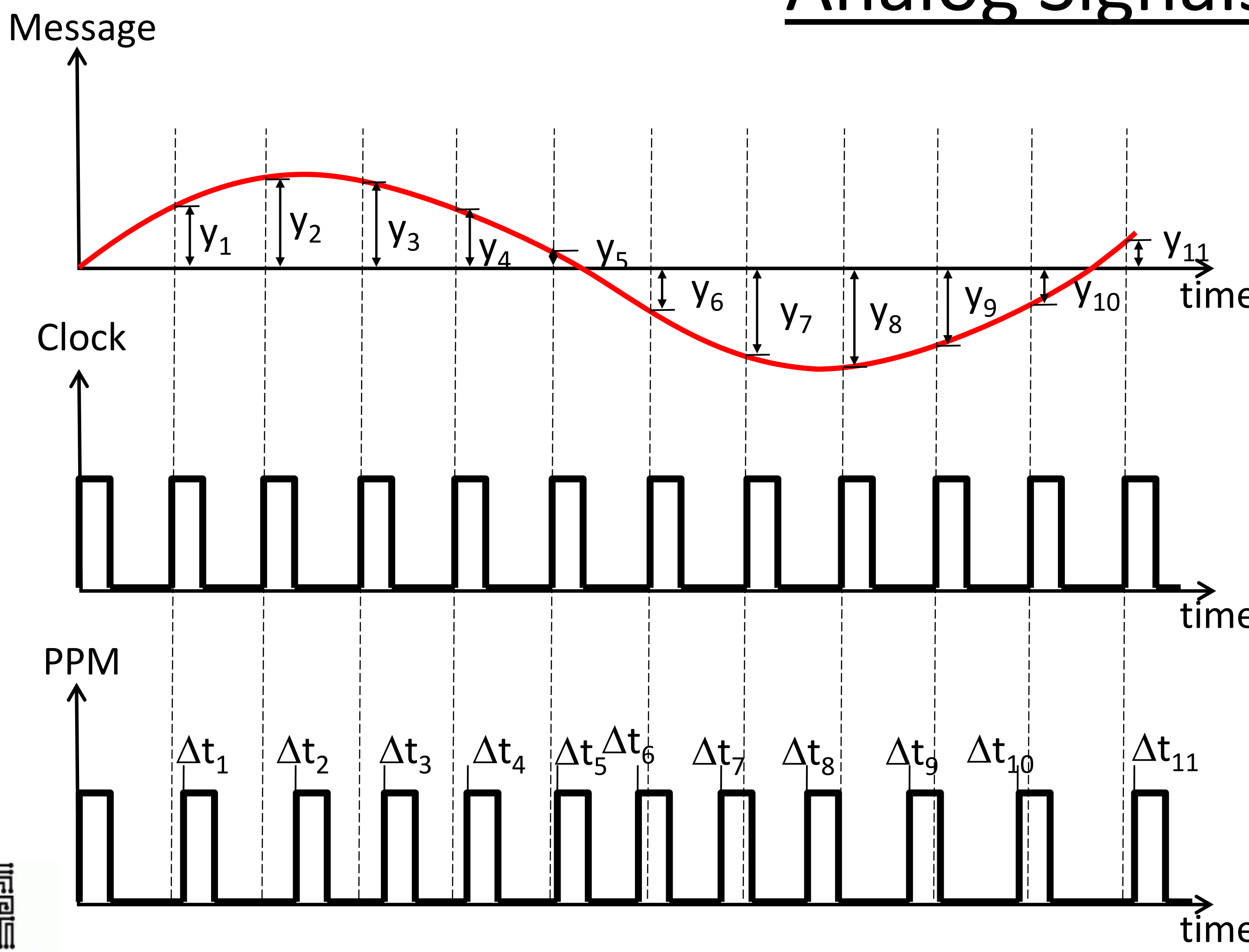


THE UNIVERSITY of NEW MEXICO

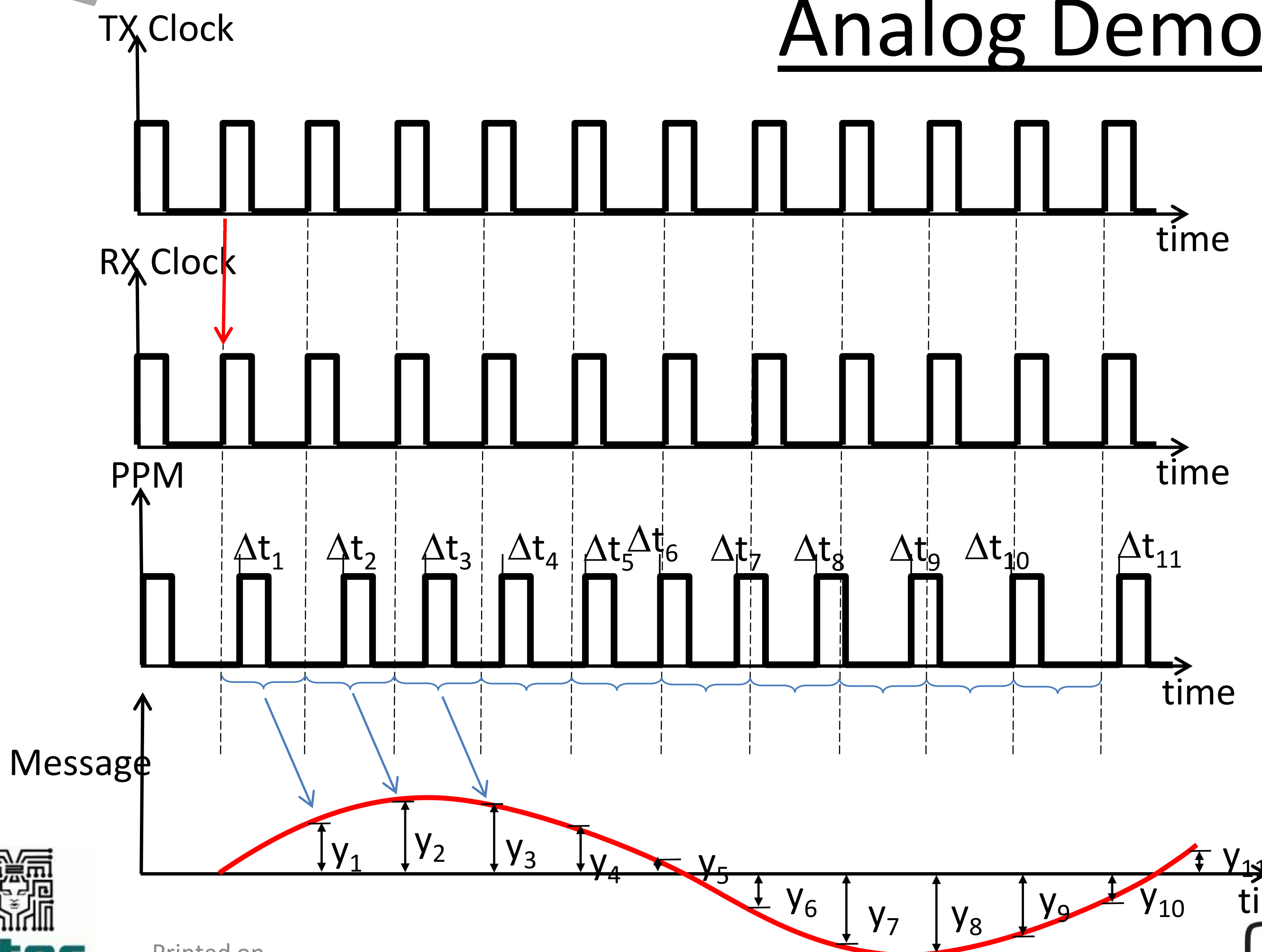
Pulse Position Modulation

What you need to know to do the Lab…

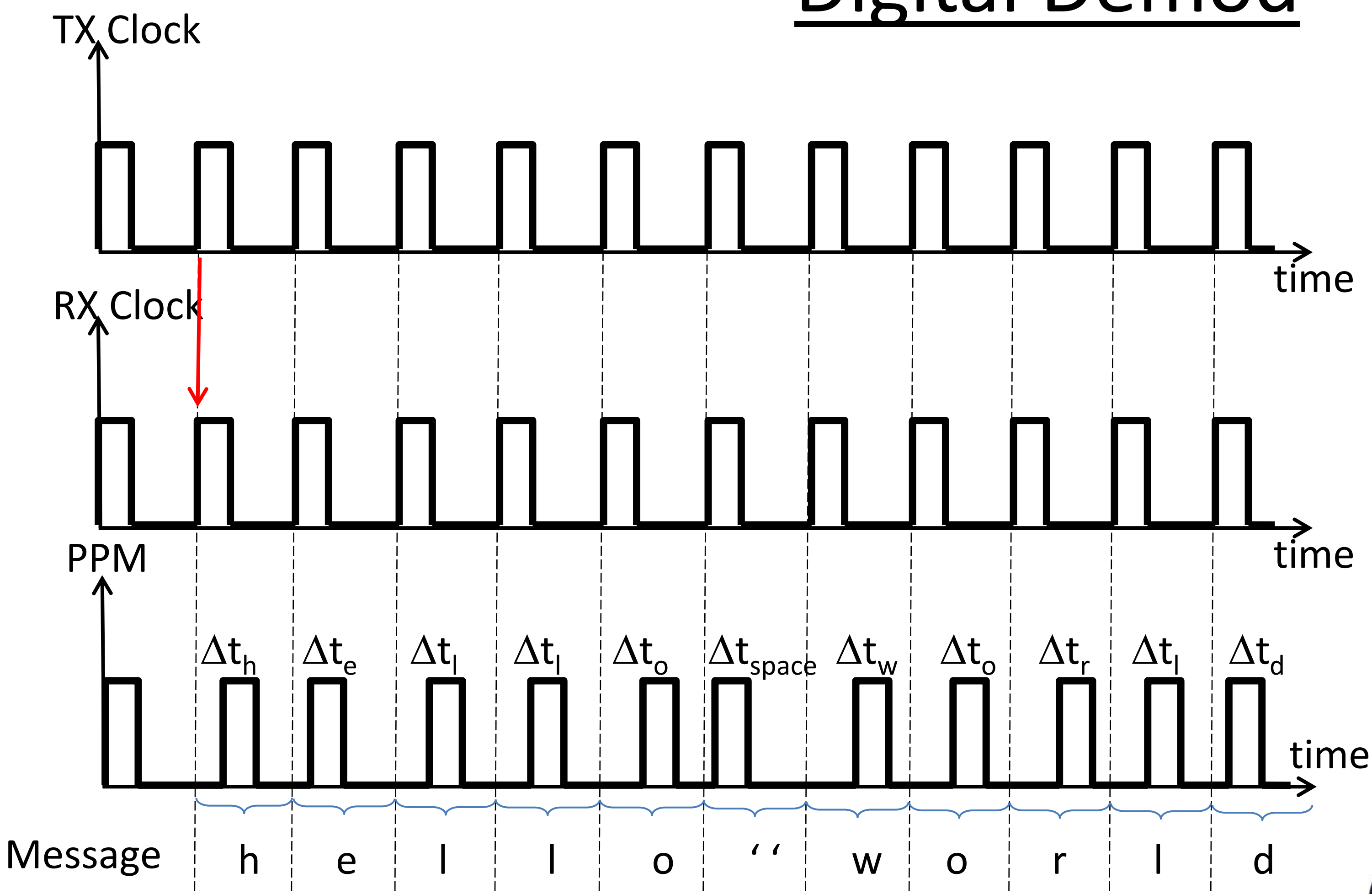
PPM Overview
Analog Signals



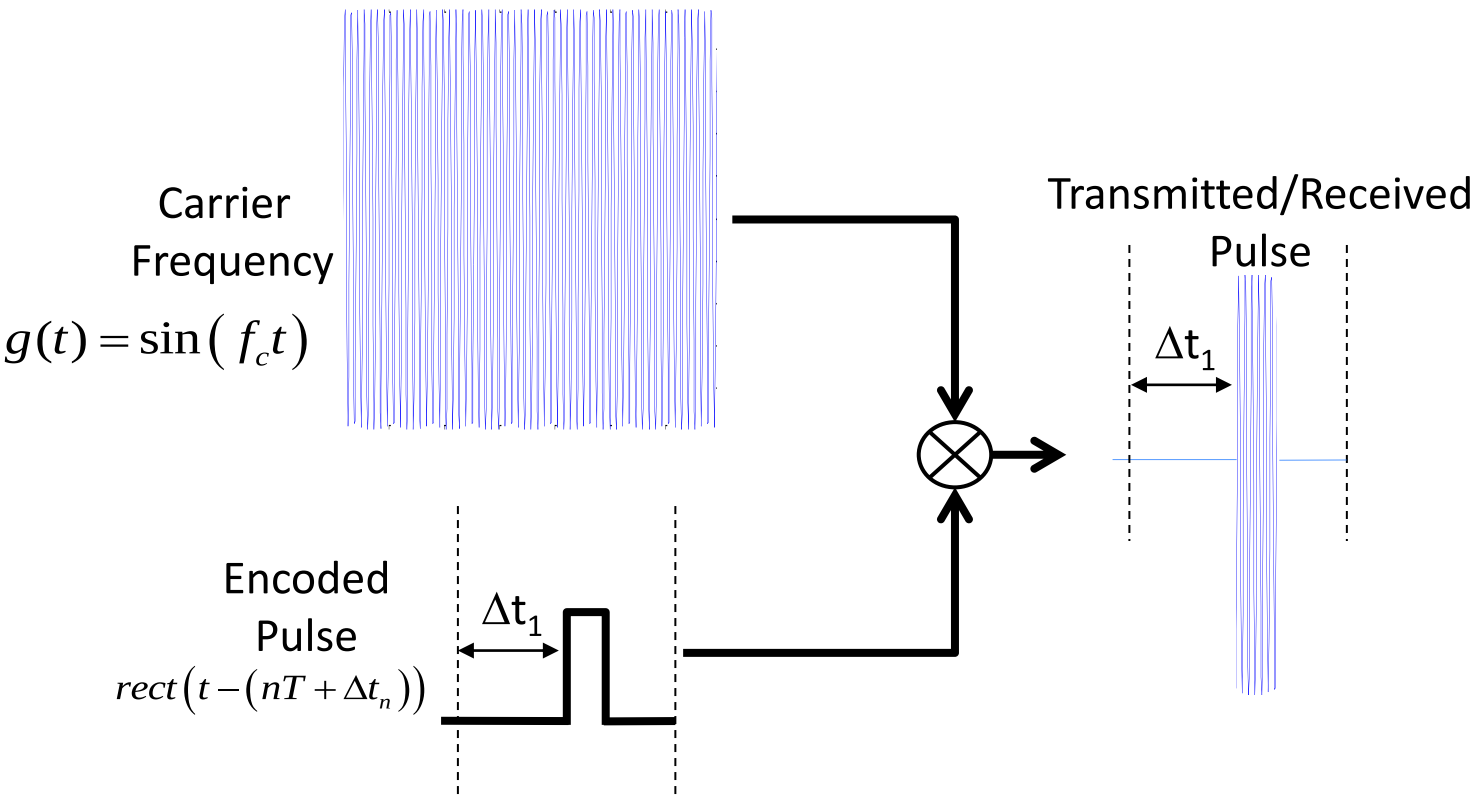
PPM Overview
Analog Demod



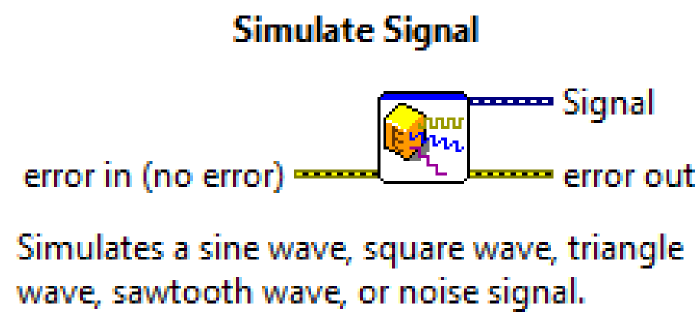
PPM Overview
Digital Demod

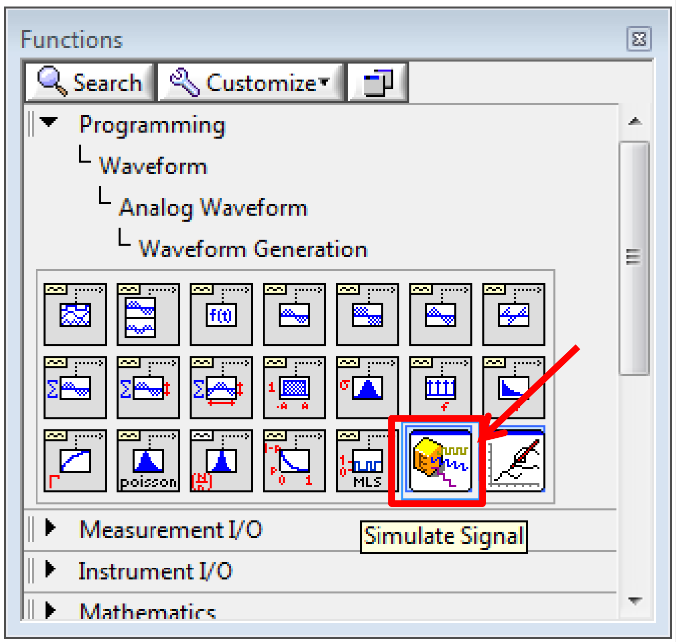


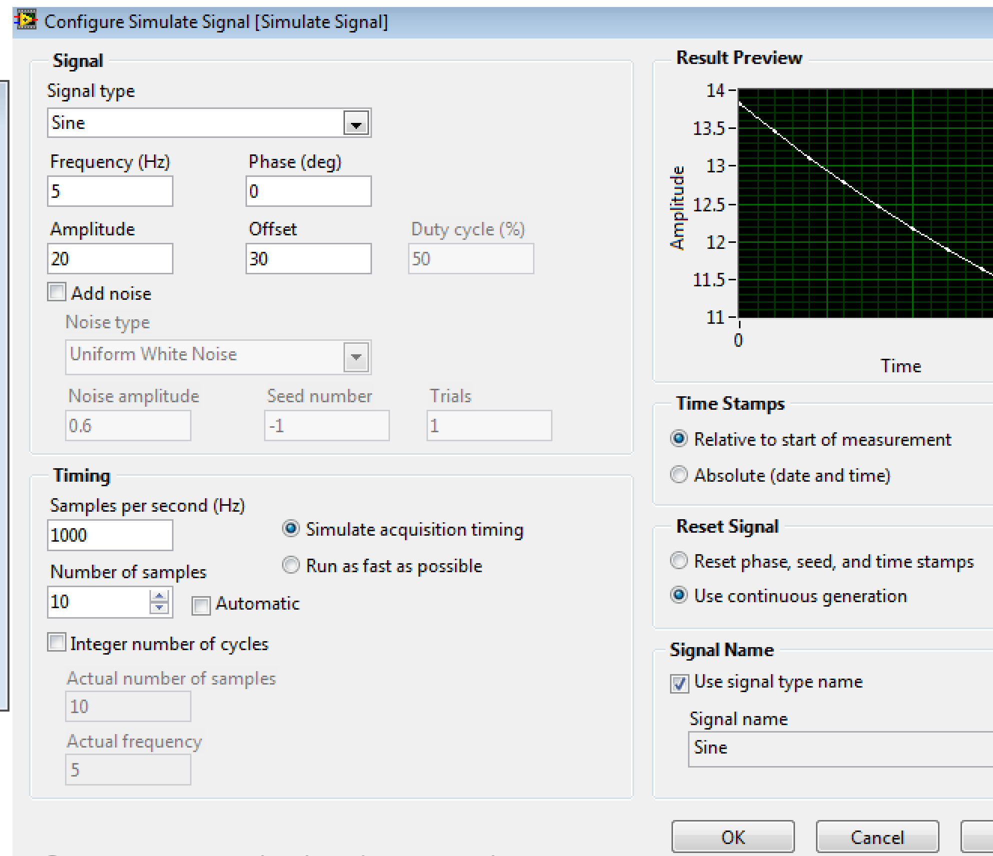
PPM Implementation



Simulate Signal VI

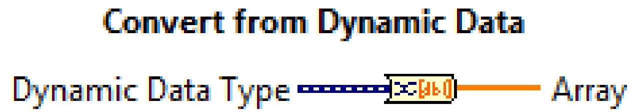




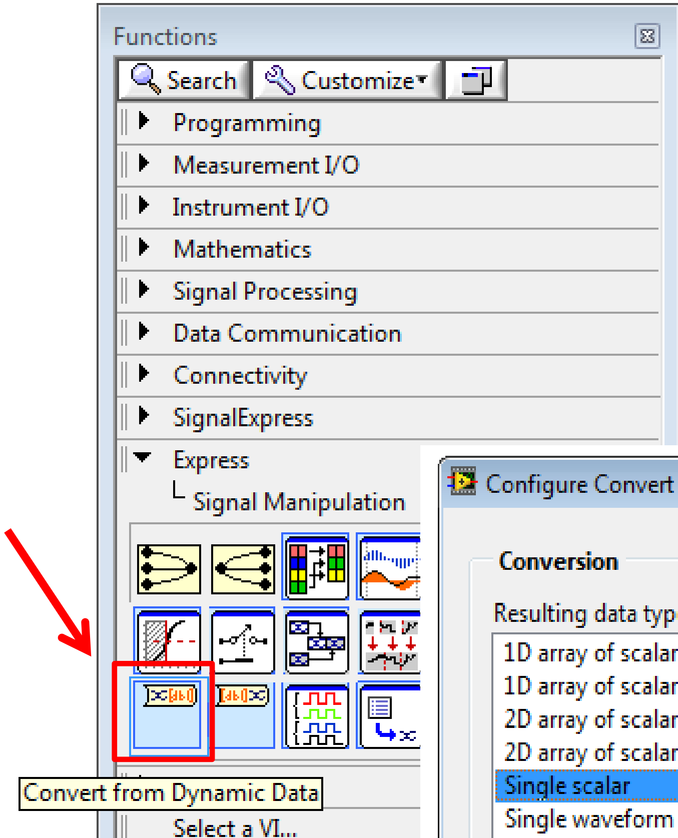


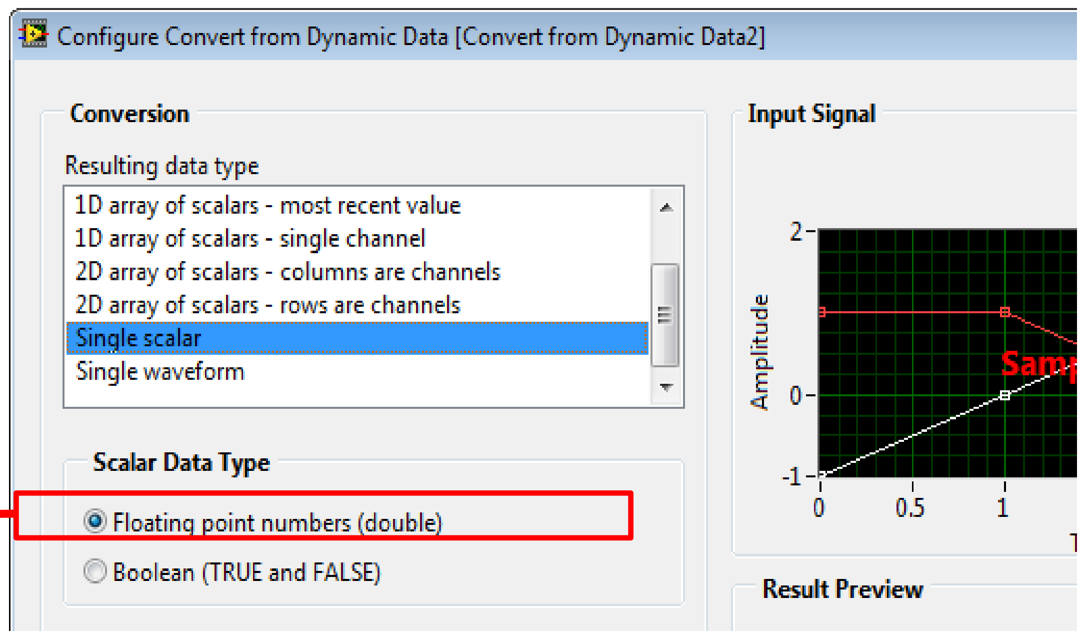
THE UNIVERSITY of NEW MEXICO

Convert From Dynamic Data subVI



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



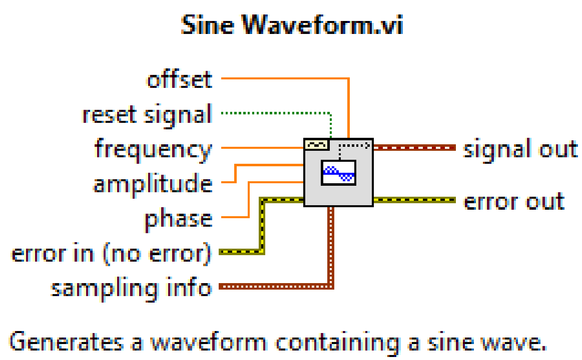


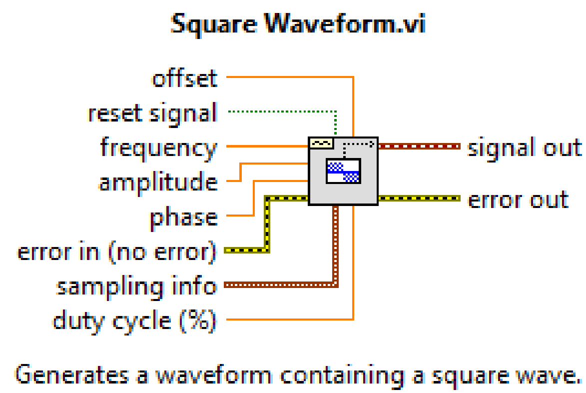
"Single Scalar"

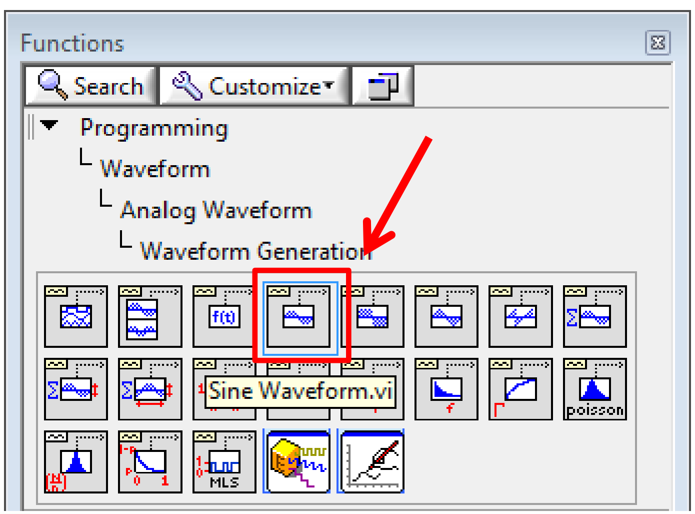
Ibero-American Science & Technology Education Consortium

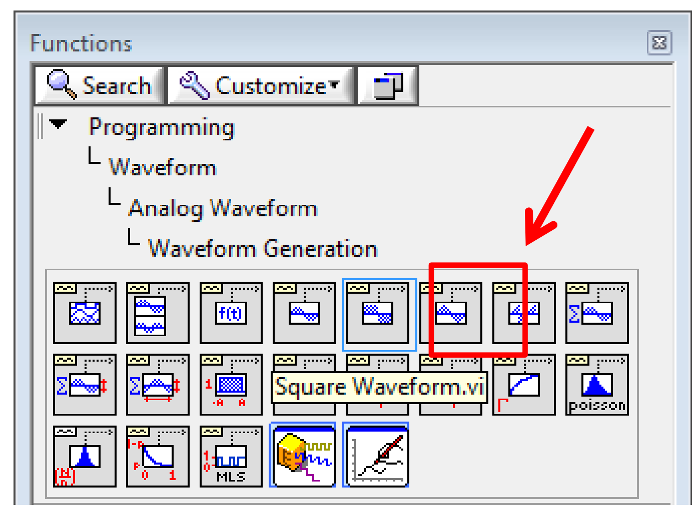
GINET Global Innovation Network for Entrepreneurship and Technology

Sine and Square Waveform subVIs



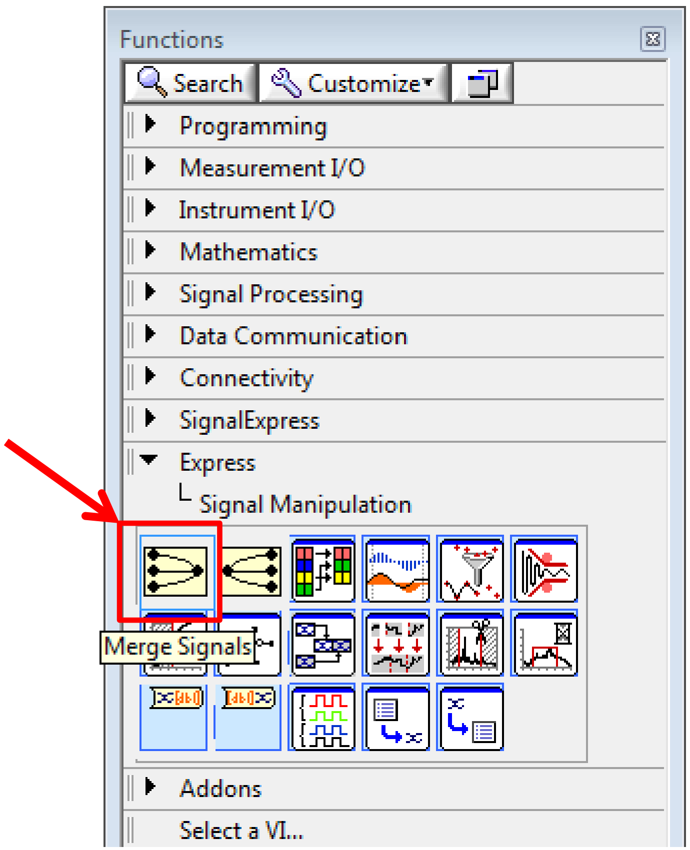


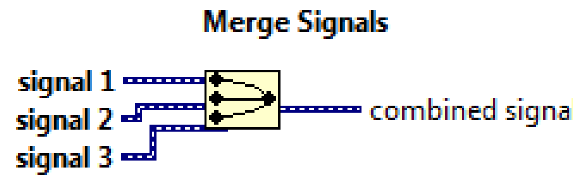




THE UNIVERSITY of NEW MEXICO

Merge Signals VI



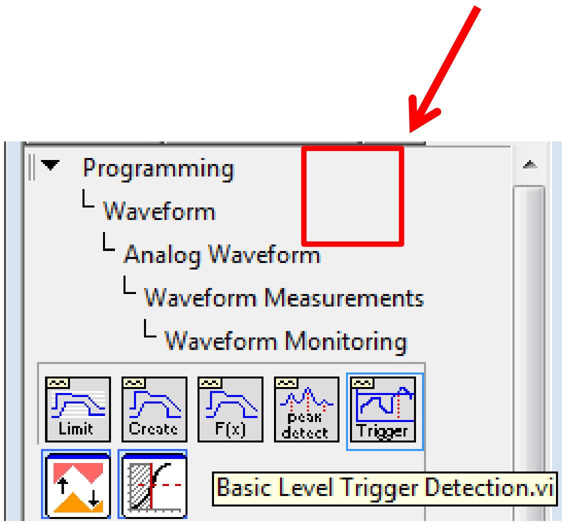


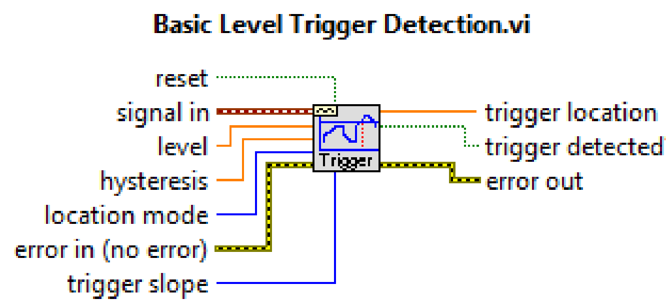
Merges two or more signals into a single output. Resize the function to add inputs. This function appears on the block diagram automatically when you wire a signal output to the wire branch of another signal.

istec Ibero-American Science & Technology Education Consortium

GINET Global Innovation Network for Entrepreneurship and Technology

Basic Level Trigger Detection VI





Finds the first level-crossing location in a waveform. You can retrieve the trigger location as an index or as a time. The trigger conditions are specified in terms of threshold **level**, **slope**, and **hysteresis**. Wire data to the **signal in** input to determine the polymorphic instance to use or manually select the instance.

Basic Level Trigger Interface

TF **reset** specifies whether the history, or internal state, of the VI has to be reset. The default is FALSE. The internal state contains the final state of the input signal. The VI uses this as the initial state the next time LabVIEW calls the VI.

signal in contains the signal in which to detect a trigger.

DBL **level** specifies the threshold value **signal in** must cross before a trigger is detected. The default is 0.

DBL **hysteresis** specifies the amount above or below **level** through which **signal in** must pass before a trigger level crossing is detected. The default is 0.

Trigger hysteresis is used to prevent noise from causing a false trigger. For a rising edge **trigger slope**, the signal must pass below **level – hysteresis** before a trigger level crossing is detected. For a falling edge **trigger slope**, the signal must pass above **level + hysteresis** before a trigger level crossing is detected.

U16 **location mode** specifies whether you want to retrieve the trigger location as an index into the Y-array of the waveform or as a point in time in seconds.

0	**Index** (default)—Retrieves the trigger location in terms of an array index.
1	**Time**—Retrieves the trigger location in terms of time in seconds. Time is computed by the following equation: time = t0 + (index*dt), where t0 and dt are contained in **signal in**. Use the To Time Stamp Function to convert this number to a time stamp data type with a time and date format.

error in describes error conditions that occur before this node runs. This input provides standard error in functionality.

U16 **trigger slope** specifies whether a trigger is detected as **signal in** crosses **level** on a rising edge or a falling edge

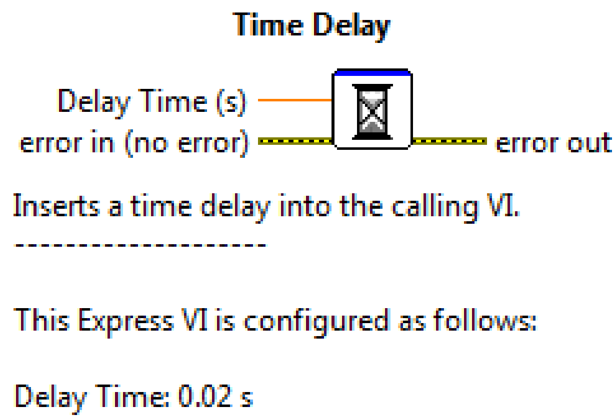
0	**Falling Edge**—The VI detects a trigger on the falling edge, or negative slope.
1	**Rising Edge** (default)—The VI detects a trigger on the rising edge, or positive slope.

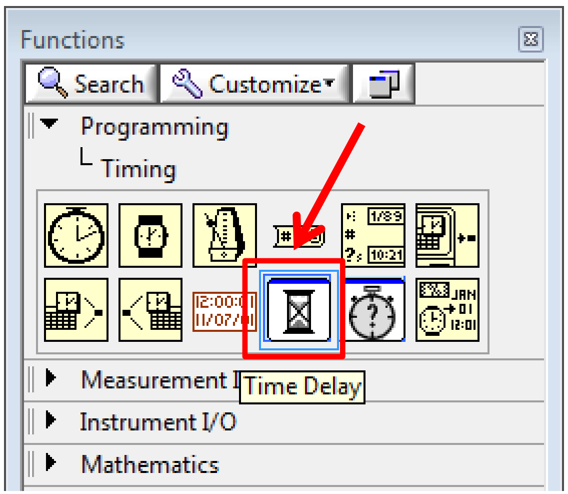
DBL **trigger location** contains the index or time, depending on the **location mode** setting, of the detected trigger. If the **location mode** is in **Time** mode and you do not want the **trigger location** value to appear in seconds on the front panel, wire the **trigger location** to a time stamp.

TF **trigger detected?** indicates whether the VI detects a valid trigger. If **trigger detected?** is TRUE, the VI detects a valid trigger.

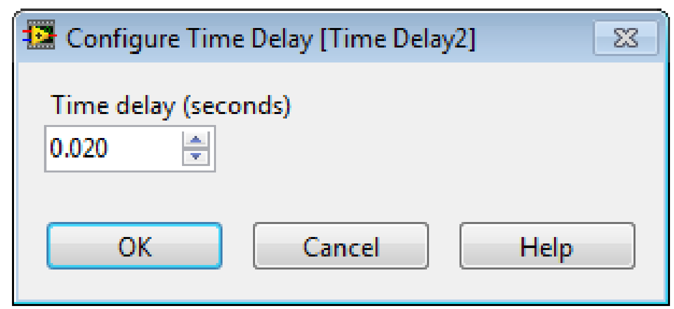
error out contains error information. This output provides standard error out functionality.

Time Delay VI





1. Select, hold and drop VI



2. Double click to set time delay in seconds.

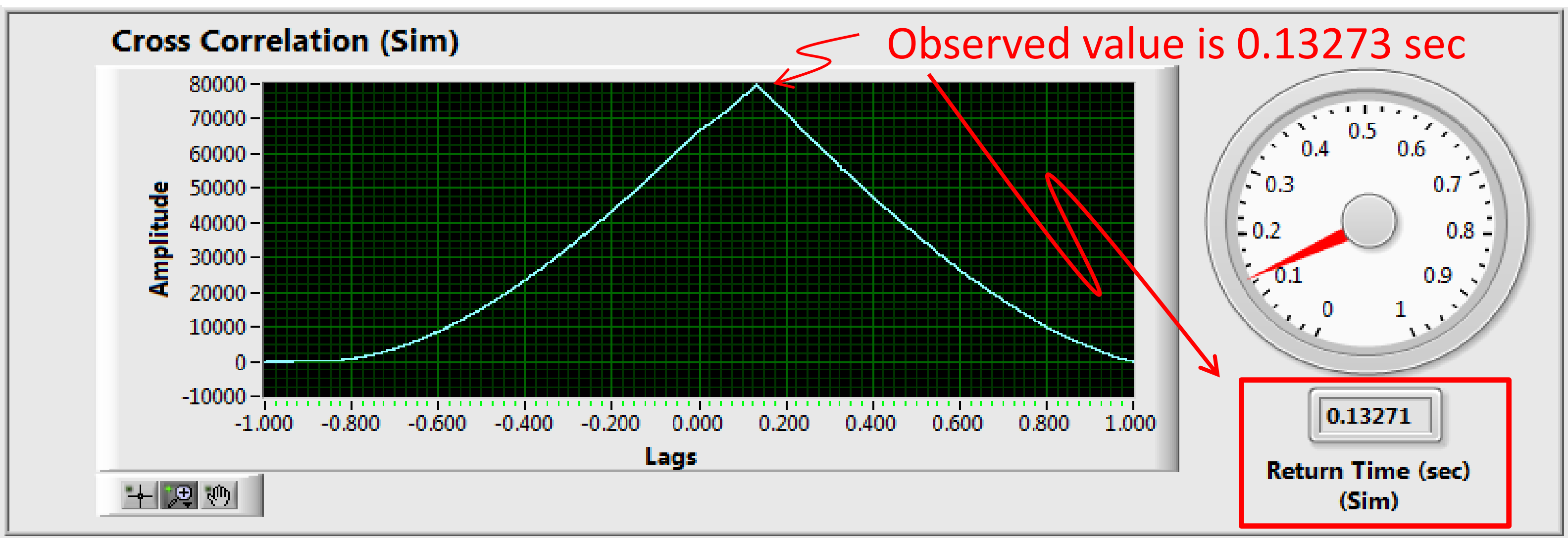
Random Process, Crosscorrelation and Power Spectral Density

What you need to know to do the Lab ...

Crosscorrelation

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

$$r_{pr}(l) = \sum_{n=-\infty}^{\infty} p^*[n]r[n+l] \qquad (1)$$

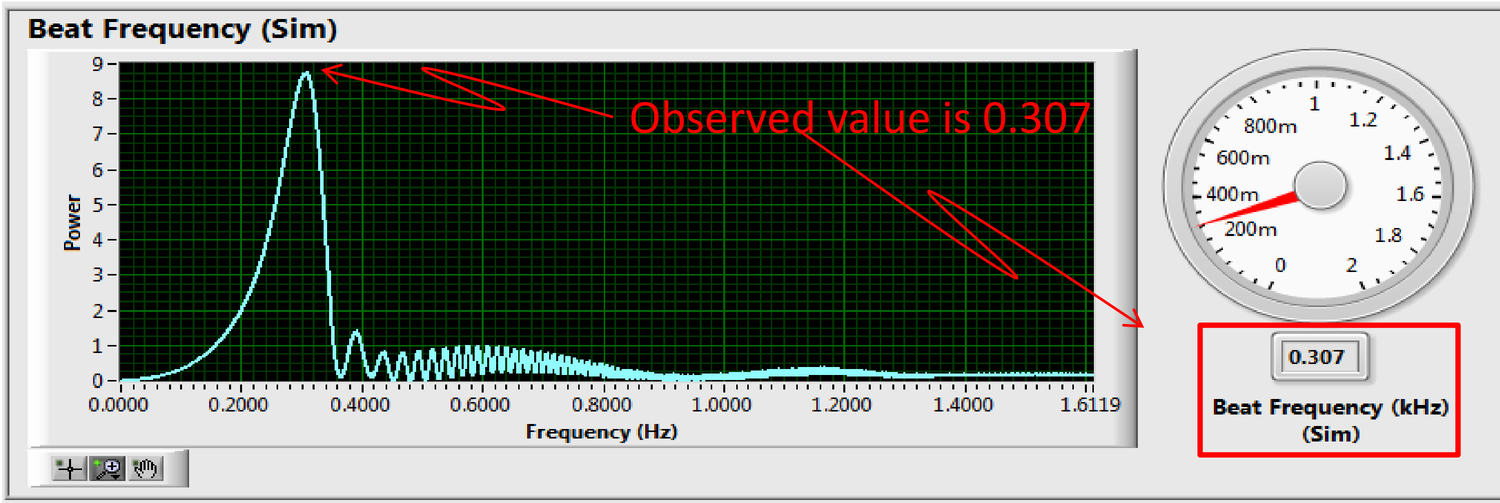


The propagation delay of the echo (τ) is $\tau = 2\,\mathrm{r}/\mathrm{c}$ where, c is the speed of light ($2.98\ x\ 10^8 m/sec$)

Power Spectral Density

The Power Spectral Density can also be used to estimate the distance. In this approach the return signal and the pulse signal are multiplied together. The product contains the sum and difference frequencies. The sum of frequencies is approximately $2f_c$. This frequency is beyond the frequencies the electronics can respond to. Only the terms related to the difference frequencies are retained (1).

$$\begin{aligned} m(t) &= a_3 \cos[\phi(t) - \phi(t-\tau)] \\ &= a_3 \cos\left(2\pi f_{beat}\, t + 2\pi f_c \tau - \frac{\pi B}{T_m}\tau^2\right) \end{aligned} \qquad (1)$$



$$\tau = \frac{15}{31}\left(f_{beat} - 0.0416\right)$$

LAB Tasks

- Build Beat Frequency analysis subVI.
- Build Cross Correlation analysis subVI.
- Wire your VIs into the J2 V2 RADAR VI.
- Basic procedure
  - You have been supplied with a set of templates and supporting VIs
  - Build both VIs and wire them in.
  - Debugging strategy
    - Use simulation page in J2 V2 RADAR VI.
    - Test case for 20,000 km

Table I – 20,000km Test Case Reference

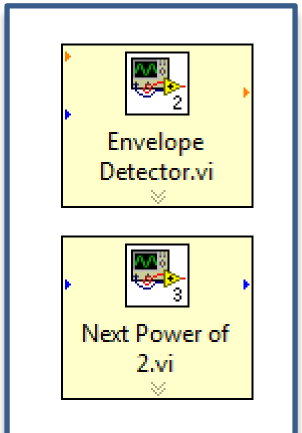
Simulated Distance to Target. (km)	***Return Signal Ramp Reset Time (Sec)***	**Return Time (Sec)**	**Beat Frequency (Hz)**
20000	0.86728 (see **Fig. 19**)	0.13272 (see **Fig. 20**)	0.337 (see **Fig. 21**)

- Please refer to debugging presentation for tools and techniques

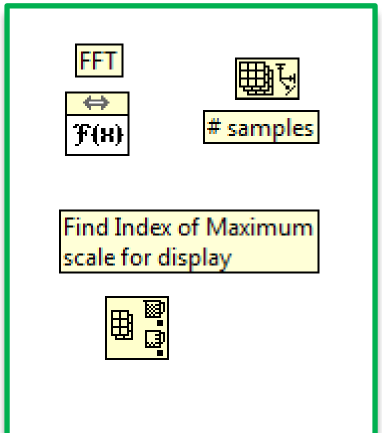
Beat Frequency SubVI

THE UNIVERSITY of NEW MEXICO

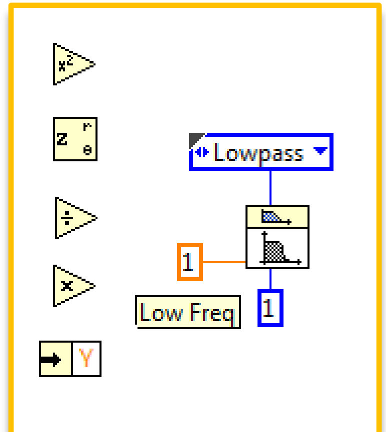
These are provided in the template



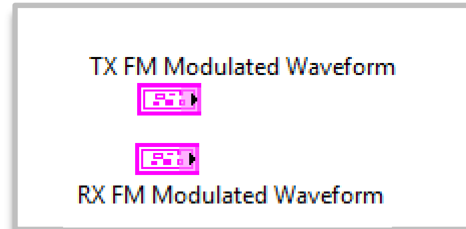
These are new blocks to be used in the lab



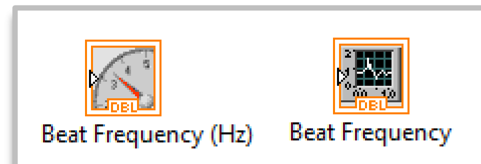
These are blocks you have used in previous labs



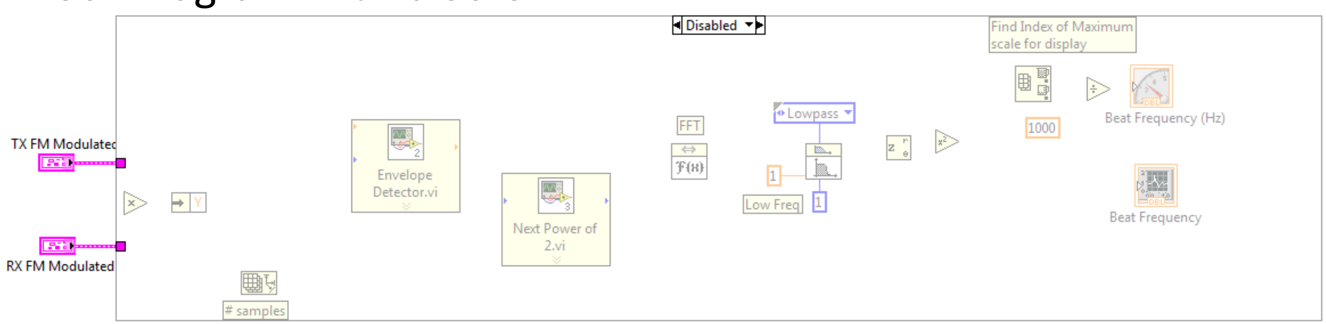
Input Controls



Output Indicators



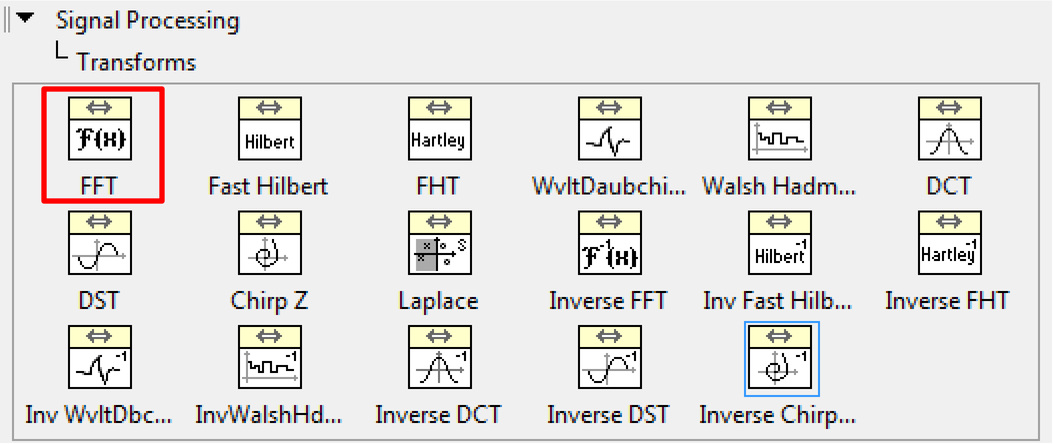
Block Diagram with blocks



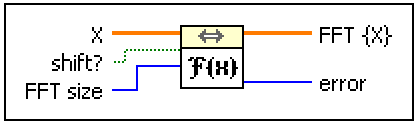
istec Ibero-American Science & Technology Education Consortium

GINET Global Innovation Network for Entrepreneurship and Technology

The FFT Block

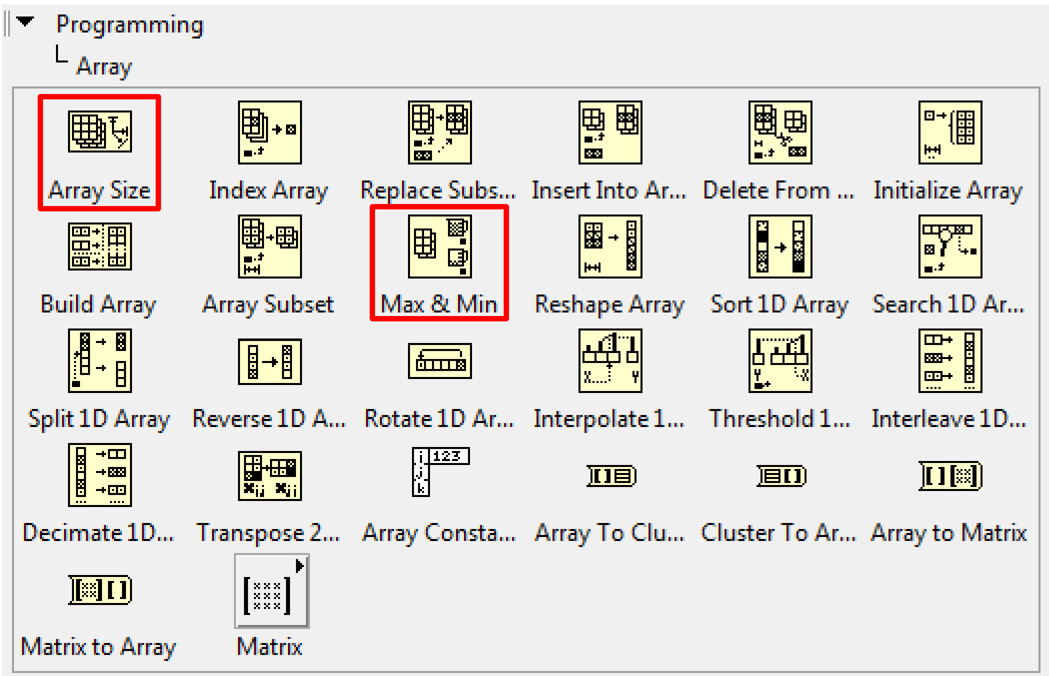


FFT VI (Fast Fourier Transform)

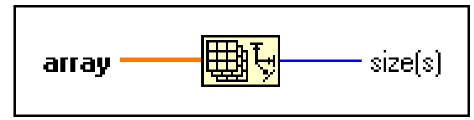


Computes the fast Fourier transform (FFT) of the input sequence **X**.

Array Data Processing

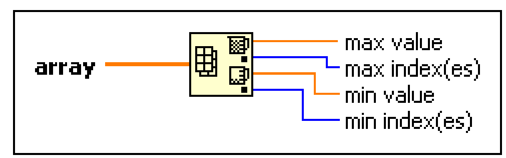


Array Size Function



Returns the number of elements in each dimension of **array**.

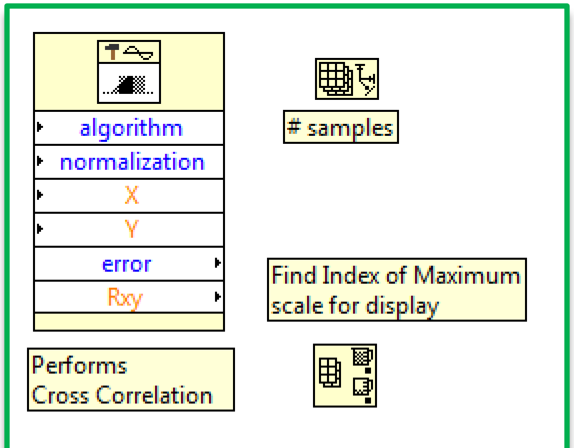
Array Max & Min Function



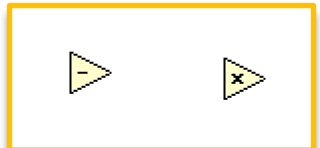
Returns the maximum and minimum values found in **array**, along with the indexes for each value.

Cross Correlation and Return Time Analyzer

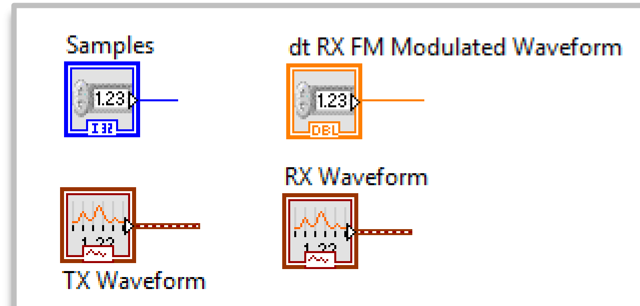
These are new blocks to be used in the lab



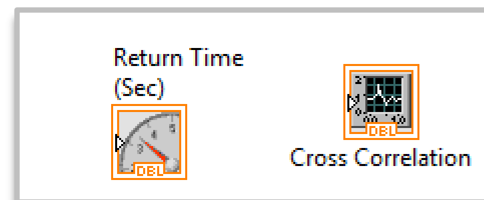
These are blocks you be used in previous labs

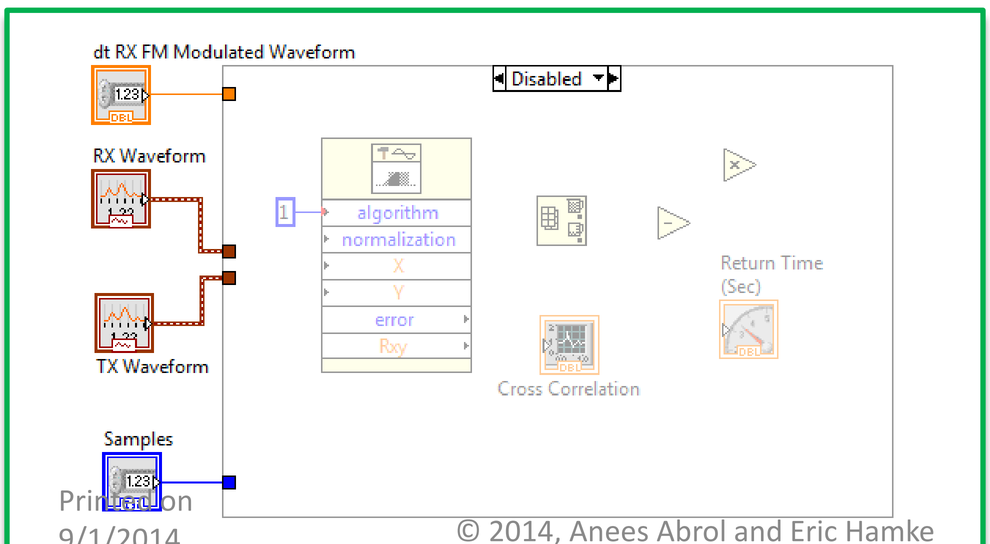


Input Controls

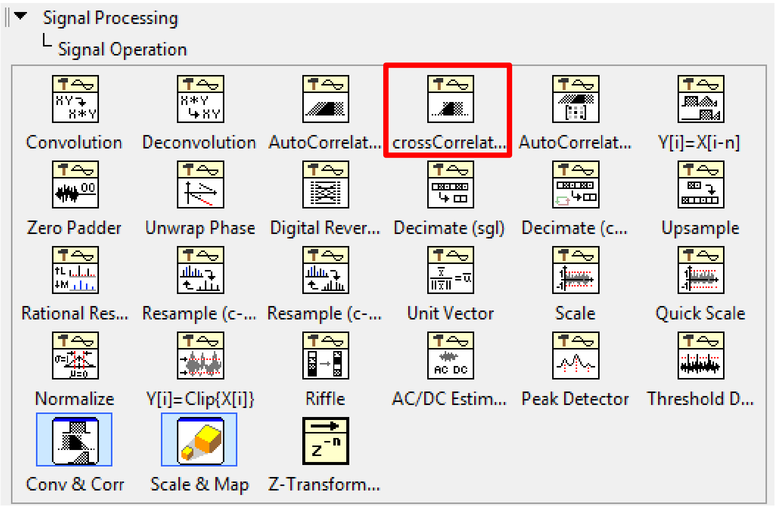


Output Indicators

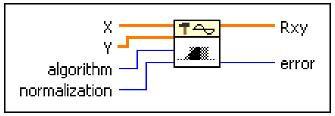




Cross Correlation



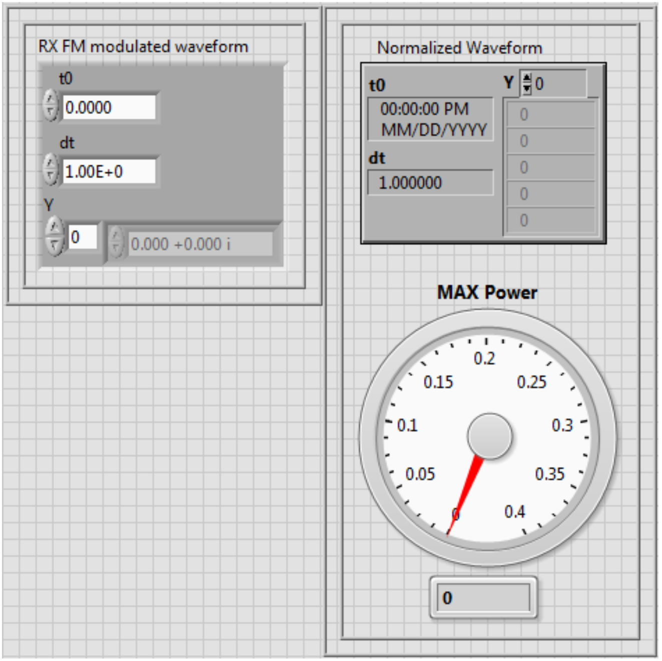
CrossCorrelation VI

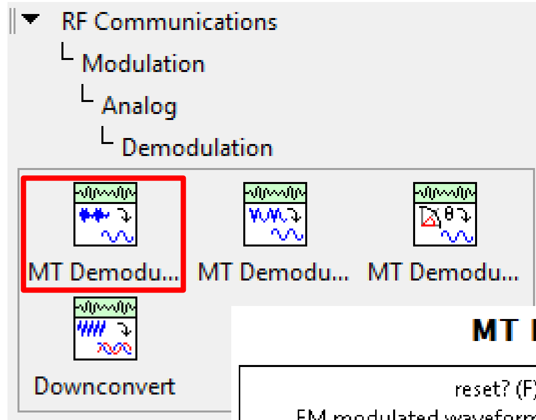


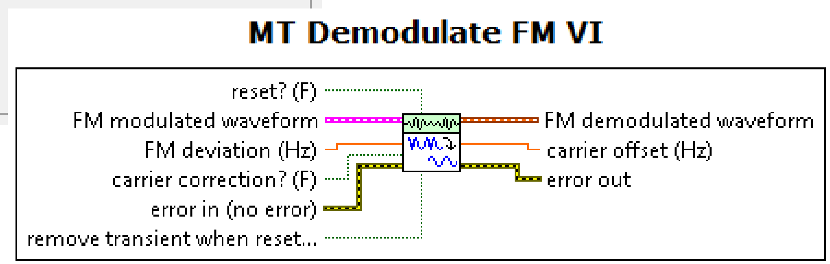
Computes the cross correlation of the input sequences **X** and **Y**. Wire data to the **X** and **Y** inputs to determine the polymorphic instance to use or manually select the instance.

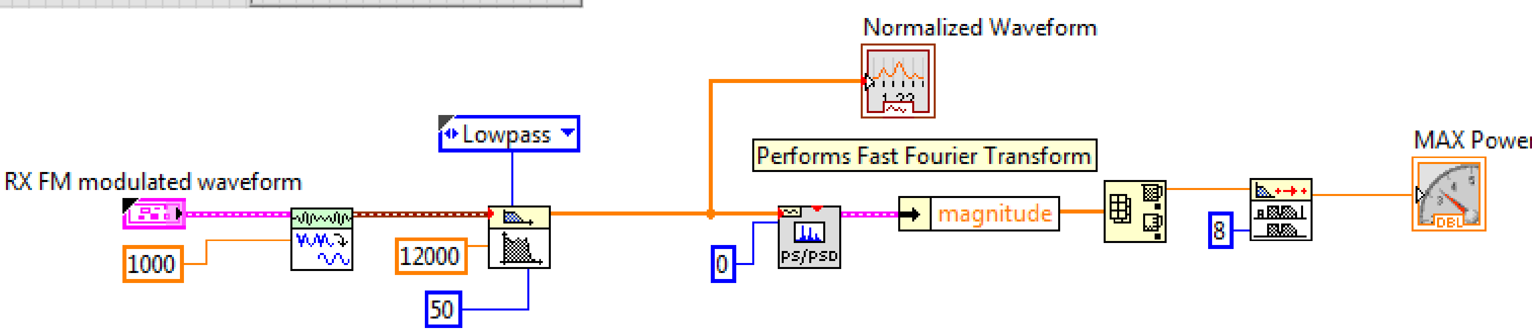
Sub VIs Provided

Demodulate SubVI

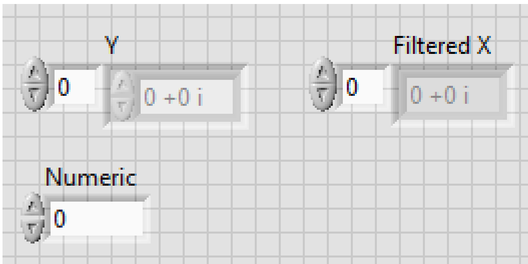


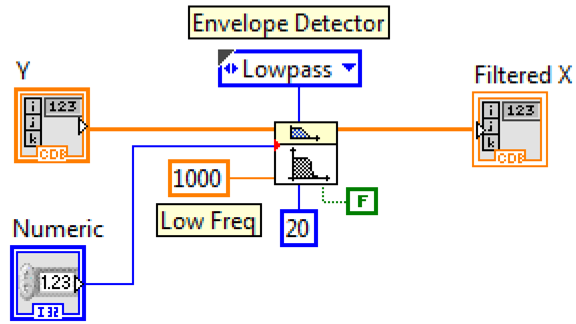






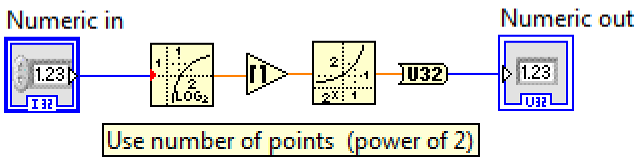
AM Envelope Detector



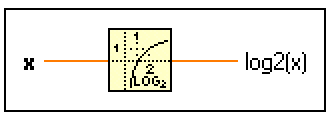


Should look familiar since you designed one on the AM Lab

Next Power of 2



Logarithm Base 2 Function



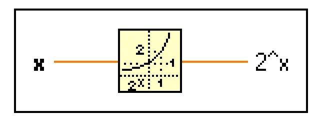
$$\log_2(x) = \frac{\ln(x)}{\ln(2)}$$

Round Toward +Infinity Function

Rounds the input to the next highest integer.

For example, if the input is 3.1, the result is 4. If the input is –3.1, the result is –3. The connector pane displays the default data types for this polymorphic function.

Power Of 2 Function



THE UNIVERSITY *of* NEW MEXICO

Amplitude Modulation with Additive Gaussian White Noise

What you need to know to do the Lab…

Ibero-American Science & Technology Education Consortium

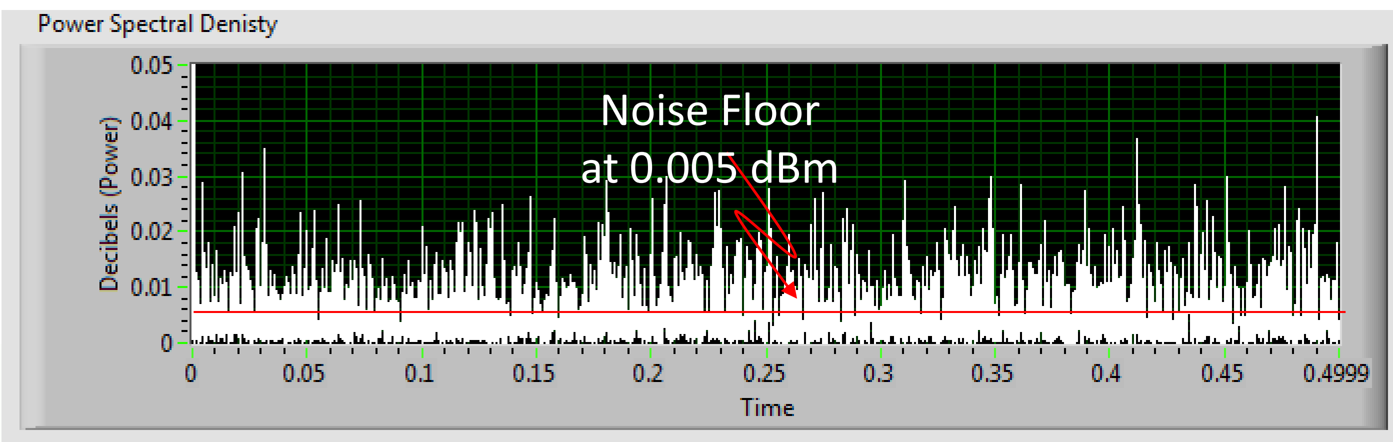
Printed on 9/1/2014

GINET Global Innovation Network for Entrepreneurship and Technology

Noise Floor

The **Noise Floor** reflects the effect of random processes that are the result of many natural sources, such as:

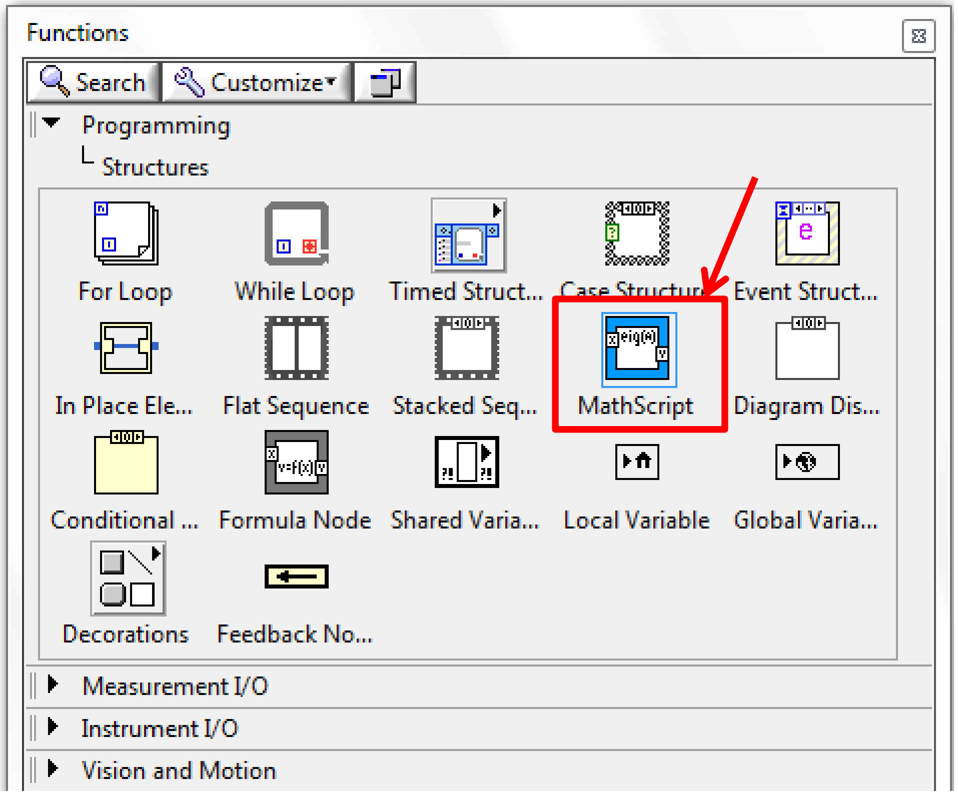
- Thermal noise is the result of vibrations of atoms in conductors resulting thermal energy;
- Shot noise is the result of random fluctuations in the movement of current in discrete electric charge quanta or electrons.
- Electromagnetic radiation emitted by the sun, earth and other large masses in thermal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



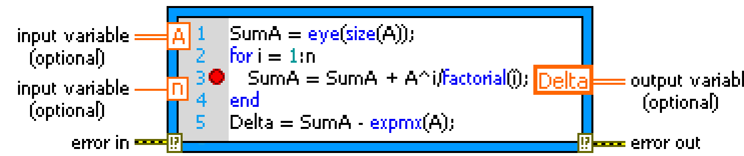
Changing the Noise Floor Using AGWN

- **Additive white Gaussian noise** (**AWGN**) is used to simulate the effect of many random processes too complicated to model explicitly.
  - The model is assumed to be linear so that the noise can be super imposed or added to the message or modulated signal.
  - A white noise process is assumed to uniformly affect all frequencies in the signal's spectrum.
  - A mean of zero is used since the process is not expected add a DC bias.
- The AGWN is simulated using a pseudorandom number generator whose statistical profile is a normal distribution with zero-mean and a standard variance (σ^2). The variance represents the power in the noise signal.

Amplitude Modulation: MathScript Node



MathScript Node



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

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

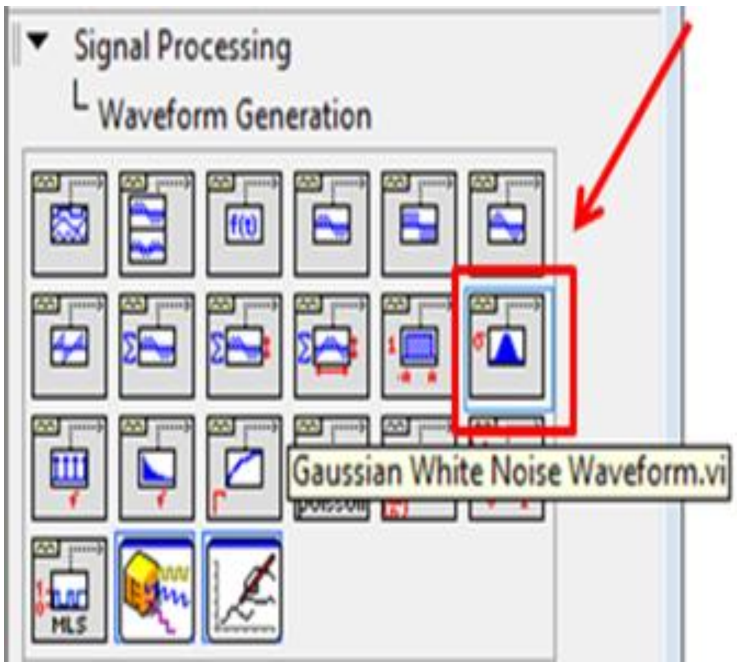
"Equations"

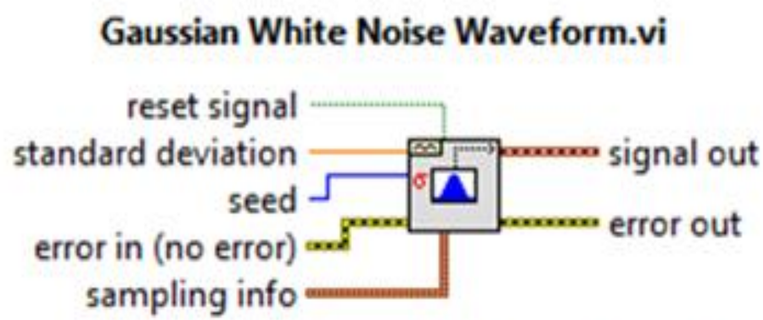
$$a = 2b - \max(d)$$
$$p = a\log(a)$$
$$s = a\,e^{2\pi p j}$$

"Text-based scripts"

a = 2*b – max (d);
p = log(a)*a;
s = a*exp(2*pi*p*j);

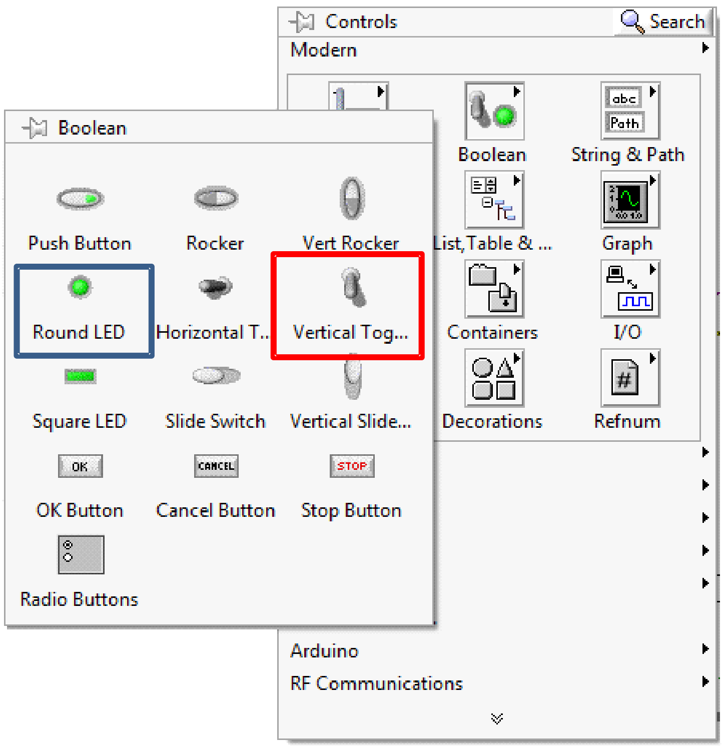
White Gaussian Noise Generation



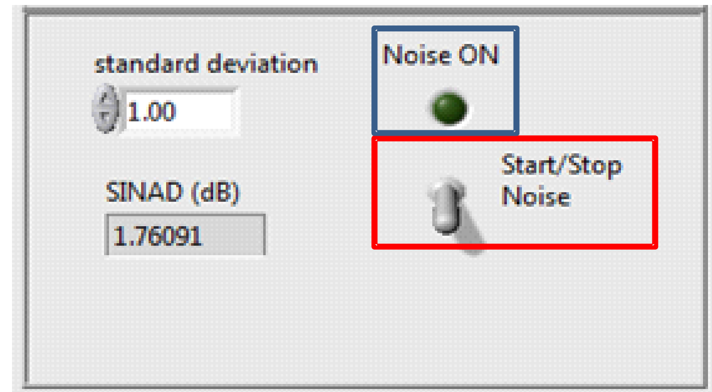


Generates a Gaussian distributed pseudorandom pattern whose statistical profile is (0,s), where s is the absolute value of the specified standard deviation.

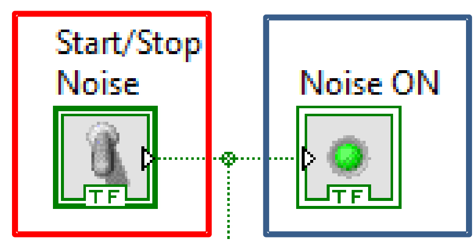
Boolean Switch and LED



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



2) Arrange the LED and switch on the front panel



To Case Statement

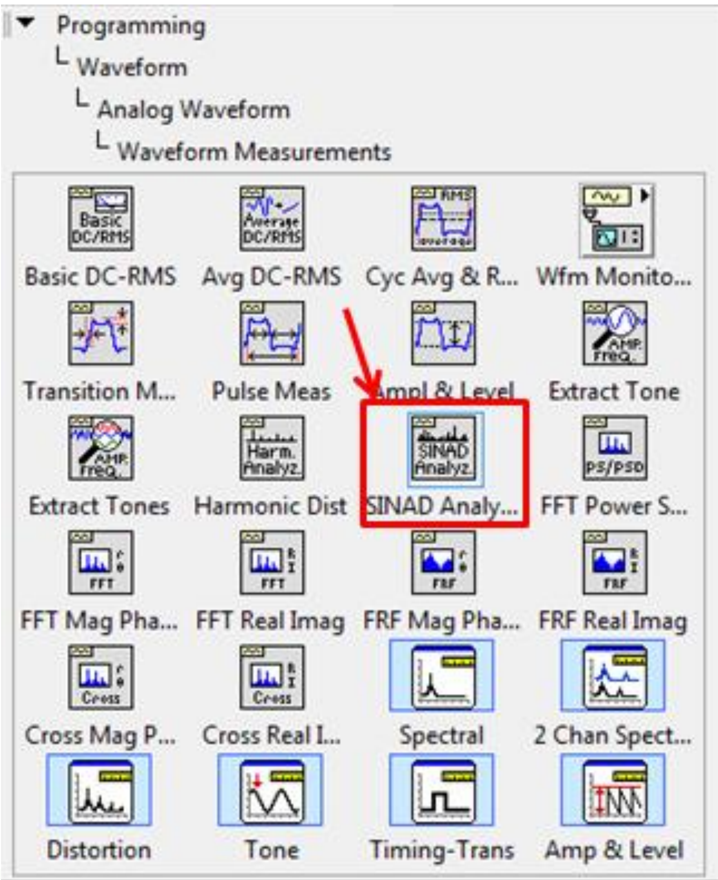
3) Arrange the LED and switch in the block diagram

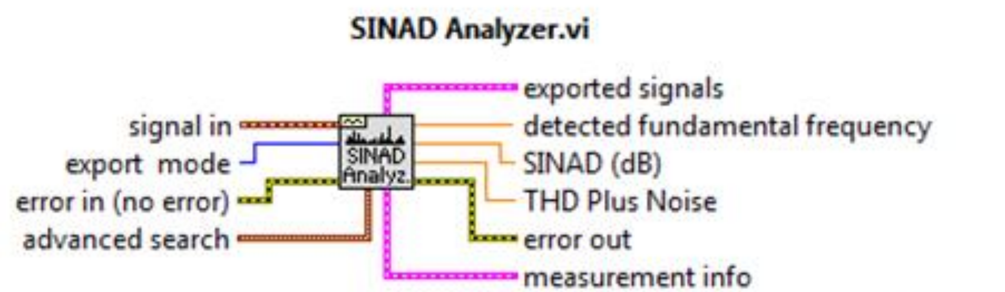
Printed on 9/1/2014

istec Ibero-American Science & Technology Education Consortium

GINET Global Innovation Network for Entrepreneurship and Technology

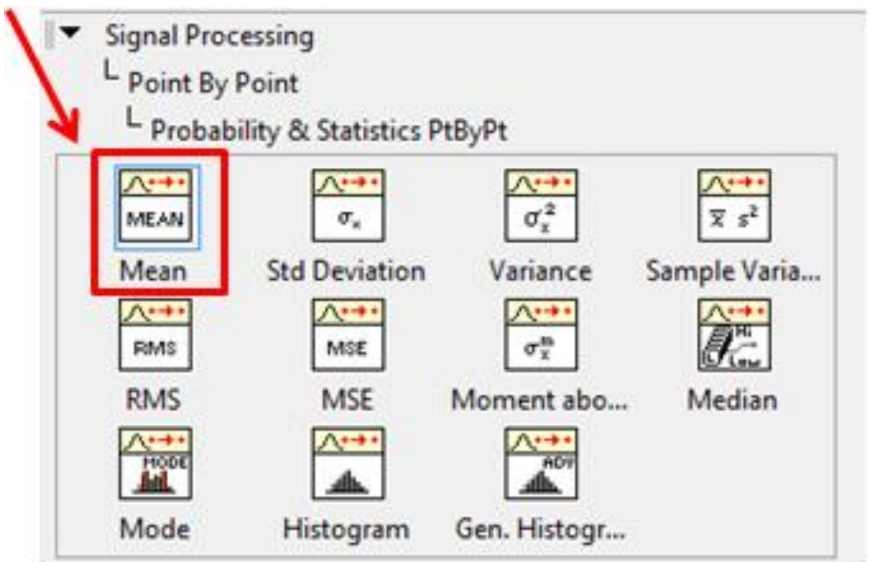
Signal to Noise & Distortion Ratio Analysis

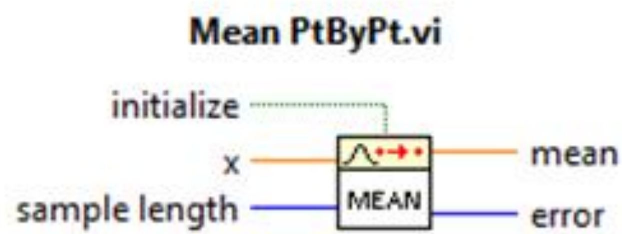




Takes a signal in and performs a full Signal in Noise and Distortion (SINAD) analysis, including measuring the fundamental frequency tone and returning the fundamental frequency and SINAD level in dB. Wire data to the **signal in** input to determine the polymorphic instance to use or manually select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

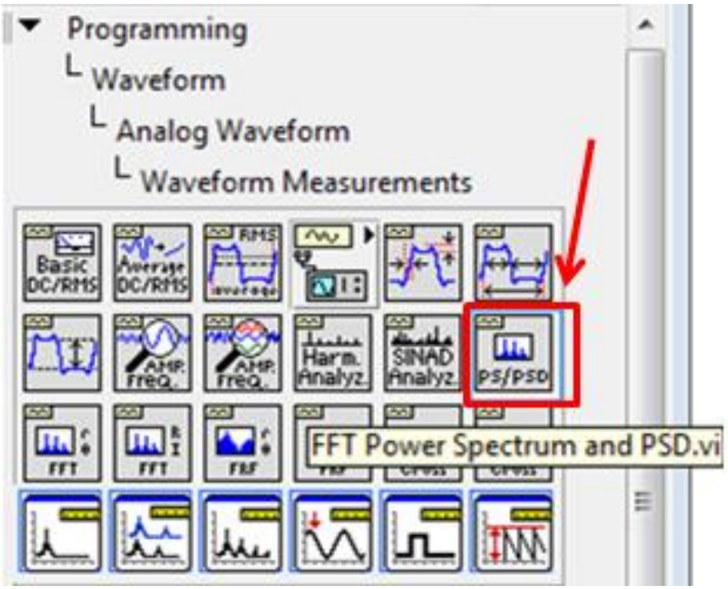
Find Point by Point Mean

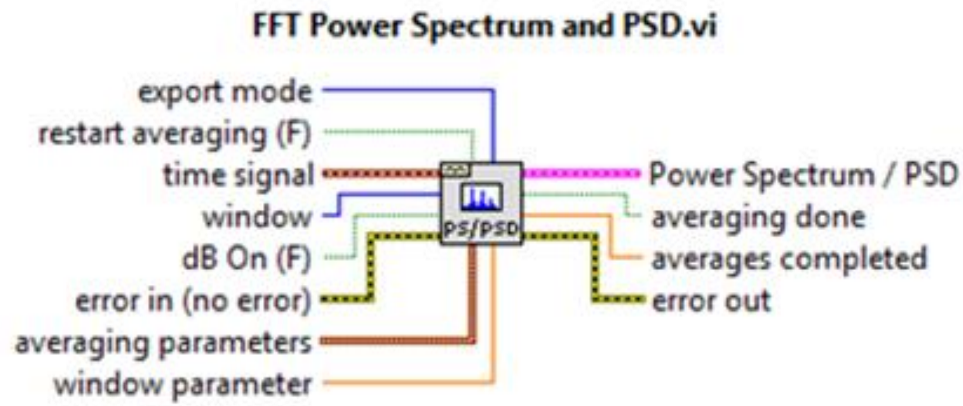




Computes the **mean**, or average, of the values in the set of input data points specified by **sample length**.

Plot Power Spectrum





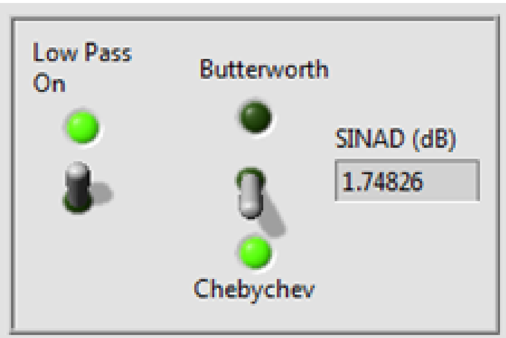
Computes the averaged auto power spectrum of **time signal**. Wire data to the **time signal** input to determine the polymorphic instance to use or manually select the instance.

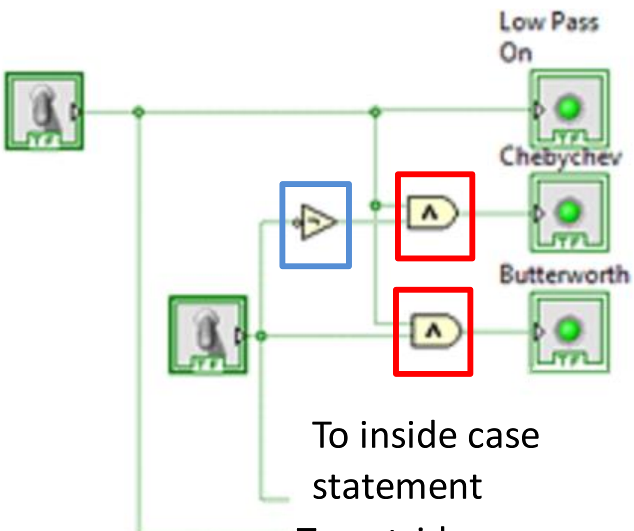
Rx Filter Selection Logic

Switch and LED Settings				
Switches		Indicator LEDs		
LPF	Filter Selector	LPF	Chebyshev	Butterworth
Off	Chebyshev	Off	On	Off
Off	Butterworth	Off	Off	On
On	Chebyshev	On	On	Off
On	Butterworth	On	Off	On

THE UNIVERSITY of NEW MEXICO

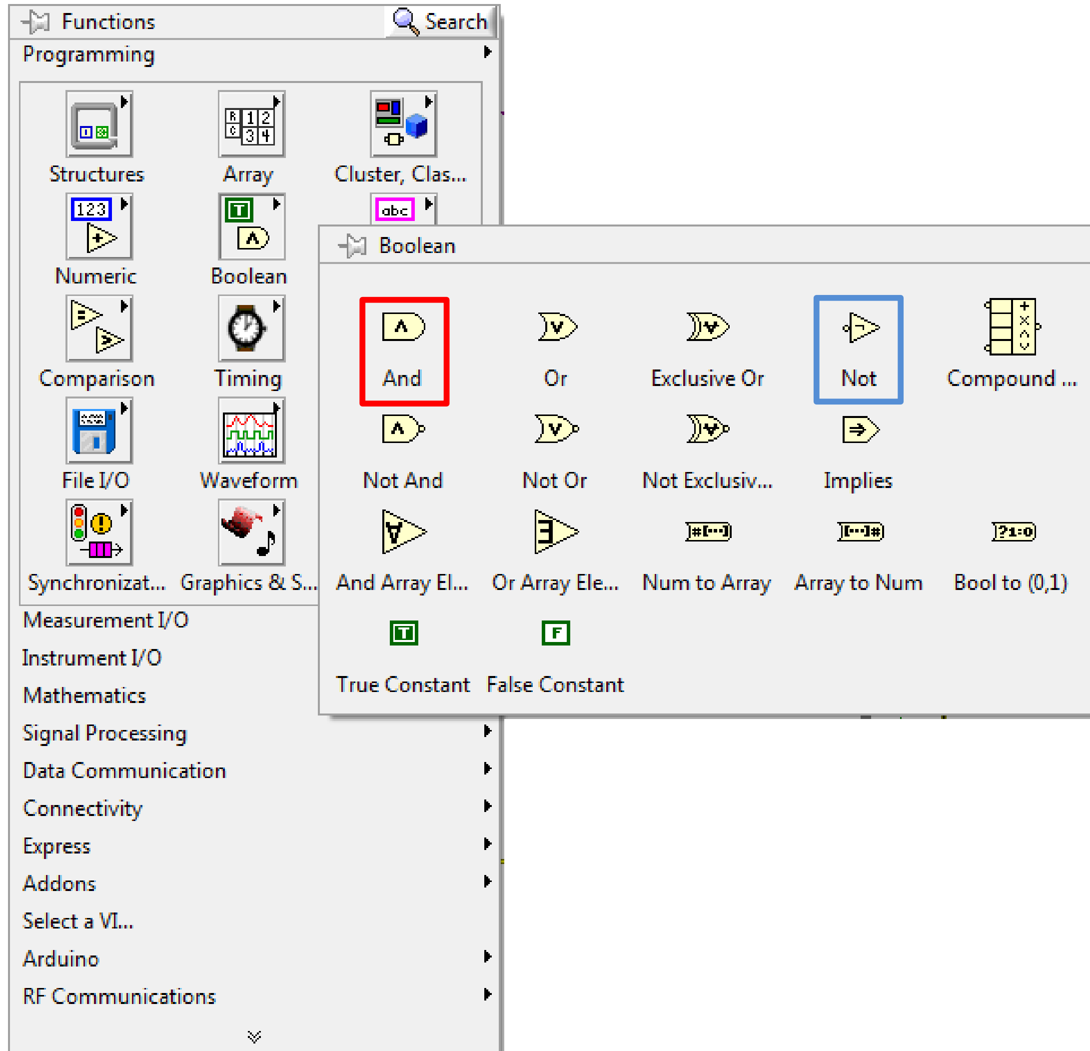
Rx Filter Selection Logic (contd.)





To inside case statement

To outside case statement



istec
Ibero-American Science & Technology Education Consortium

Entrepreneurship and Technology

THE UNIVERSITY of NEW MEXICO

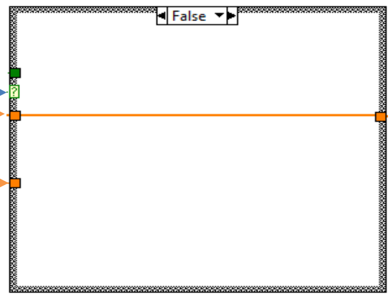
Rx Filter Selection Logic (contd.)

Outer Case Structure is FALSE

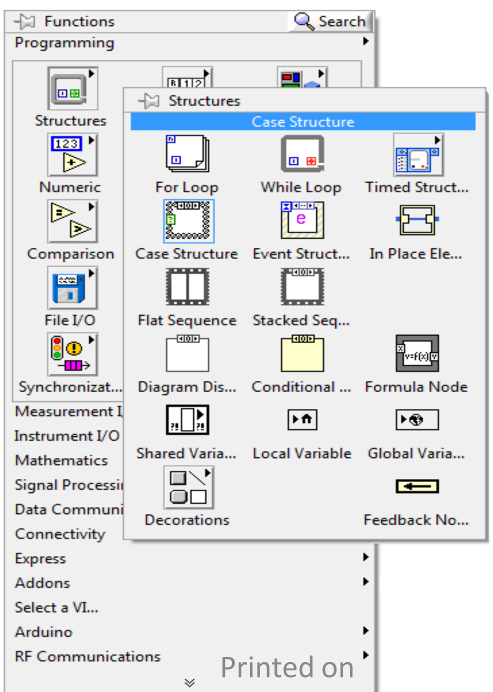
To outside case statement

From Bandpass

From Sample Info



Filtered Signal



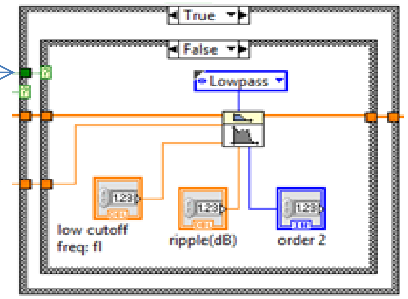
Outer Case Structure is TRUE

To inside case statement

To outside case statement

From Bandpass

From Sample Info



Filtered Signal

Inner Case Structure is FALSE (Chebyshev Filter)

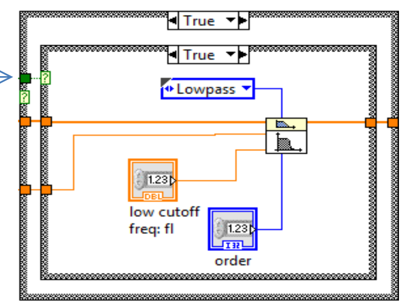
Outer Case Structure is TRUE

To inside case statement

To outside case statement

From Bandpass

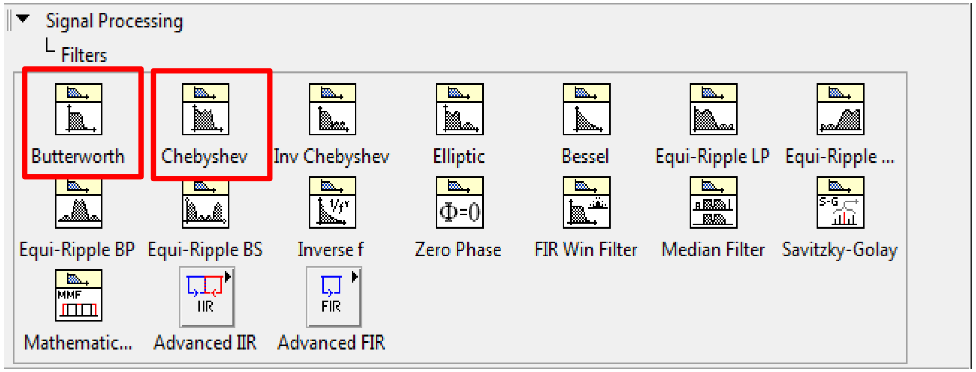
From Sample Info



Filtered Signal

Inner Case Structure is TRUE (Butterworth Filter)

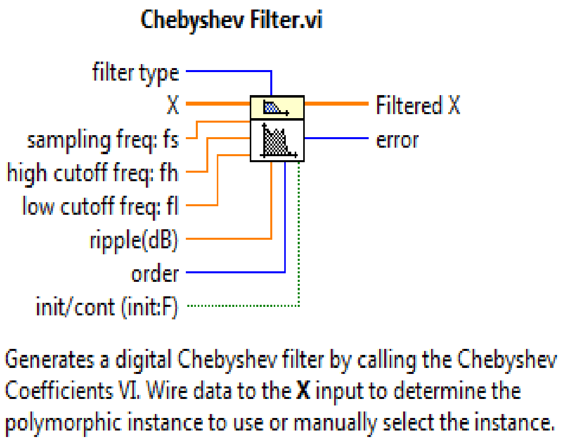
Demodulation: Filters

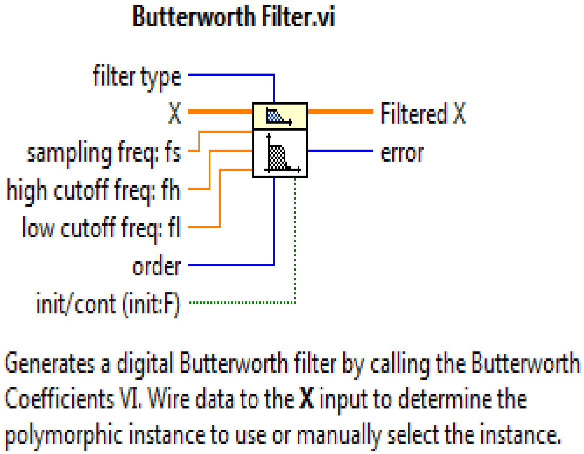


"Set filter parameters as constants"

"Chebyshev clears noise around carrier frequency"

"Butterworth implemented after full wave rectification to complete envelope detection"

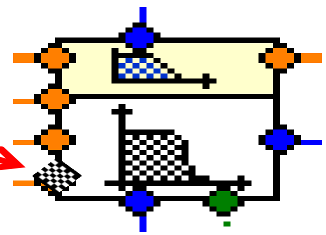




THE UNIVERSITY of NEW MEXICO

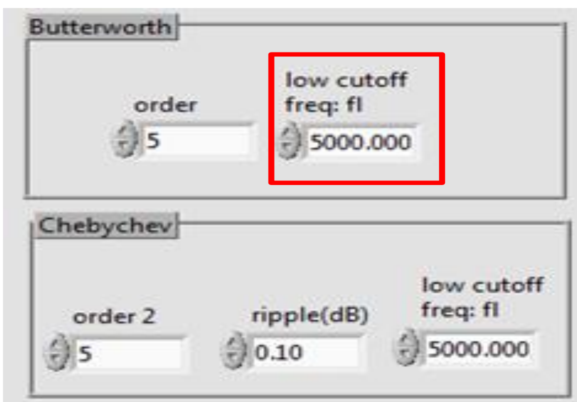
Setting Filter Parameters/ Specifications

1) Place cursor on terminal (terminal label will appear)

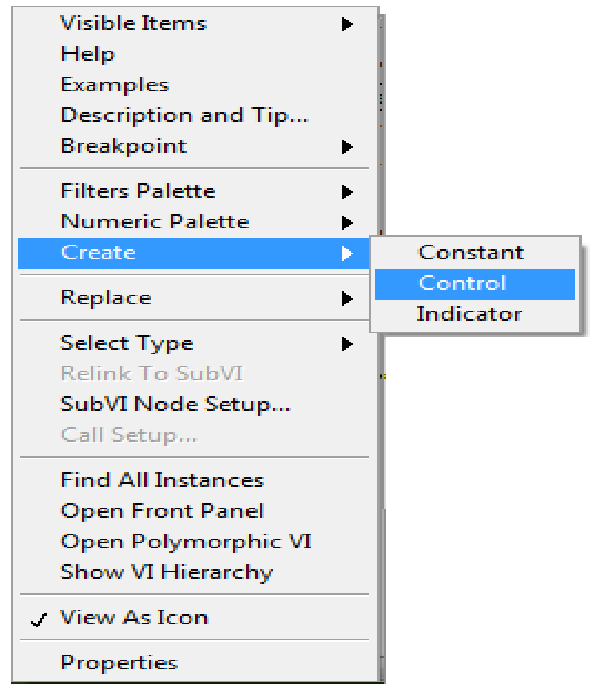


Low Cutoff Frequency fl

2) Right Click and menu will appear



3) Control on the front panel



THE UNIVERSITY *of* NEW MEXICO

Frequency Modulation with Additive Gaussian White Noise

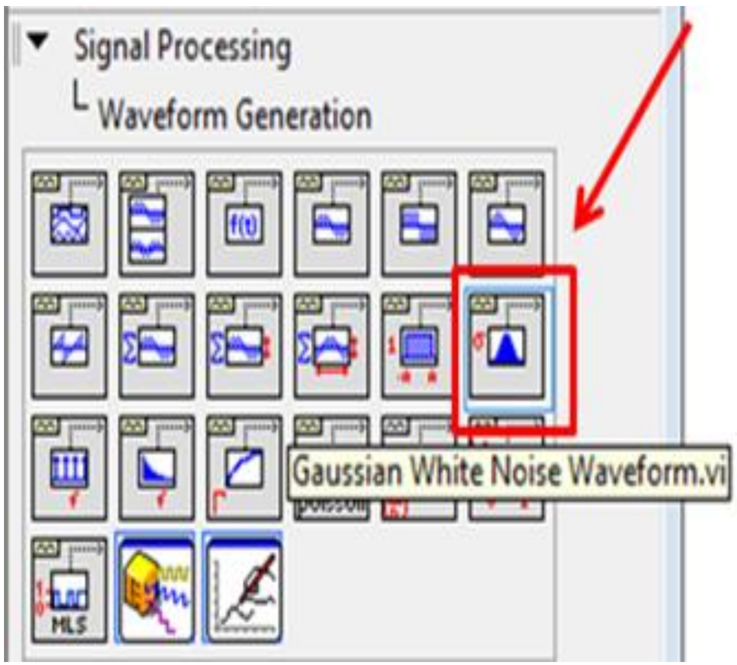
What you need to know to do the Lab…

Ibero-American Science & Technology Education Consortium

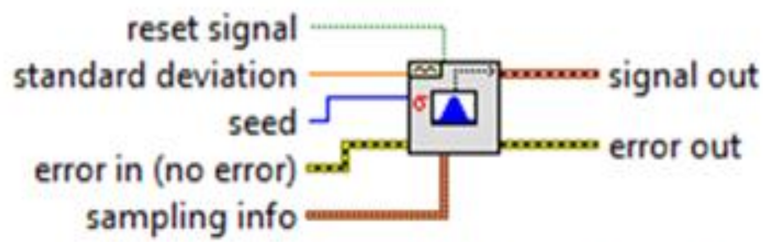
Printed on 9/1/2014

GINET Global Innovation Network for Entrepreneurship and Technology

White Gaussian Noise Generation

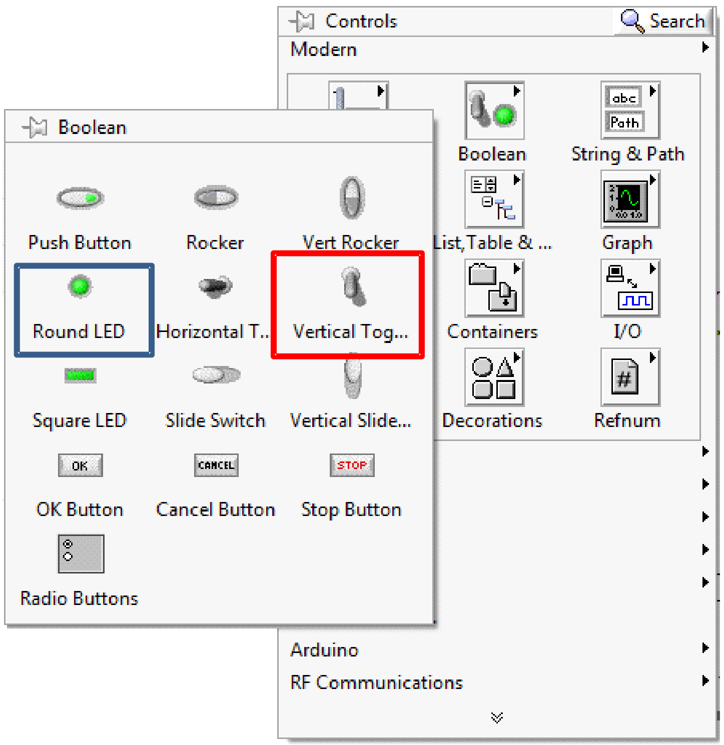


Gaussian White Noise Waveform.vi

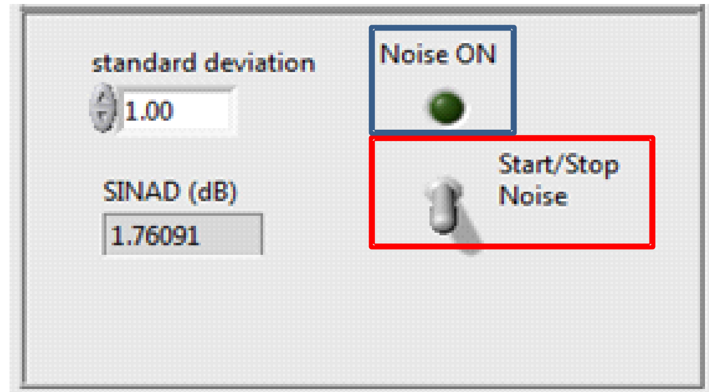


Generates a Gaussian distributed pseudorandom pattern whose statistical profile is (0,s), where s is the absolute value of the specified standard deviation.

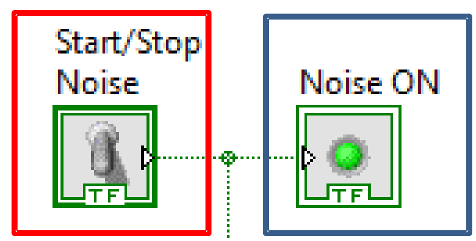
Switch and LED



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



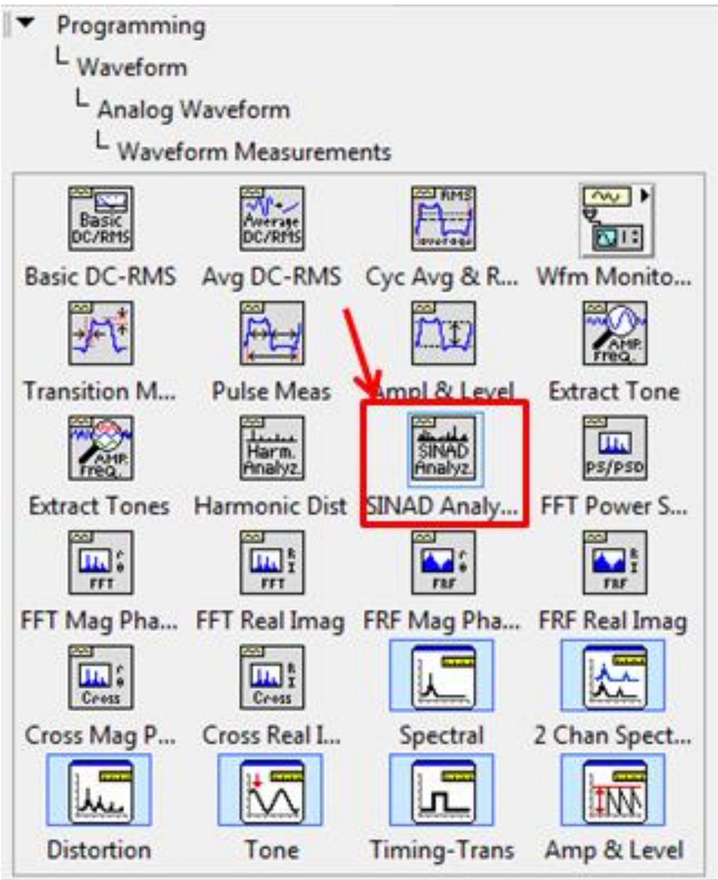
2) Arrange the LED and switch on the front panel

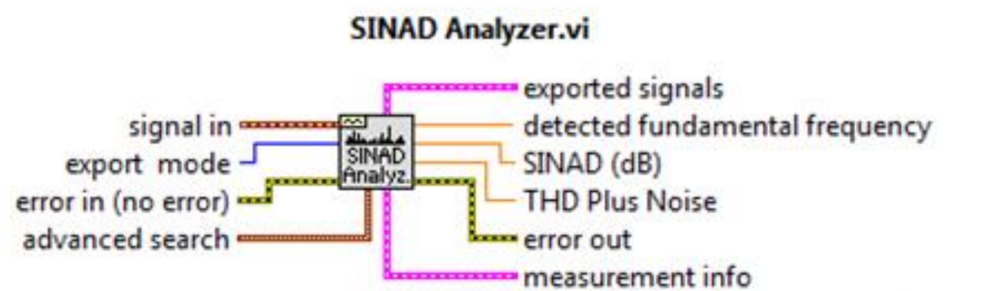


To Case Statement

3) Arrange the LED and switch in the block diagram

Signal to Noise & Distortion Ratio Analysis

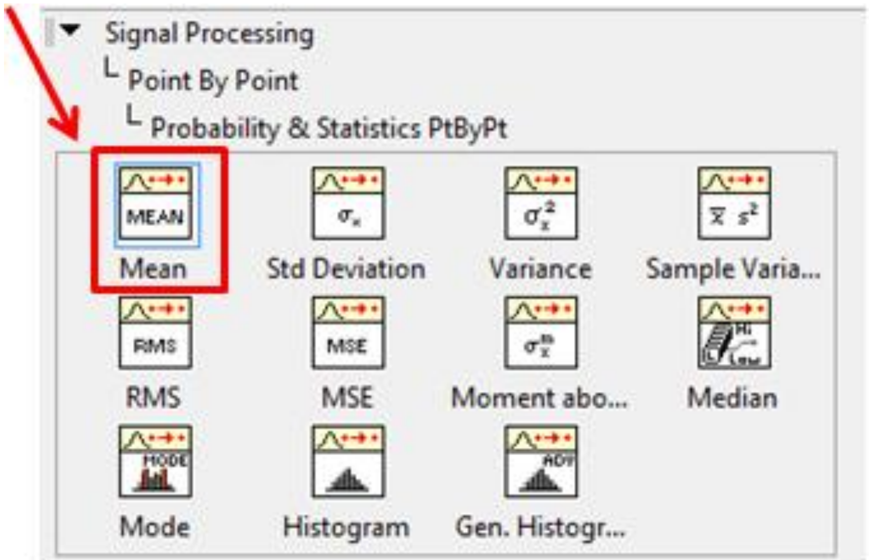




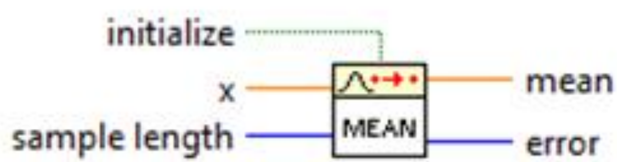
Takes a signal in and performs a full Signal in Noise and Distortion (SINAD) analysis, including measuring the fundamental frequency tone and returning the fundamental frequency and SINAD level in dB. Wire data to the **signal in** input to determine the polymorphic instance to use or manually select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

THE UNIVERSITY of NEW MEXICO

Find Point by Point Mean



Mean PtByPt.vi

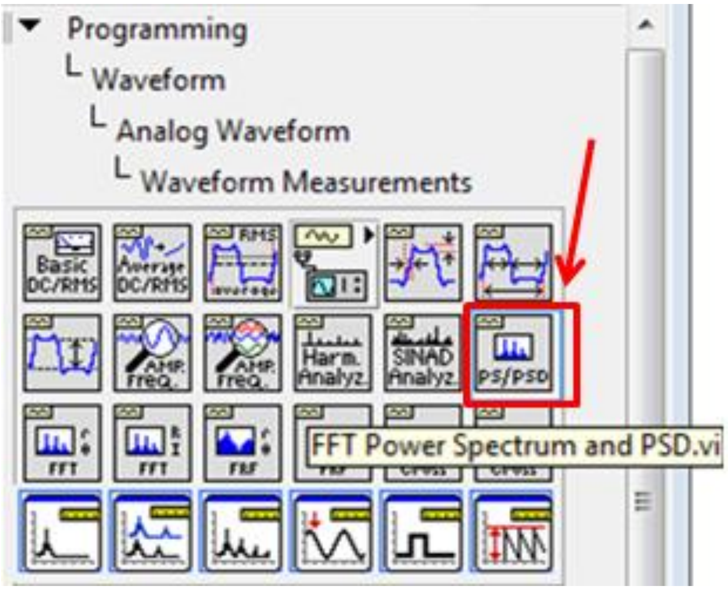


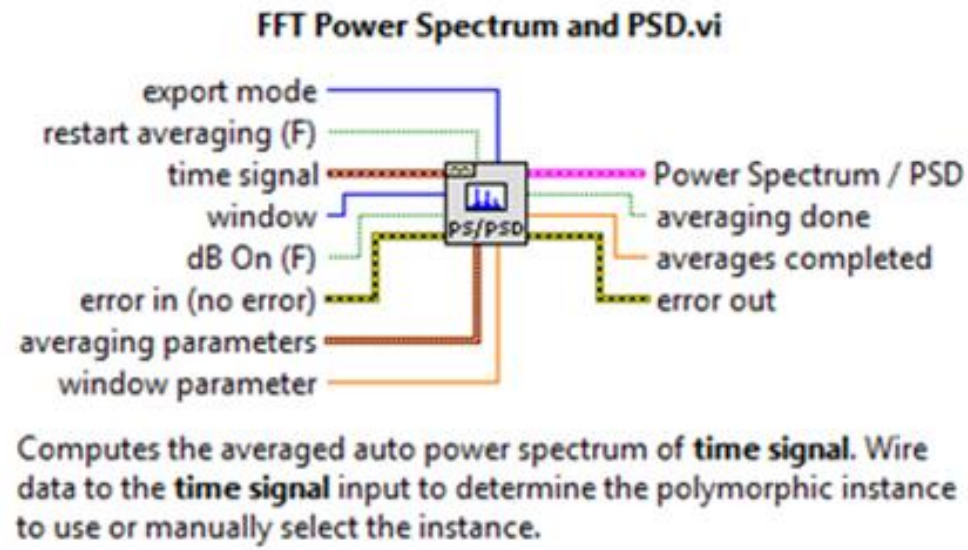
Computes the **mean**, or average, of the values in the set of input data points specified by **sample length**.

istec Ibero-American Science & Technology Education Consortium

GINET Global Innovation Network for Entrepreneurship and Technology

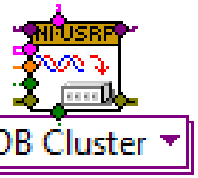
Plot Power Spectrum

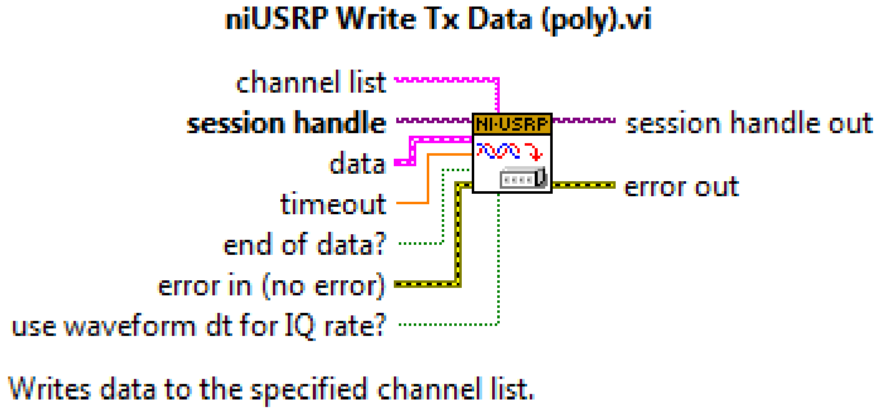




niUSRP Write Tx Data VI

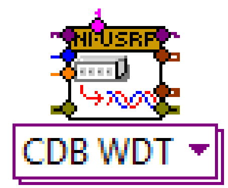
“Buffer to transmit data to receiver”



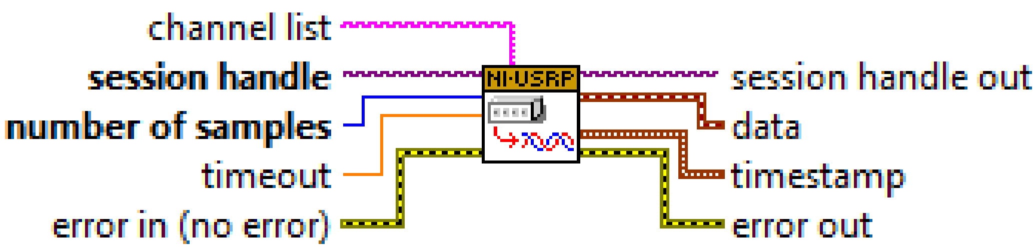


niUSRP Fetch Rx Data VI

“Buffer to receive data from transmitter”

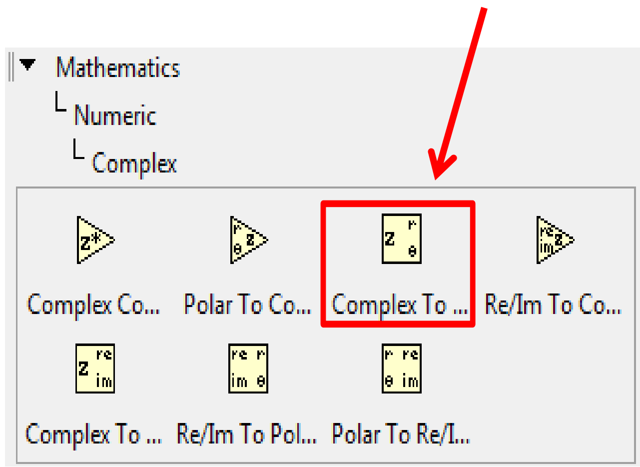


niUSRP Fetch Rx Data (poly).vi

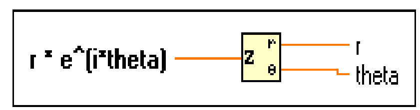


Fetches data from the specified channel list.

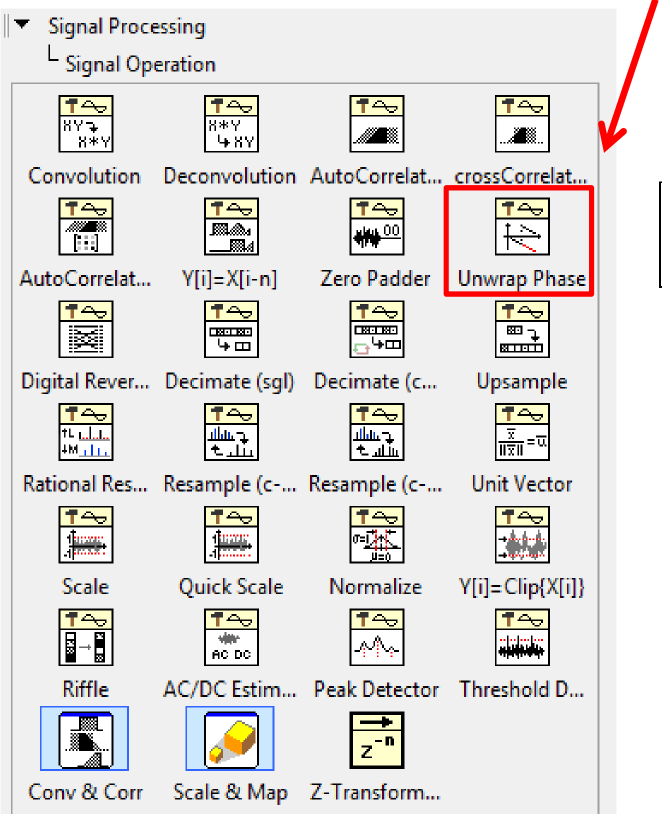
Get Angle (Phase) component by converting from Complex to Polar form



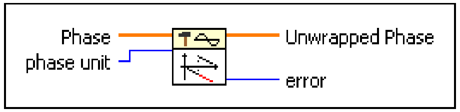
Complex To Polar Function



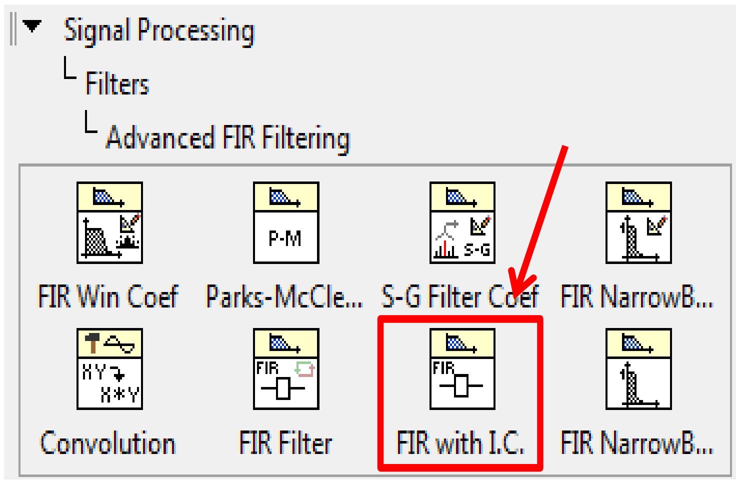
Unwrap the Phase Angle



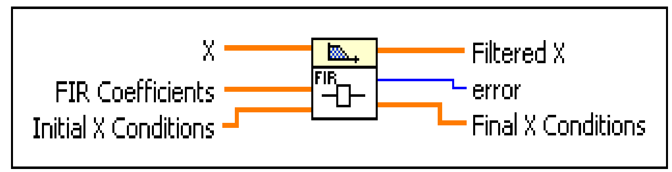
Unwrap Phase VI



Implement Difference Equation



FIR Filter with I.C. VI

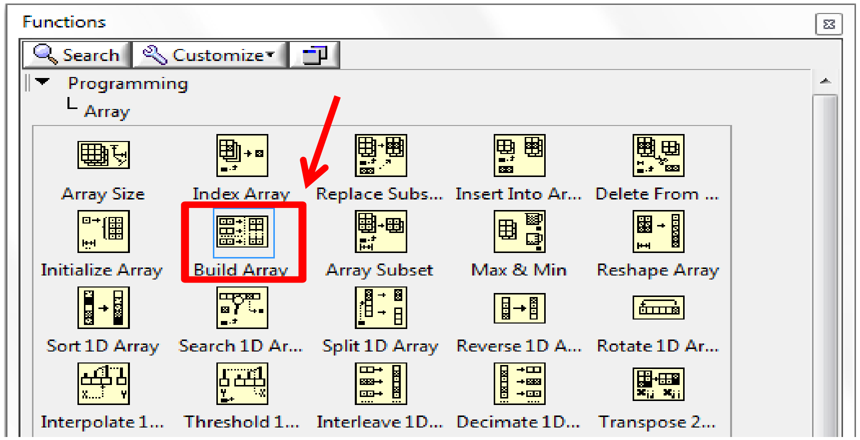


FIR Co-efficients

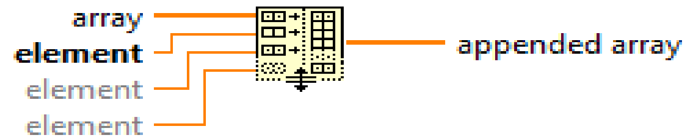
$$\frac{d\theta}{dt} = \frac{\theta[n] - \theta[n-1]}{T}$$

$$\left(\frac{1}{T}, -\frac{1}{T}\right).$$

FIR Coefficients Array

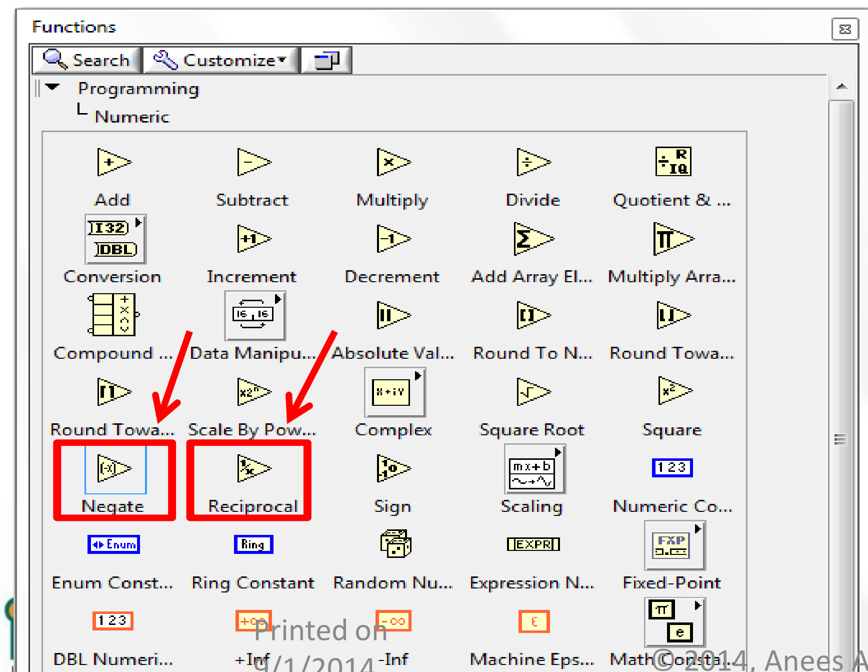


Build Array

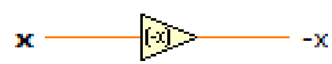


$$\left(\frac{1}{T}, -\frac{1}{T}\right).$$

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

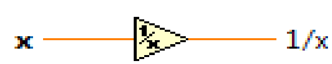


Negate



Negates the input value.

Reciprocal

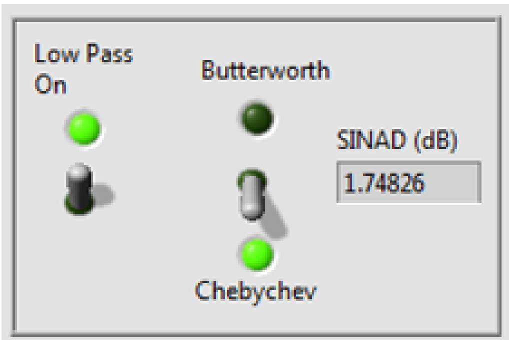


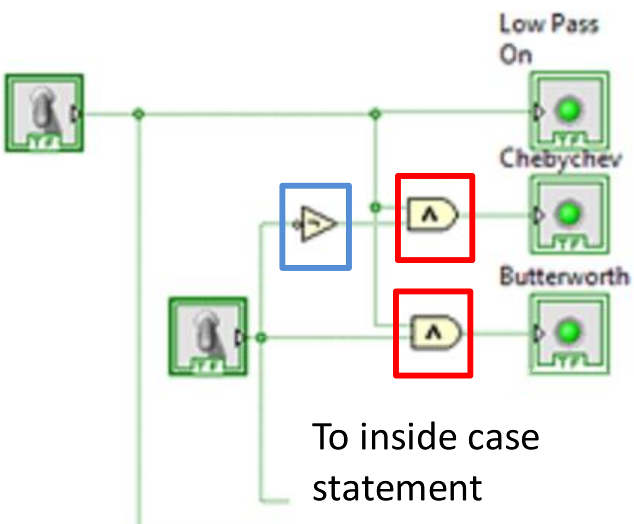
Divides 1 by the input value.

Rx Filter Selection Logic

Switch and LED Settings				
Switches		**Indicator LEDs**		
LPF	**Filter Selector**	**LPF**	**Chebyshev**	**Butterworth**
Off	*Chebyshev*	*Off*	*On*	*Off*
Off	*Butterworth*	*Off*	*Off*	*On*
On	*Chebyshev*	*On*	*On*	*Off*
On	*Butterworth*	*On*	*Off*	*On*

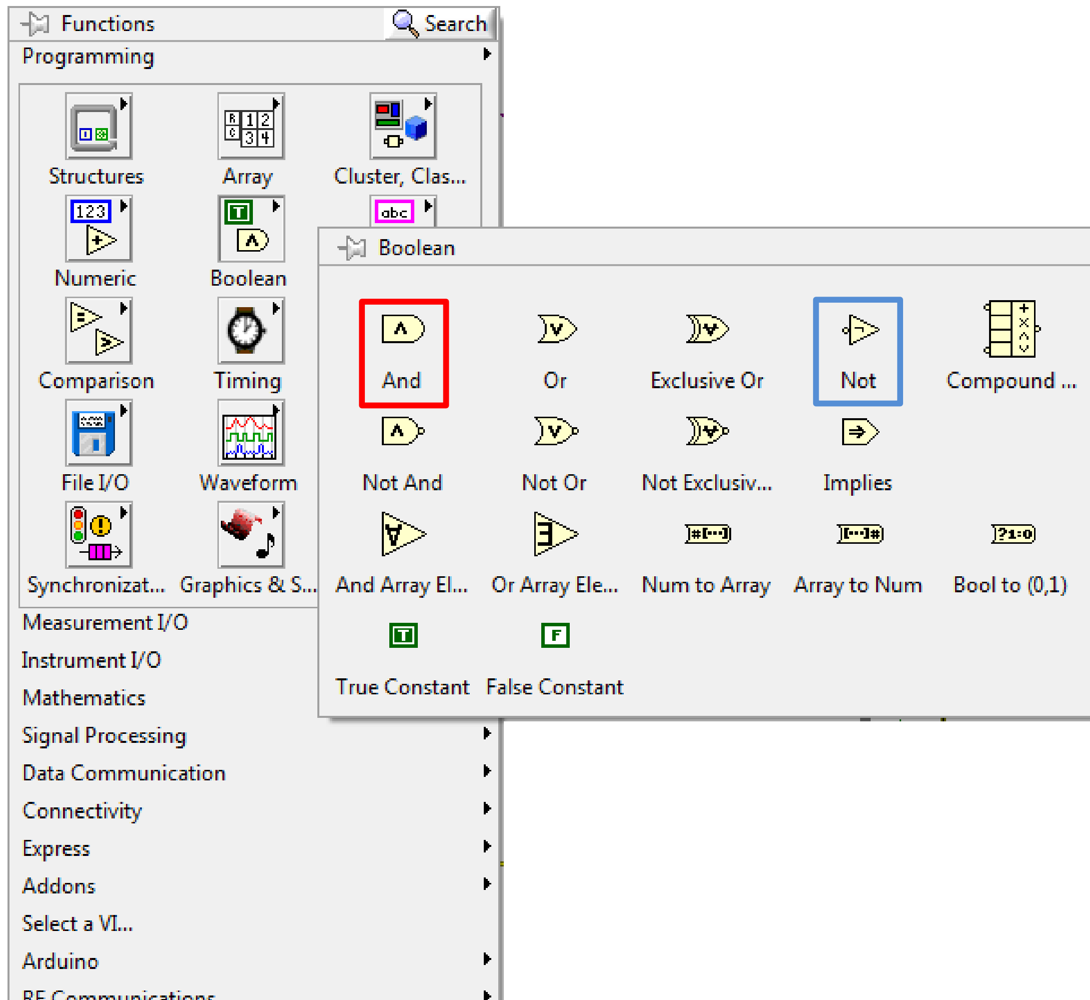
Rx Filter Selection Logic (contd.)





To inside case statement

To outside case statement



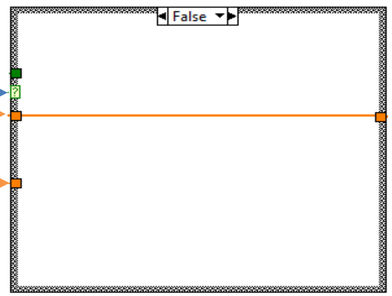
Rx Filter Selection Logic (contd.)

Outer Case Structure is FALSE

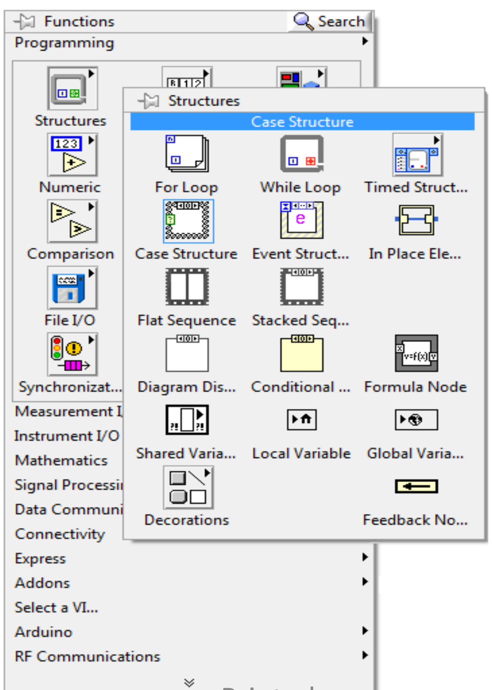
To outside case statement

From Bandpass

From Sample Info



Filtered Signal



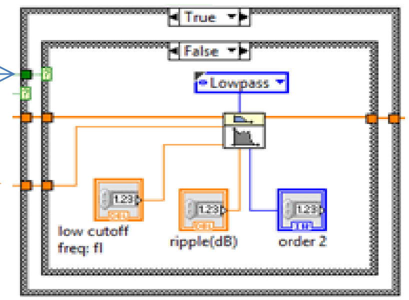
Outer Case Structure is TRUE

To inside case statement

To outside case statement

From Bandpass

From Sample Info



Filtered Signal

Inner Case Structure is FALSE (Chebyshev Filter)

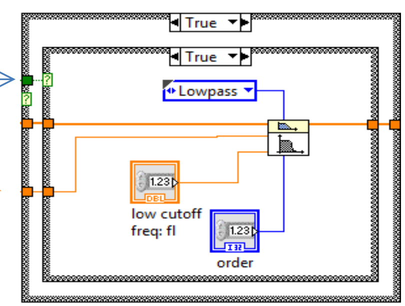
Outer Case Structure is TRUE

To inside case statement

To outside case statement

From Bandpass

From Sample Info



Filtered Signal

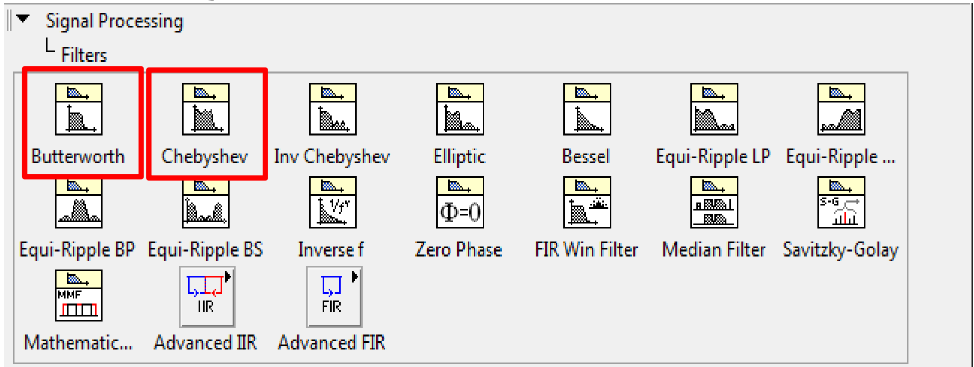
Inner Case Structure is TRUE (Butterworth Filter)

Printed on 9/1/2014

Ibero-American Science & Technology Education Consortium

GINET Global Innovation Network for Entrepreneurship and Technology

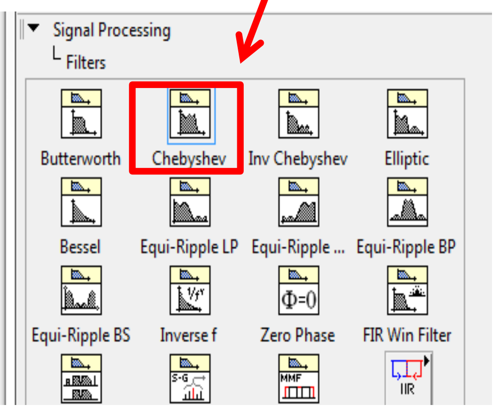
Demodulation: Filters

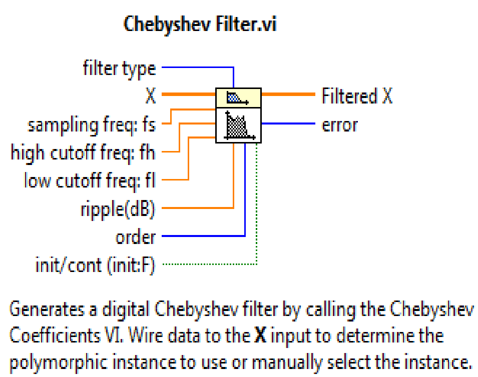


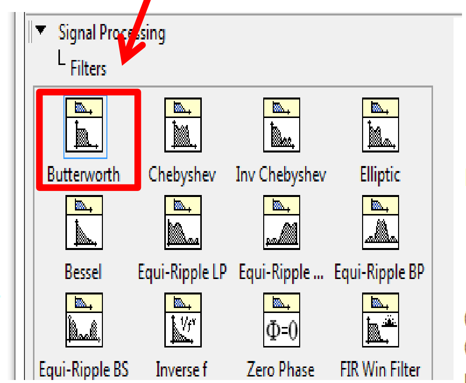
"Set filter parameters as constants"

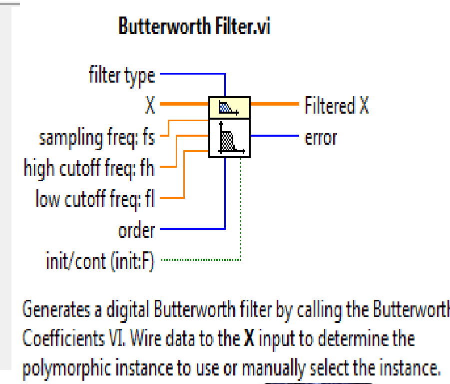
"Chebyshev clears noise around carrier frequency"

"Butterworth implemented after full wave rectification to complete envelope detection"





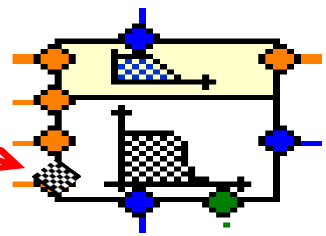




THE UNIVERSITY of NEW MEXICO

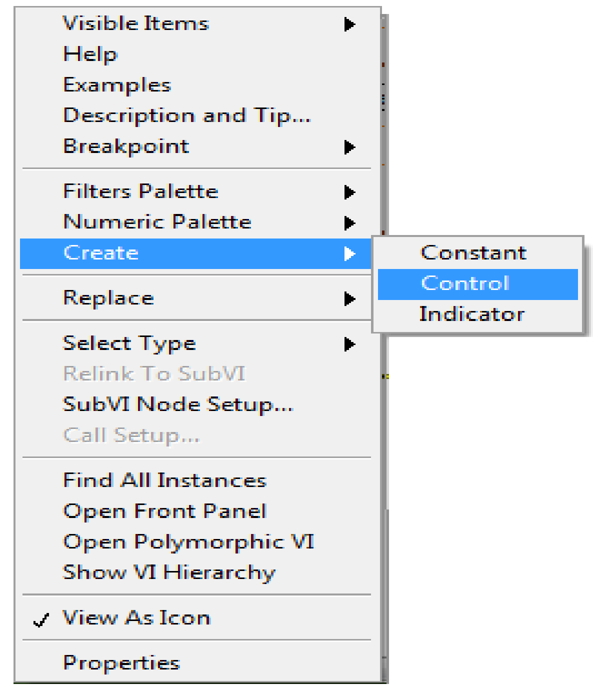
Setting Filter Parameters/ Specifications

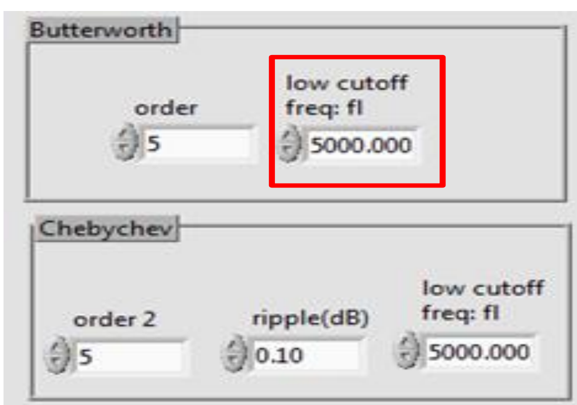
1) Place cursor on terminal (terminal label will appear)



Low Cutoff Frequency fl

2) Right Click and menu will appear



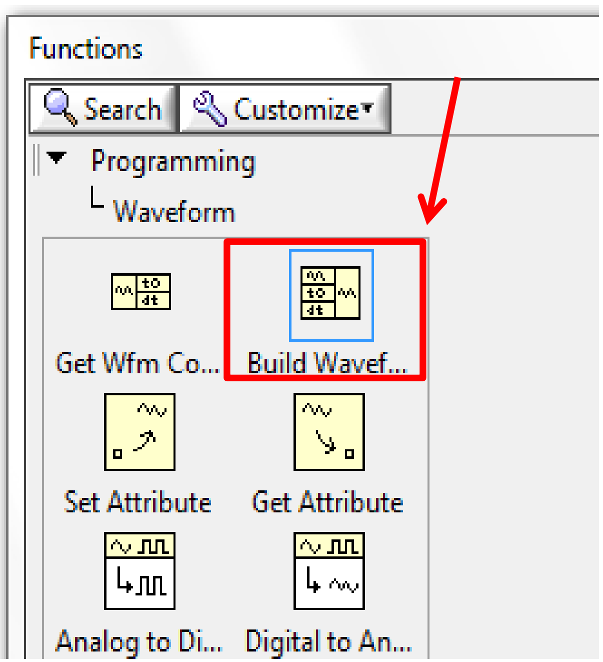


3) Control on the front panel

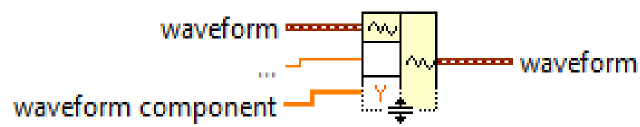
istec Ibero-American Science & Technology Education Consortium

GINET Global Innovation Network for Entrepreneurship and Technology

Build Waveform VI



Build Waveform

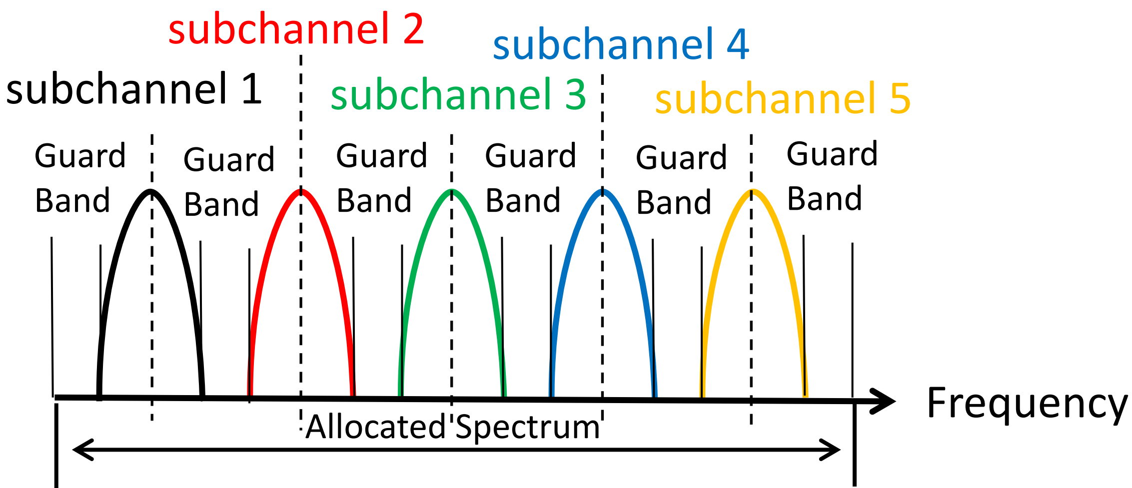


Builds an analog waveform or modifies an existing waveform. If you do not wire the **waveform** input, the function creates a new waveform based on the components you wire. If you wire the **waveform** input, the function modifies the waveform based on the components you wire.

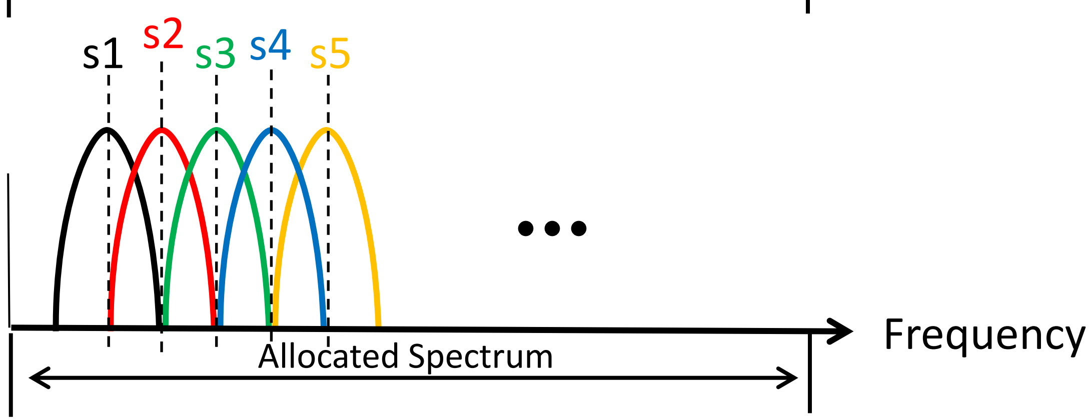
Frequency Domain Multiplexing

What you need to know to do the Lab

Allocating Spectrum to Subchannels

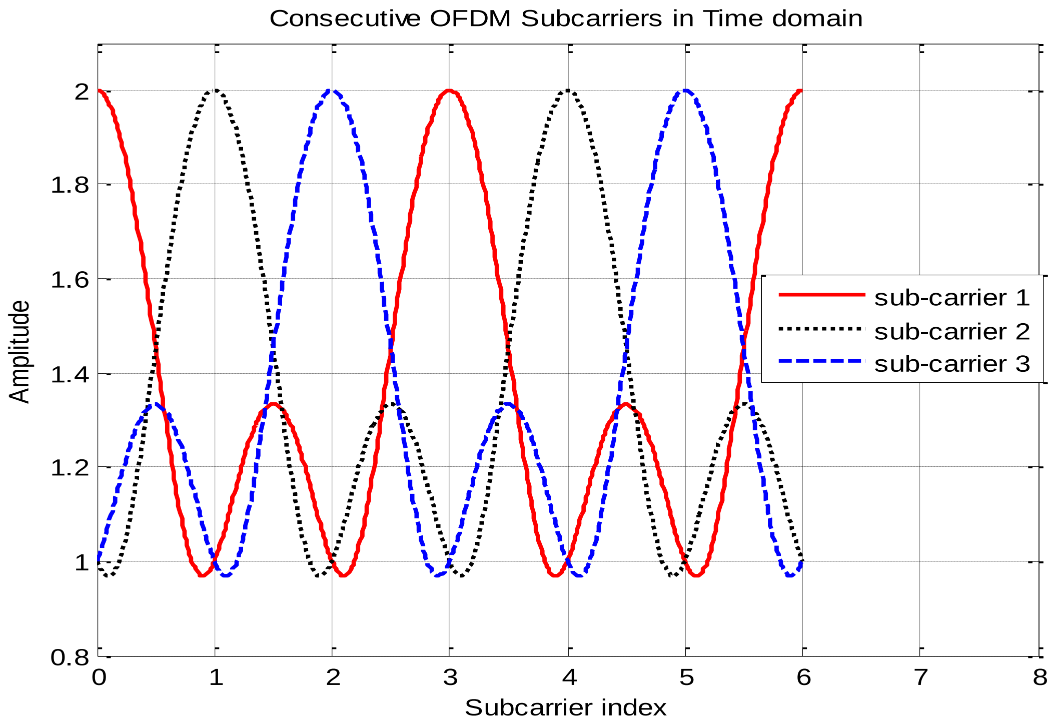


Conventional Multicarrier Modulation (FMDA)



Orthogonal Frequency Division Multiplexing (OFDM)

FDM Concepts



In this experiment you will be using two frequencies or sub carriers.

You will build a transmitter and receiver VI and will examine the affects of inter-carrier or subchannel interference.

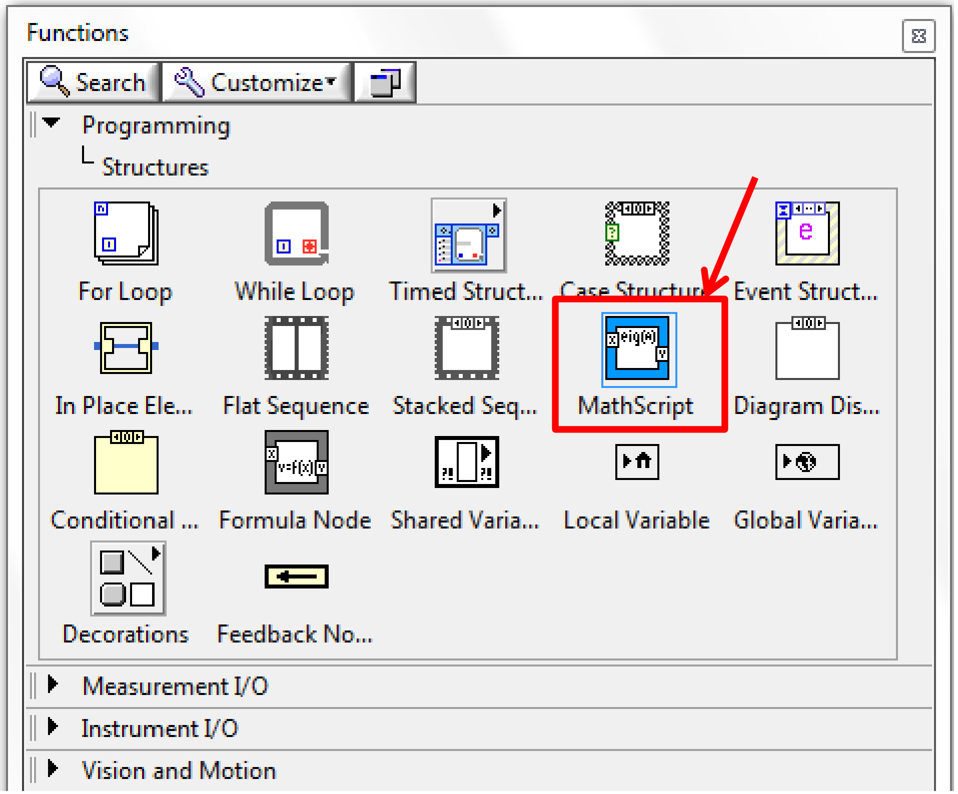
PRE-LAB Tasks

- A template for the transmitter has been provided in the file FDM_Tx_Template.vi. To complete the transmitter you will be asked to perform two tasks:
  - Create a sub-vi that modulates a message using Amplitude Modulation.
  - Update the transmitter template to combine the modulated messages to form the OFDM signal.
- A template for the receiver is also provided, FDM_Rx_Template.vi. To complete the lab, you will need to
  - Design a band pass filter to isolate each message signal.
  - Create an envelope detector similar to the one designed in Amplitude Modulation Lab.

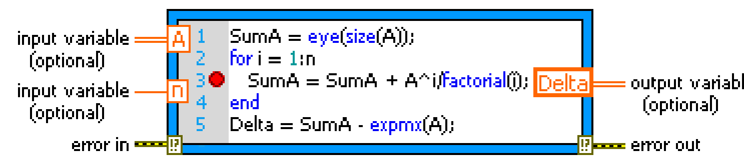
AM_on_Sub-carrier subVI

(AM modulation Review)

Modulation: MathScript Node



MathScript Node



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

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

"Equations"

$$a = 2b - \max(d)$$
$$p = a\log(a)$$
$$s = a\,e^{2\pi p j}$$

"Text-based scripts"

```
a = 2*b – max (d);
p = log(a)*a;
s = a*exp(2*pi*p*j);
```


Ibero-American Science & Technology Education Consortium

GINET Global Innovation Network for Entrepreneurship and Technology

THE UNIVERSITY of NEW MEXICO

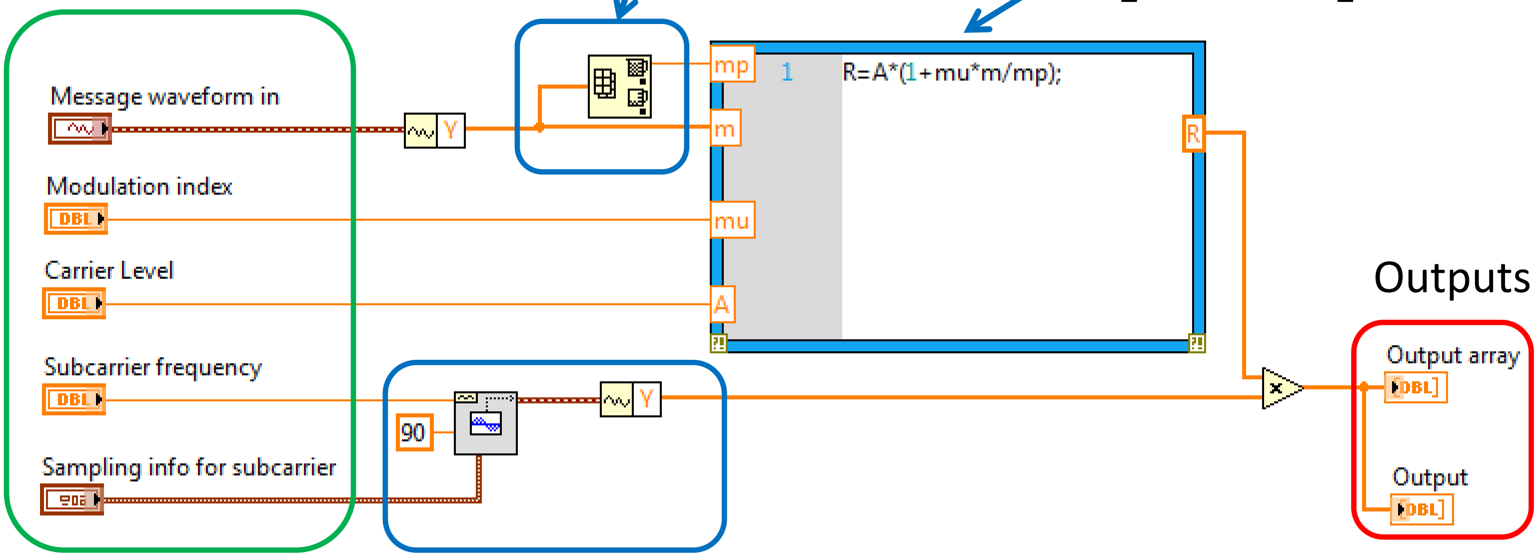
SubVI Overview

$$m_p(t) = \frac{m(t)}{\max\left(\left|m(t)\right|\right)}$$

$$A\left[1+\mu_1\frac{m(t)}{m_p}\right]$$

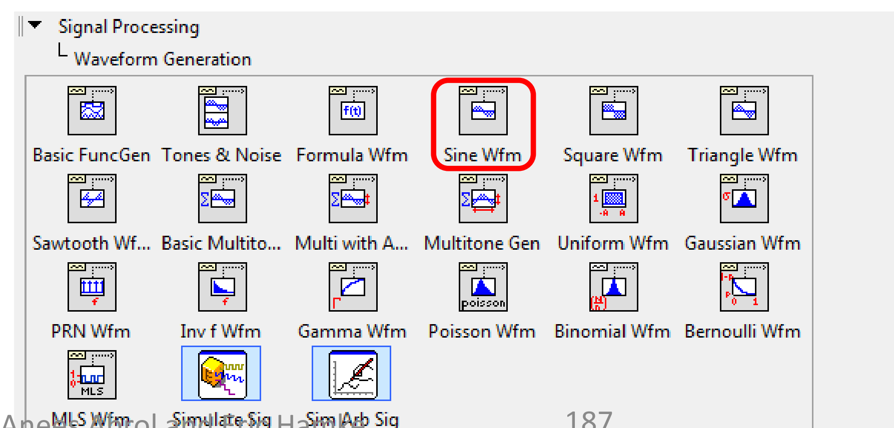
Inputs

Outputs



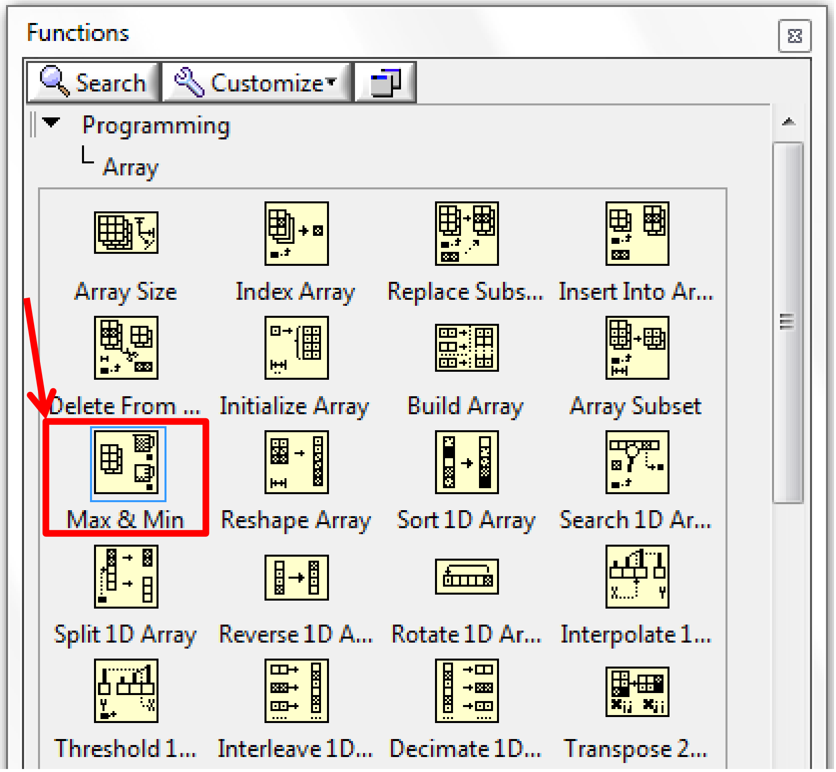
Generate sub-carrier

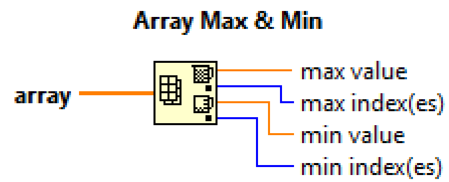
$$\sin(\theta+90°) = \cos(\theta)$$



istec
Ibero-American Science & Technology Education Consortium

Array Max & Min VI

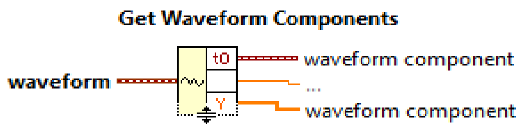




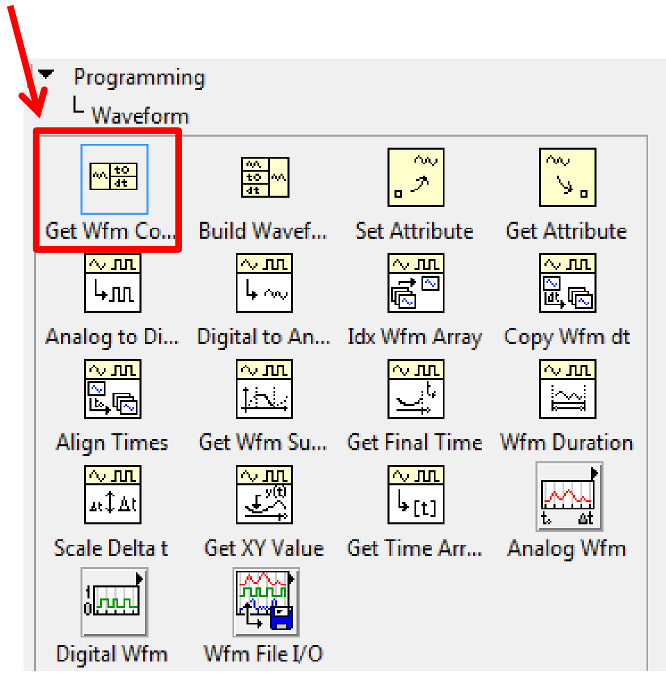
Returns the maximum and minimum values found in **array**, along with the indexes for each value.

THE UNIVERSITY of NEW MEXICO

Get Waveform Components VI

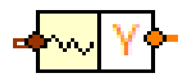


Returns the analog waveform you specify. You specify components by clicking on the center of the output terminal and selecting the component you want.

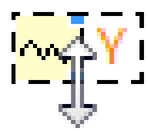


"Waveform attribute selection"

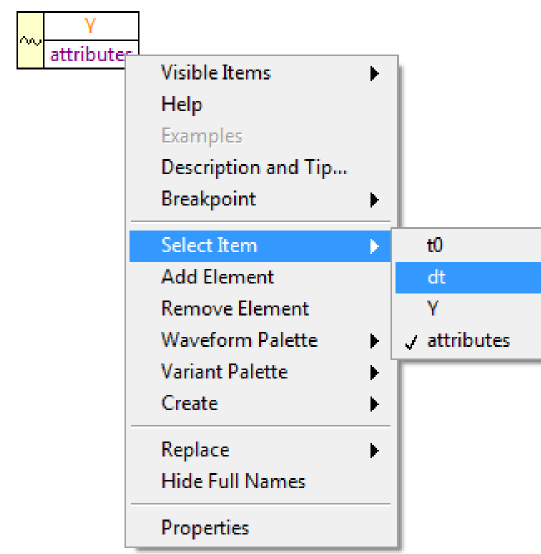
1. Select, hold and drop VI



2. Click on bottom line, hold and extend



3. Right-click on attributes, scroll to "Select Item" and pick the attribute.



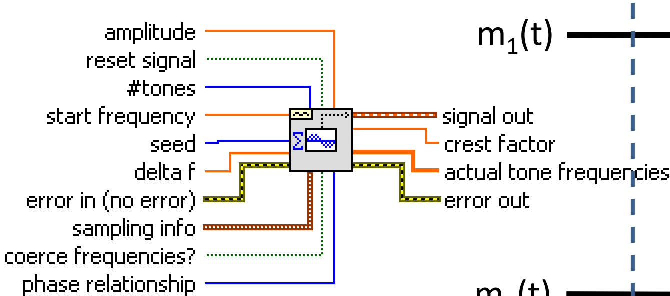
istec Ibero-American Science & Technology Education Consortium

Printed on 9/1/2014

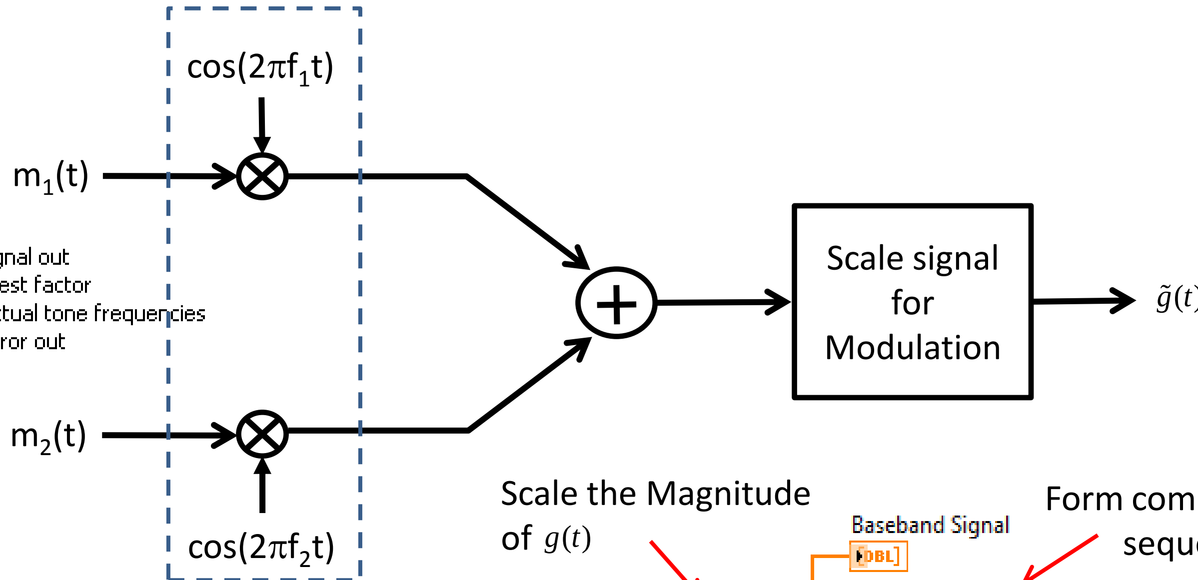
GINET Global Innovation Network for Entrepreneurship and Technology

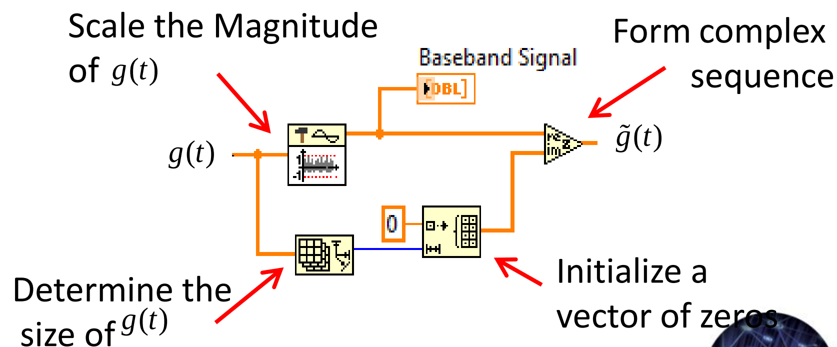
Combine the Modulated Messages

Superposition

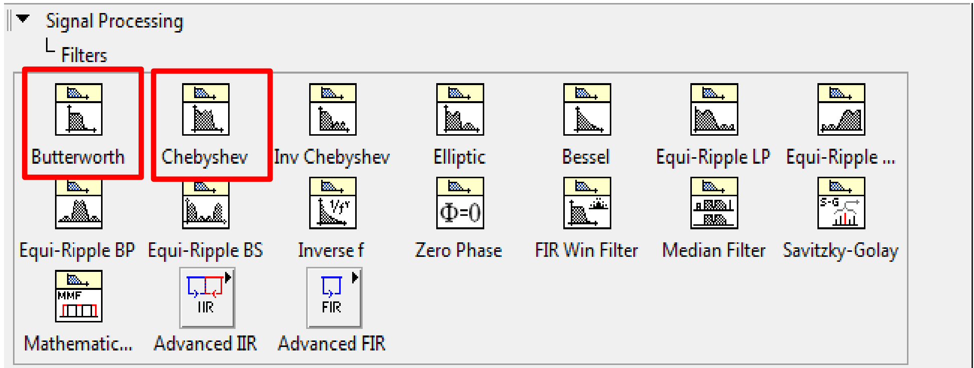


AM_on_Sub-carrier subVI





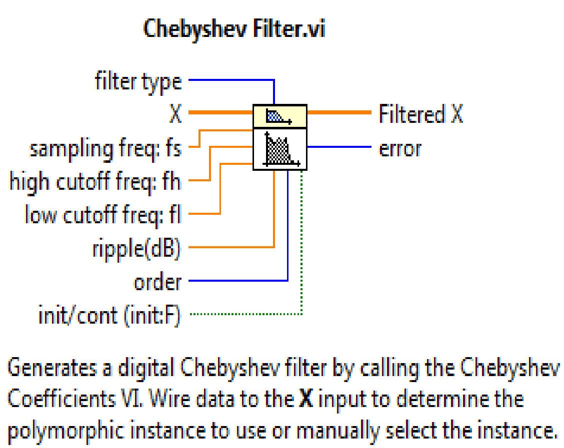
Demodulation: Filters

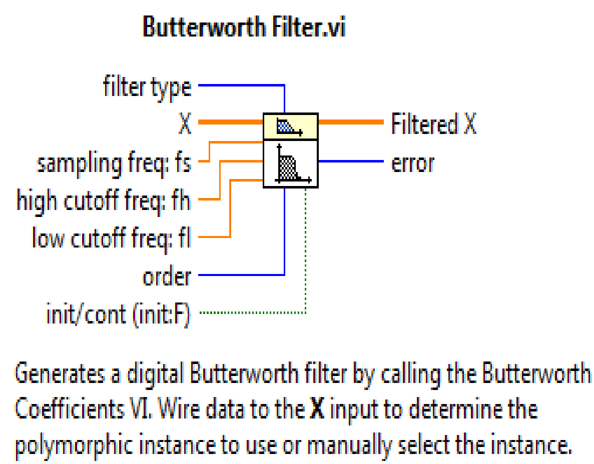


"Set filter parameters as constants"

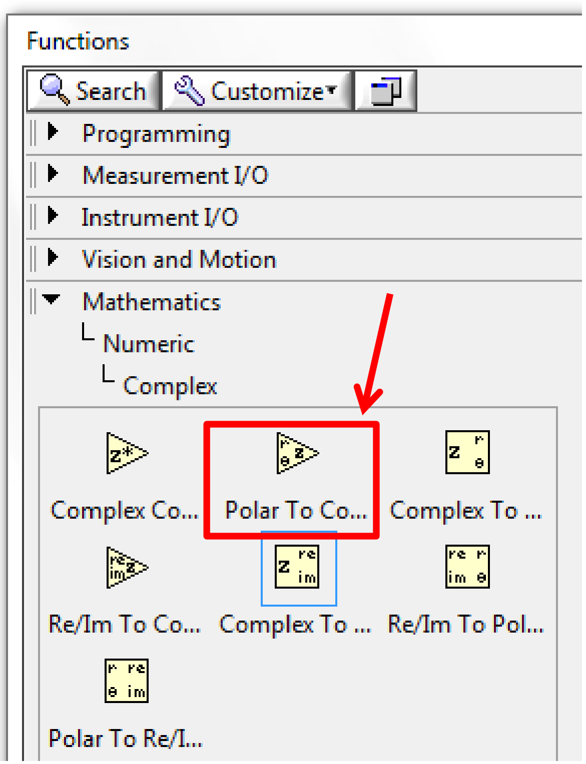
"Chebyshev clears noise around carrier frequency"

"Butterworth implemented after full wave rectification to complete envelope detection"

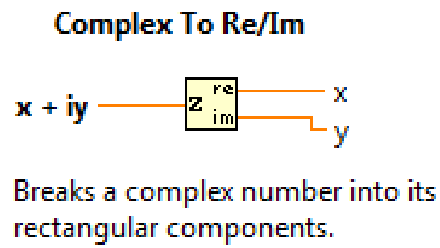




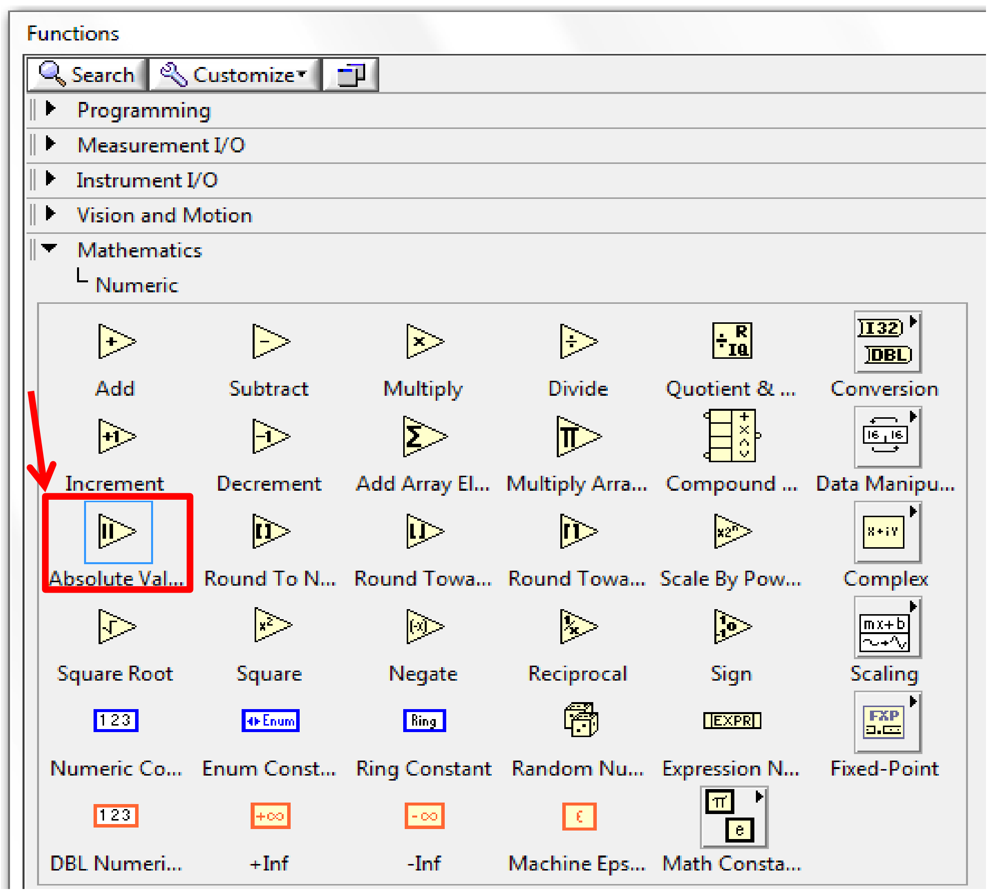
Complex to Real/Imaginary



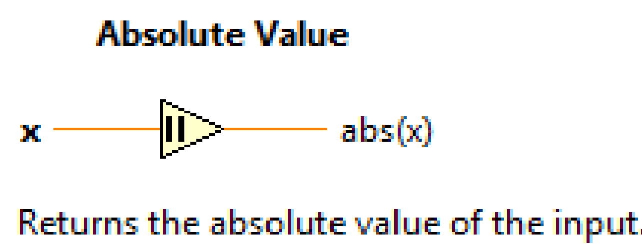
"Extract real part from complex data values"



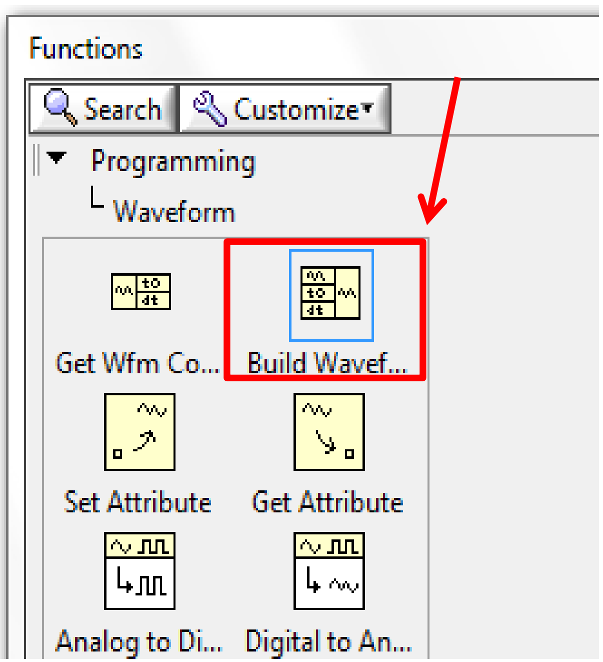
Absolute Value VI

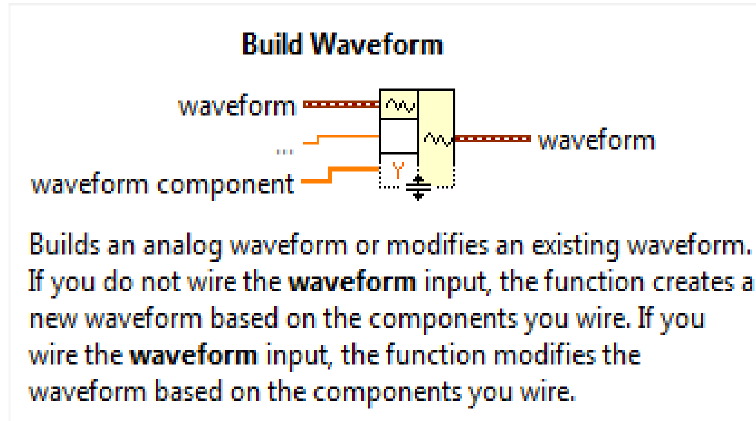


"Full-wave Rectifier"



Build Waveform VI





"Waveform attribute selection"

Same as "Get Waveform Components"

THE UNIVERSITY of NEW MEXICO

Entropy and Coding Efficiency

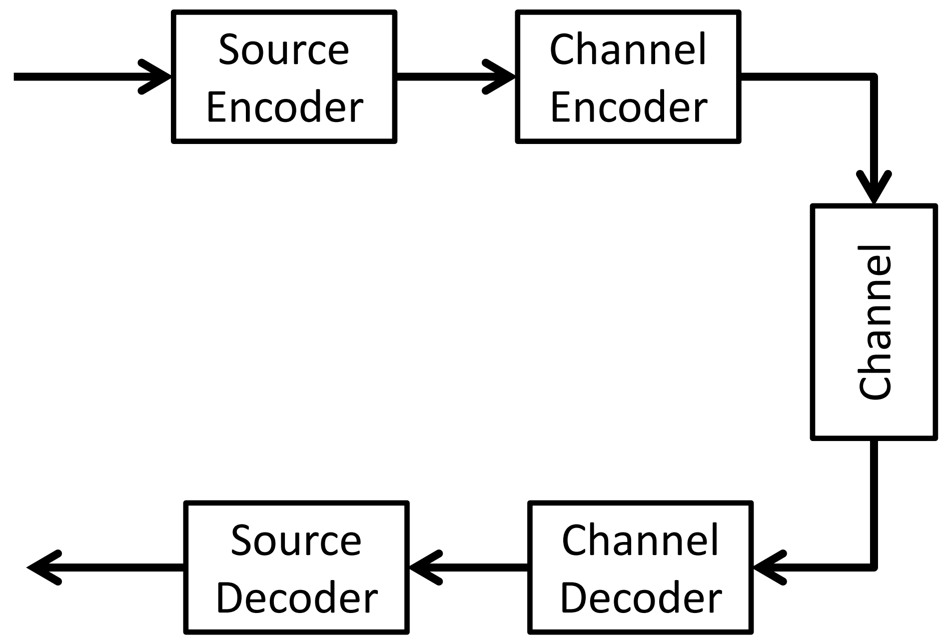
What you need to know to do the Lab…

Ibero-American Science & Technology Education Consortium

Global Innovation Network for Entrepreneurship and Technology

Digital Communication Block Diagram

- The source encoder converts the source to a binary sequence
- The channel encoder (often called includes the modulator and redundancy coding) . It processes the binary sequence for transmission over the channel.
- The channel decoder (demodulator) recreates the incoming binary sequence
- The source decoder recreates the source output.



English Language Statistics

A typical example of the number of times (relative frequency) we would expect to see the letters (symbols) appear in a random piece of English text consisting of 40,000 letters..

A typical Huffman code generated for this sample of text. The average number of bits used to transmit the symbols in the text is approximately 4.25 bits/symbol

Relative Frequency of Letters in the English Language

Letter	Relative Frequency	Letter	Relative Frequency	Letter	Relative Frequency
a	3256	j	60	s	2524
b	596	k	308	t	3612
c	1108	l	1604	u	1100
d	1696	m	960	v	392
e	5184	n	2692	w	940
f	888	o	2992	x	60
g	804	p	768	y	788
h	2432	q	36	z	28
i	2780	r	2388	--	--

Huffman Code Letters in the English Language

Letter	Huffman Code	Letter	Huffman Code	Letter	Huffman Code
e	100	d	11111	p	110001
t	000	l	11110	b	110000
a	1110	c	01001	v	001000
o	1101	u	01000	k	0010011
i	1011	m	00111	j	001001011
n	1010	w	00110	x	001001010
s	0111	f	00101	q	001001001
h	0110	g	110011	z	001001000
r	0101	y	110010	--	--

Pulling the Data Together

Table XLII -Relative Frequency of Letters in the English Language

Letter	Length	Relative Frequency	Letter	Length	Relative Frequency	Letter	Length	Relative Frequency
a	4	0.0814	**j**	9	0.0401	**s**	4	0.0275
b	6	0.0149	**k**	7	0.0240	**t**	3	0.0098
c	5	0.0277	**l**	5	0.0673	**u**	5	0.0235
d	5	0.0424	**m**	5	0.0748	**v**	6	0.0015
e	3	0.1296	**n**	4	0.0192	**w**	5	0.0197
f	5	0.0222	**o**	4	0.0009	**x**	9	0.0007
g	6	0.0201	**p**	6	0.0597	**y**	6	0.02750
h	4	0.0608	**q**	9	0.0401	**z**	9	0.0098
i	4	0.0695	**r**	4	0.0240	--	--	--

Efficiency

The Entropy is essentially the measure of uncertainity of a random variable with an associated probability set, $\mathrm{p}(x_i)$.

$$H(X) = -\sum_{i=1}^{n} (\mathrm{p}(x_i) \log \mathrm{p}(x_i))$$

In the following sections of the lab, you will be asked to determine the average word length **Error! Reference source not found.** and efficiency of the code **Error! Reference source not found.** given by

$$Average\ length = \bar{L} = E\{\ell\} = \sum_{i=1}^{n} \mathrm{p}(x_i)\, \ell_i$$

and,

$$Efficency = H(x)/\bar{L}$$

where $\mathrm{p}(x_i)$ is the probability set of the random variable, ℓ_i is the length of i[th] word, and $H(x)$ is the entropy of the source.

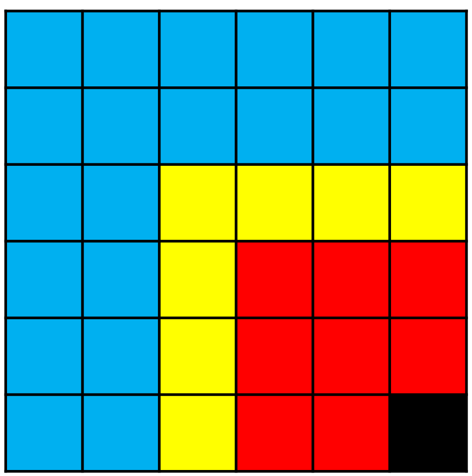
Using the frequency table and the Huffman code along with the equations, the average word length is 4.2015 average bits and the entropy is 4.1722 average bits. So the code's efficiency is 0.9930.

Image Compression

Complete the table by counting the number of squares with the color code. This is the data you will need to perform the experimental procedure. Note there are 4 color codes so N equals 4.

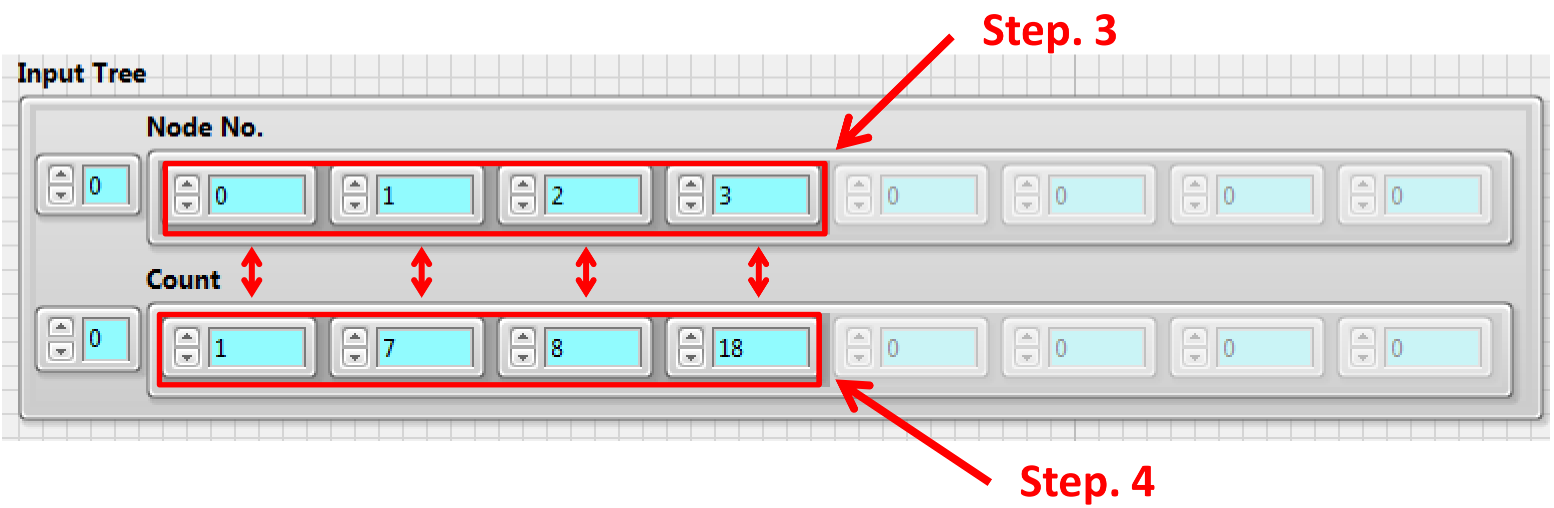
Image Frequency Counts

Color (Node Number)	Relative Frequency (Count)
0	
1	
2	
3	

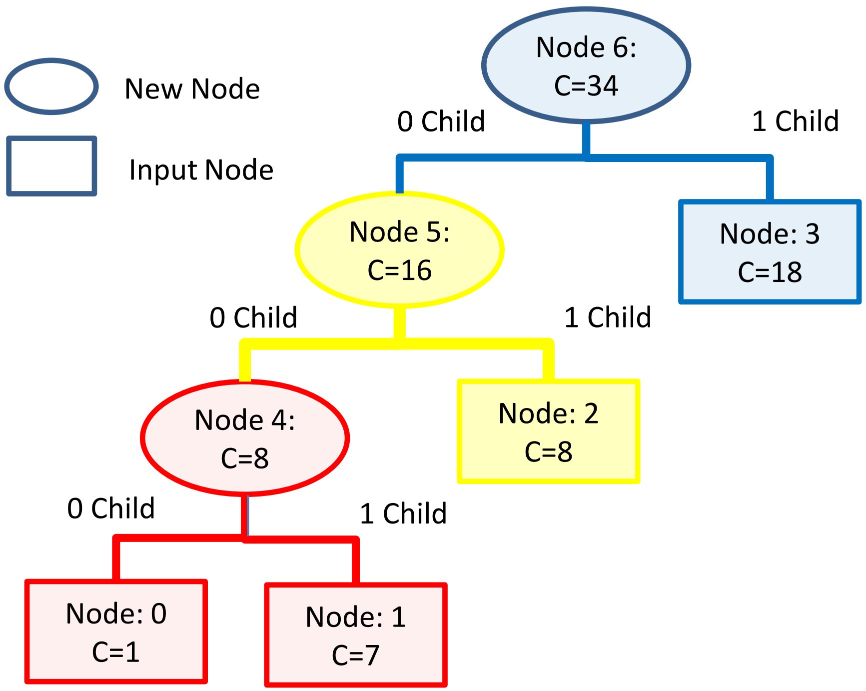


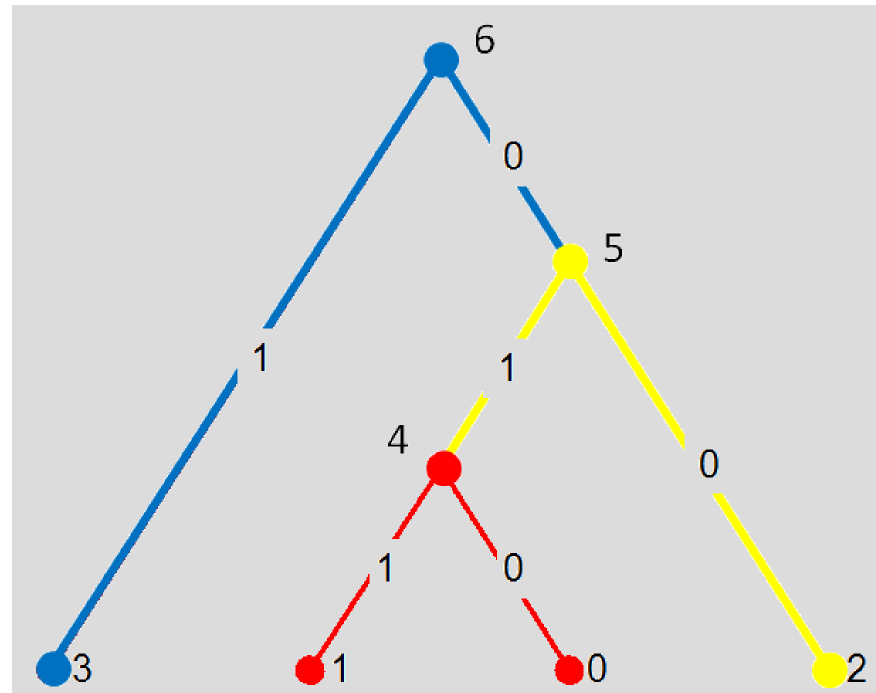
3	3	3	3	3	3
3	3	3	3	3	3
3	3	1	1	1	1
3	3	1	2	2	2
3	3	1	2	2	2
3	3	1	2	2	0

Entering The Data



Interpreting the Output





Asynchronous Serial Communication

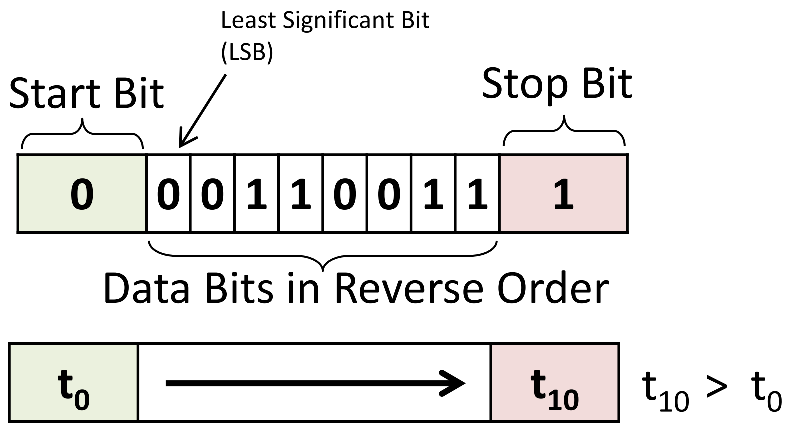
What you need to know to do the Lab

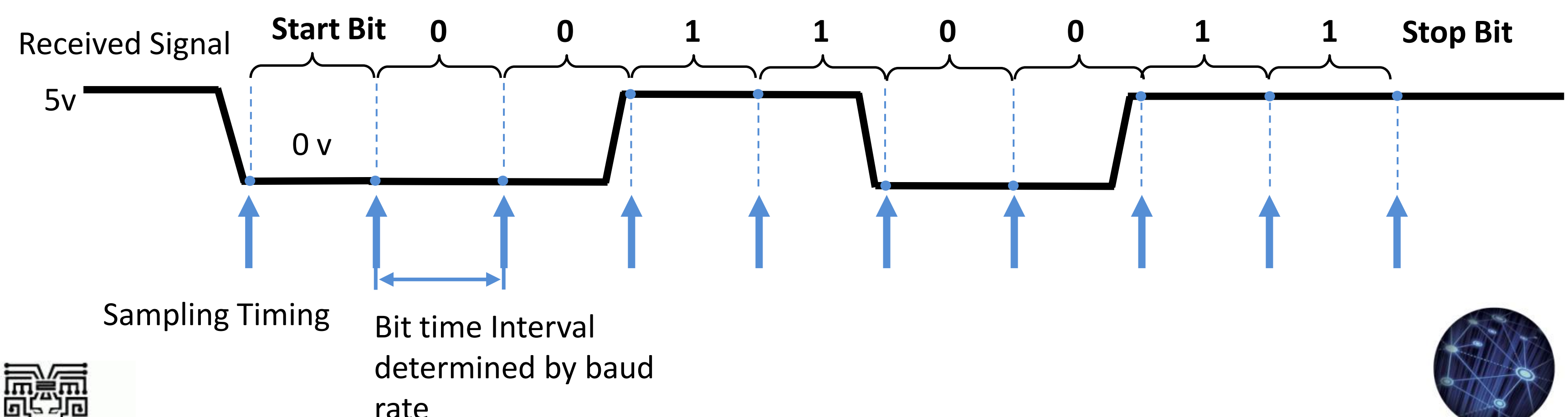
What You Are Doing

- You will be responsible for building the receiver portion of the UART for this lab.
  - This lab addresses the link between source coding/decoding and channel encoding/decoding.
  - Starts with a text string already encoded using the American Standard Code for Information Interchange (ASCII).
  - Additional 3 copies of each bit are used as the channel encoding.
  - The link is a serial interface that uses an UART to convert the encoded text into a sequence or stream.
  - To simplify the lab, the transmitted bit stream is passed directly to a UART receiver that reconverts the stream into the ASCII codes.

THE UNIVERSITY of NEW MEXICO

The Serial Data Packet



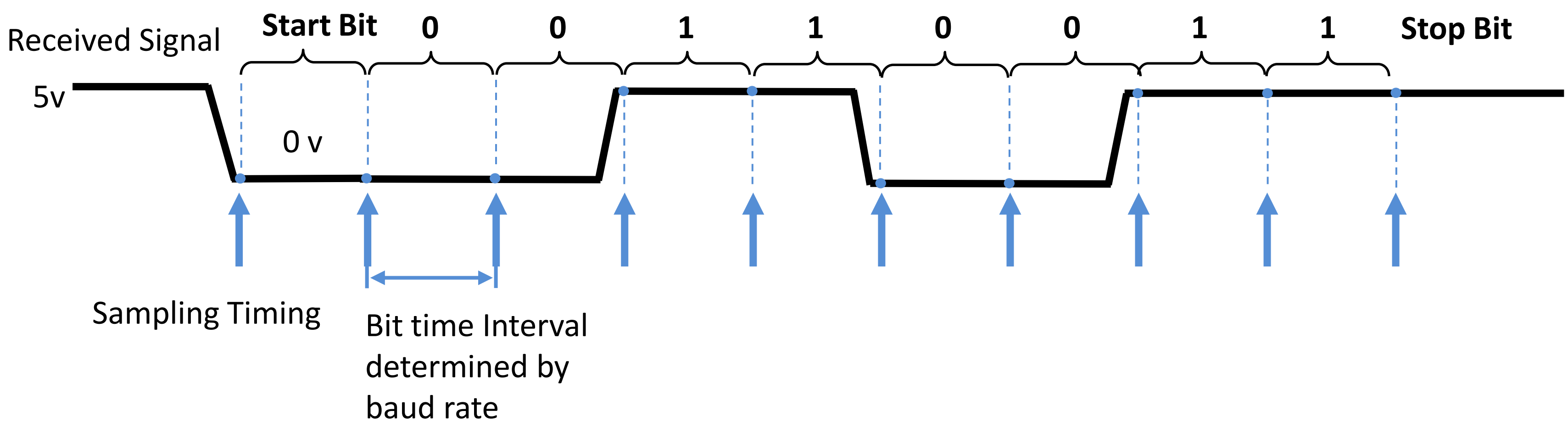


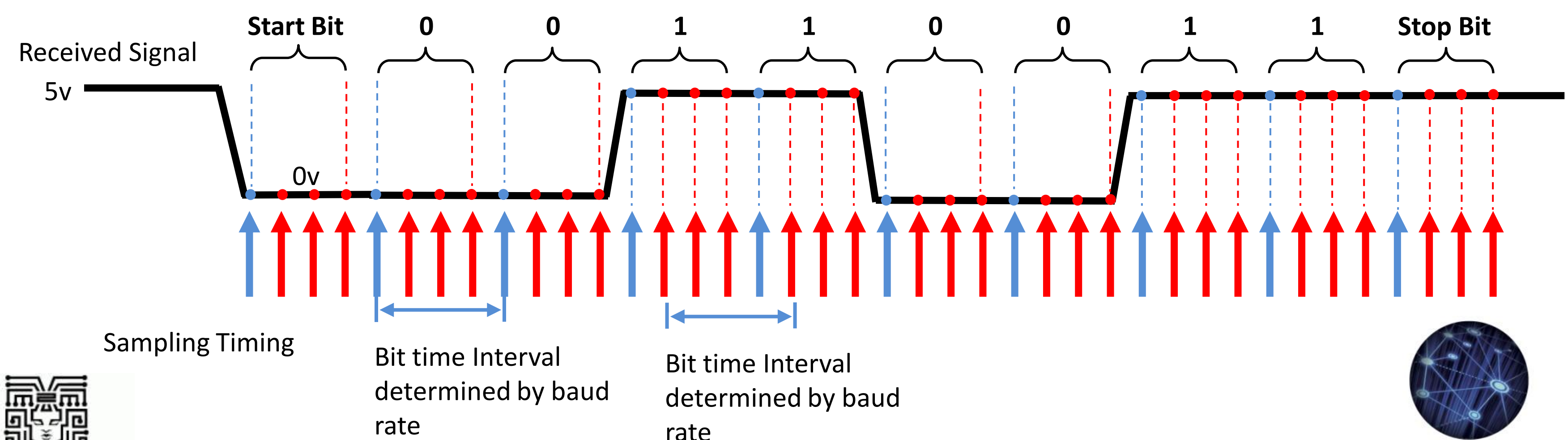
istec
Ibero-American Science & Technology Education Consortium

GINET
Global Innovation Network for Entrepreneurship and Technology

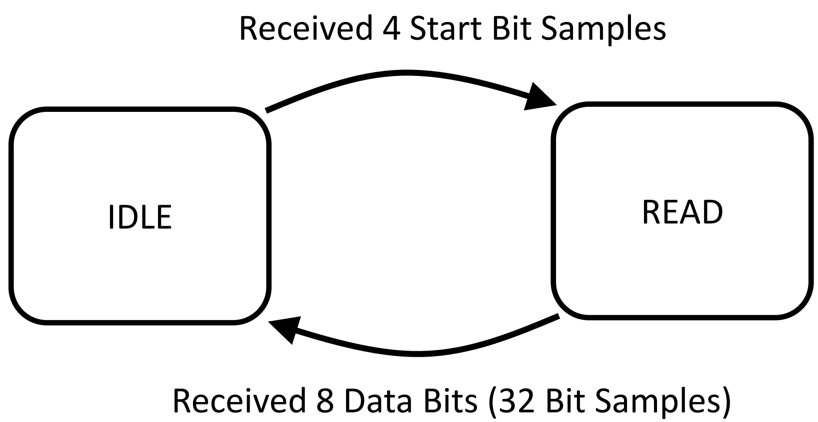
THE UNIVERSITY of NEW MEXICO

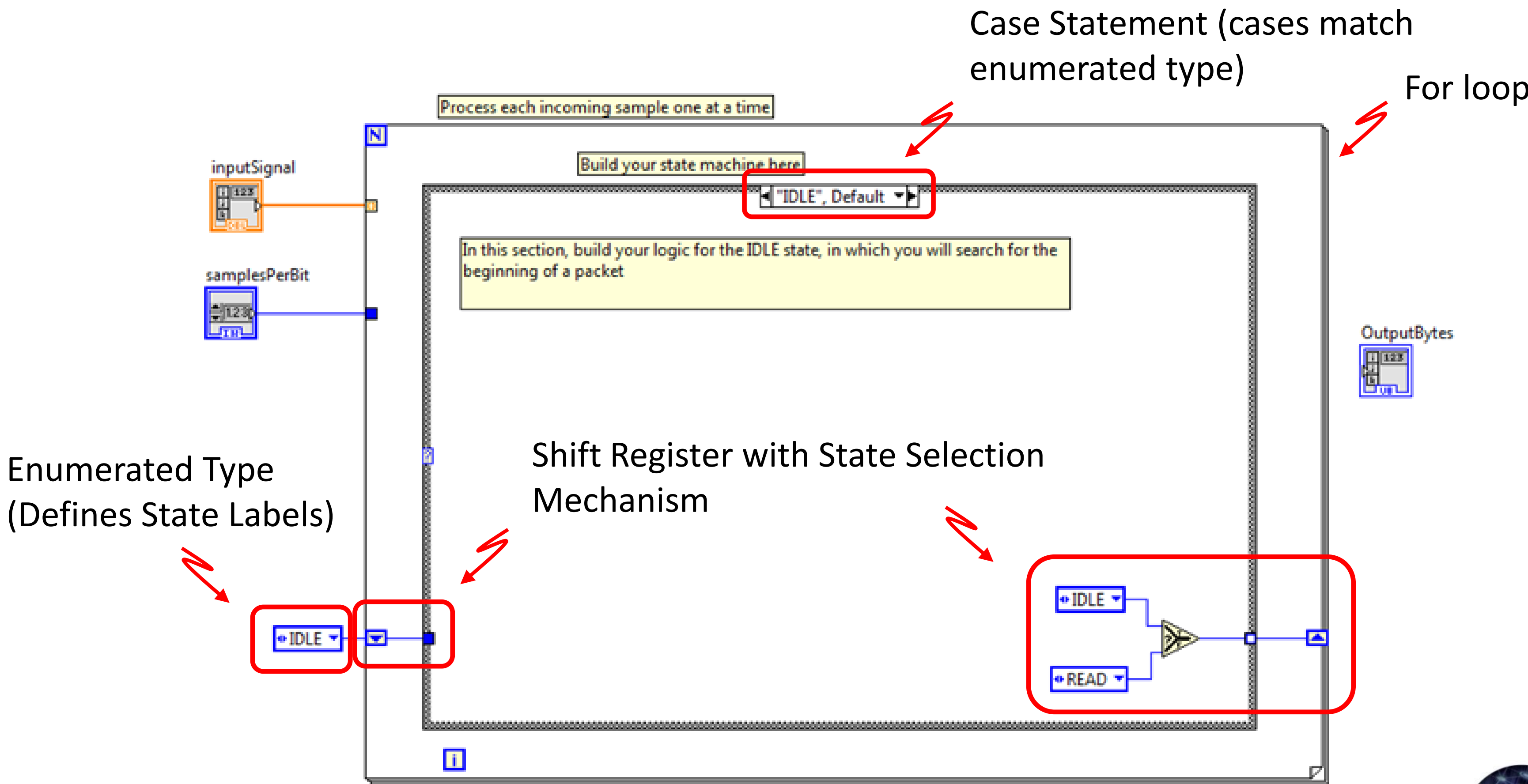
Redundancy Bits Added



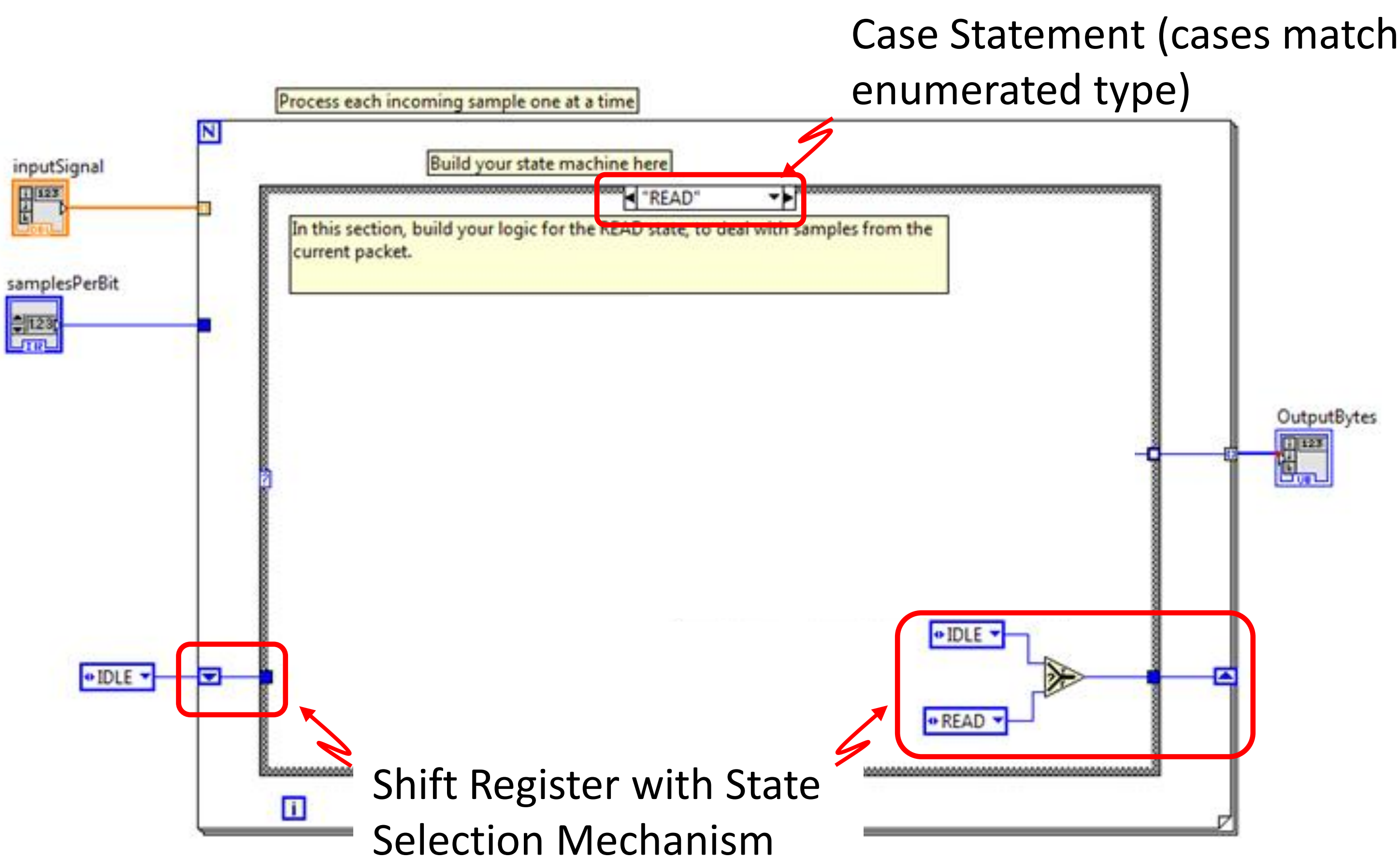


Receiver State Machine




Case Statement (cases match enumerated type)
For loop
Process each incoming sample one at a time
Build your state machine here
inputSignal
samplesPerBit
"IDLE", Default
In this section, build your logic for the IDLE state, in which you will search for the beginning of a packet
OutputBytes
Enumerated Type (Defines State Labels)
Shift Register with State Selection Mechanism
IDLE
IDLE
READ

THE UNIVERSITY of NEW MEXICO



istec
Ibero-American Science & Technology Education Consortium

GINET
Global Innovation Network for Entrepreneurship and Technology

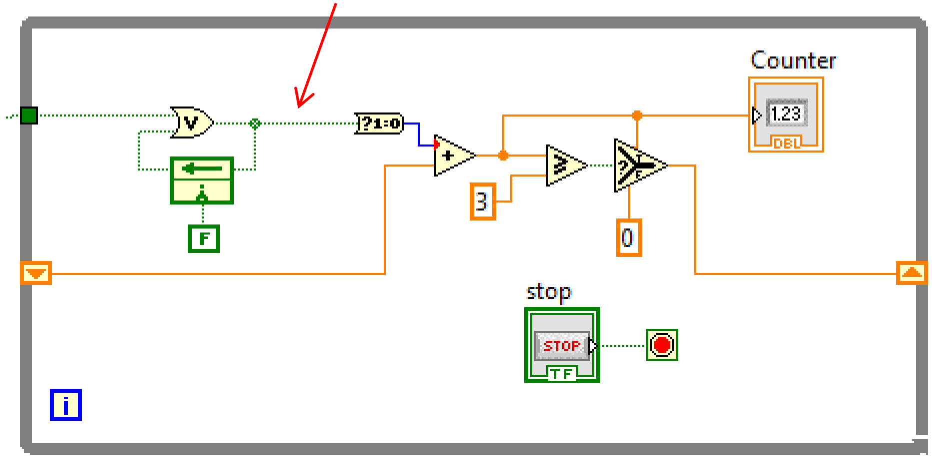
Data Latching & Counters

Latch Sets at t_2.

t_0	t_1	t_2	t_3	t_4	t_5
0	0	1	1	1	1

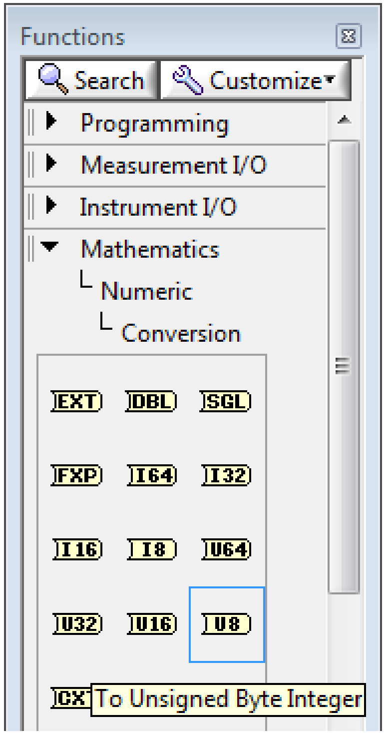
Counter start pulse is sent at t_2.

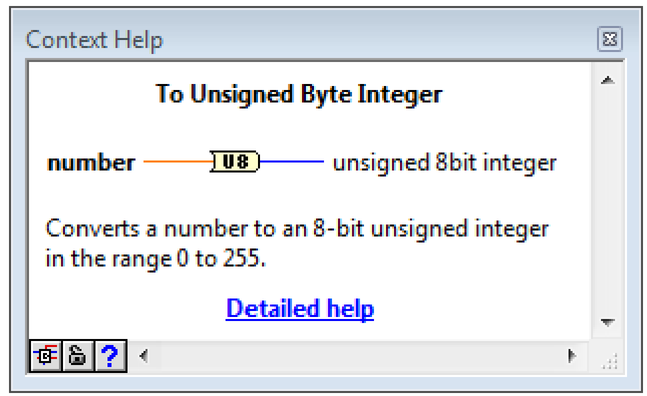
t_0	t_1	t_2	t_3	t_4	t_5
0	0	1	0	0	0

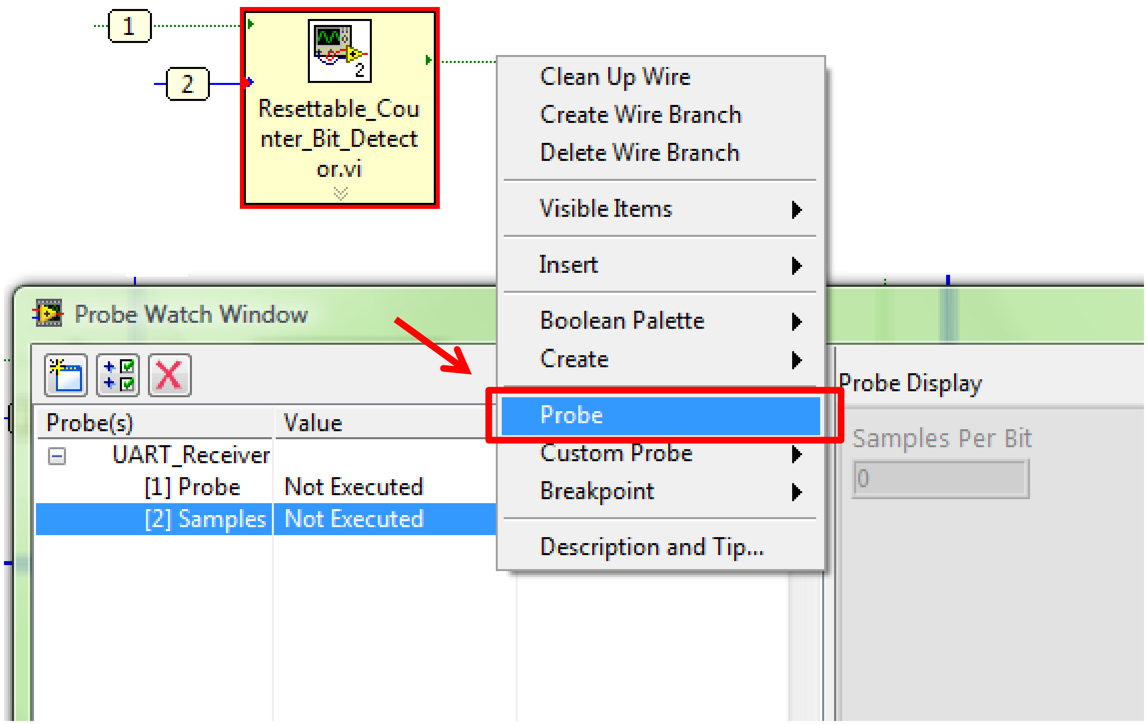


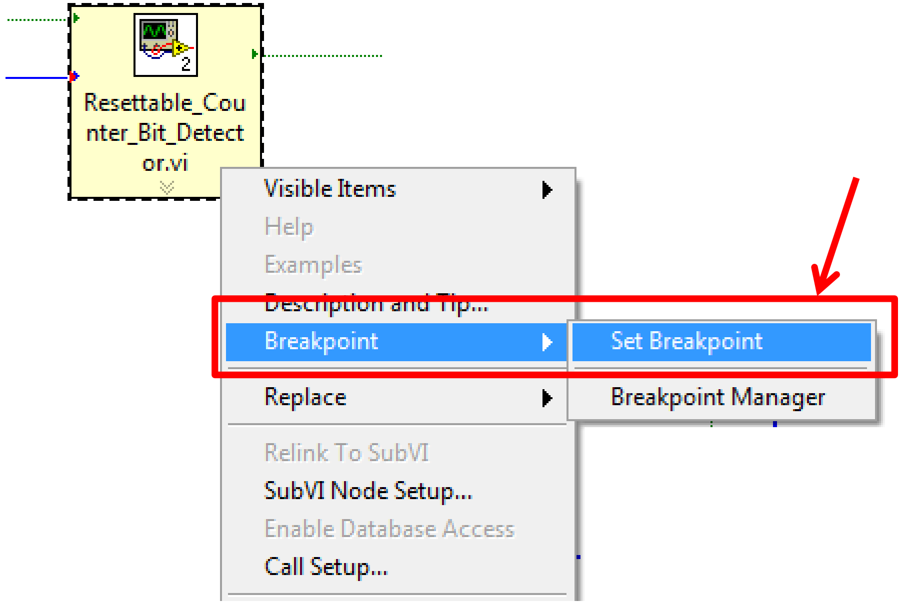
Counter outputs sequence 1,2,3 repeatedly.

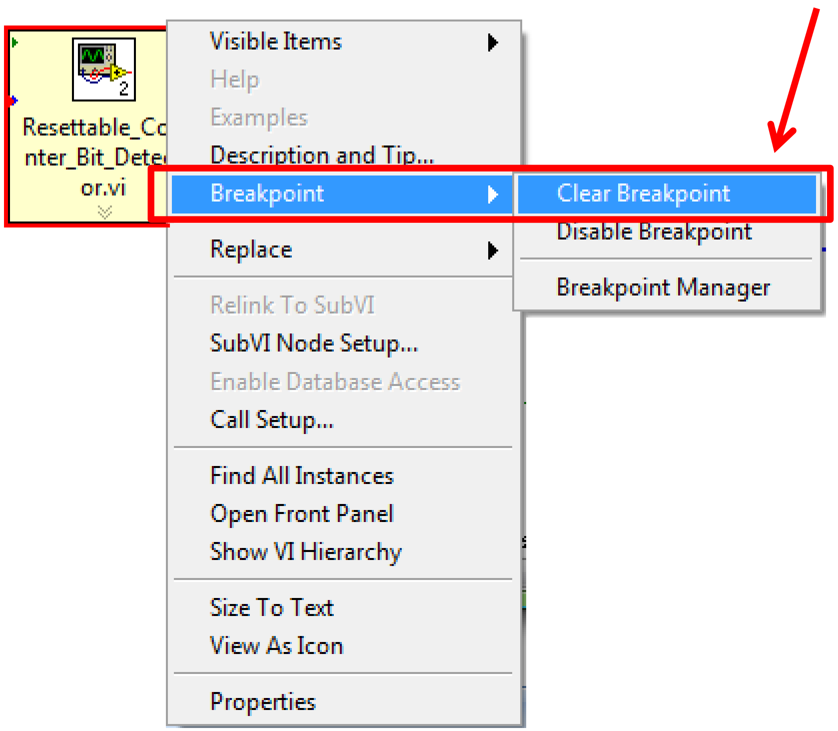
t_0	t_1	t_2	t_3	t_4	t_5
0	0	1	2	3	1


Functions
Search
Customize
Programming
Measurement I/O
Instrument I/O
Mathematics
Numeric
Conversion
EXT
DBL
SGL
FXP
I64
I32
I16
I8
U64
U32
U16
U8
To Unsigned Byte Integer


Context Help
To Unsigned Byte Integer
number
U8
unsigned 8bit integer
Converts a number to an 8-bit unsigned integer in the range 0 to 255.
Detailed help


1
2
Resettable_Counter_Bit_Detector.vi
Clean Up Wire
Create Wire Branch
Delete Wire Branch
Visible Items
Insert
Boolean Palette
Create
Probe
Custom Probe
Breakpoint
Description and Tip...
Probe Watch Window
Probe(s)
Value
UART_Receiver
[1] Probe
Not Executed
[2] Samples
Not Executed
Probe Display
Samples Per Bit
0


Resettable_Counter_Bit_Detector.vi
Visible Items
Help
Examples
Description and Tip...
Breakpoint
Set Breakpoint
Breakpoint Manager
Replace
Relink To SubVI
SubVI Node Setup...
Enable Database Access
Call Setup...

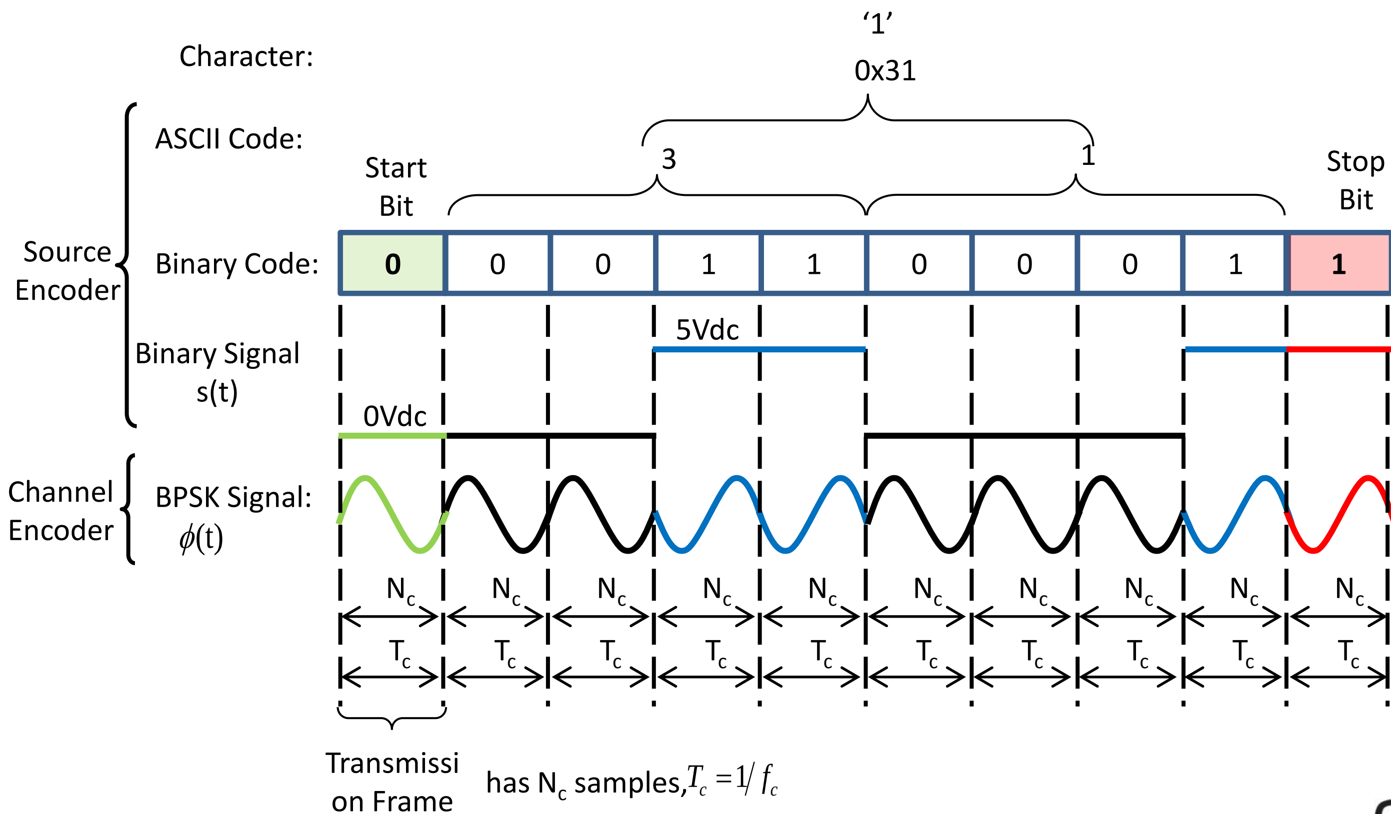

Resettable_Counter_Bit_Detector.vi
Visible Items
Help
Examples
Description and Tip...
Breakpoint
Clear Breakpoint
Disable Breakpoint
Breakpoint Manager
Replace
Relink To SubVI
SubVI Node Setup...
Enable Database Access
Call Setup...
Find All Instances
Open Front Panel
Show VI Hierarchy
Size To Text
View As Icon
Properties

THE UNIVERSITY of NEW MEXICO

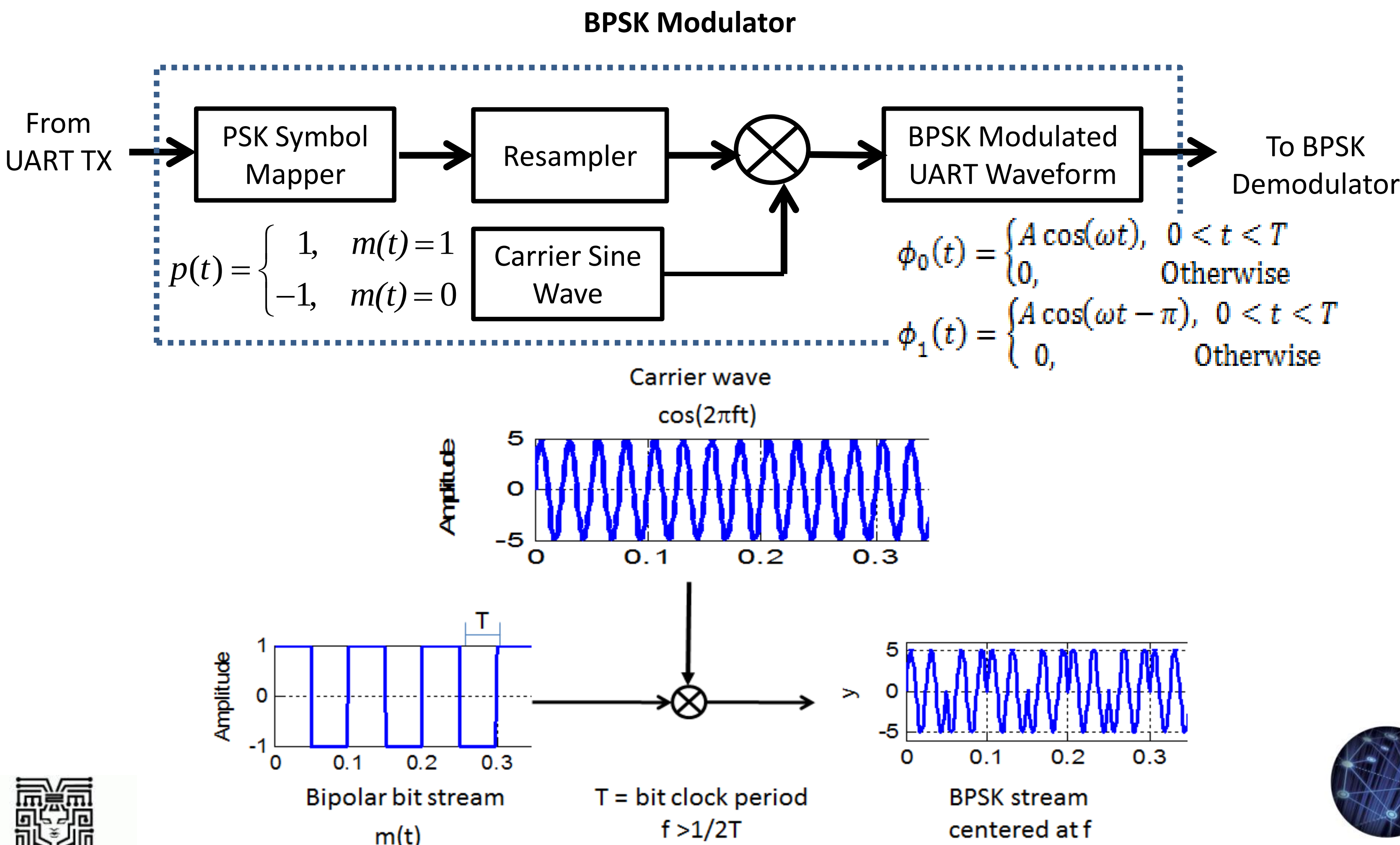
Binary Phase Shift Keying

THE UNIVERSITY of NEW MEXICO

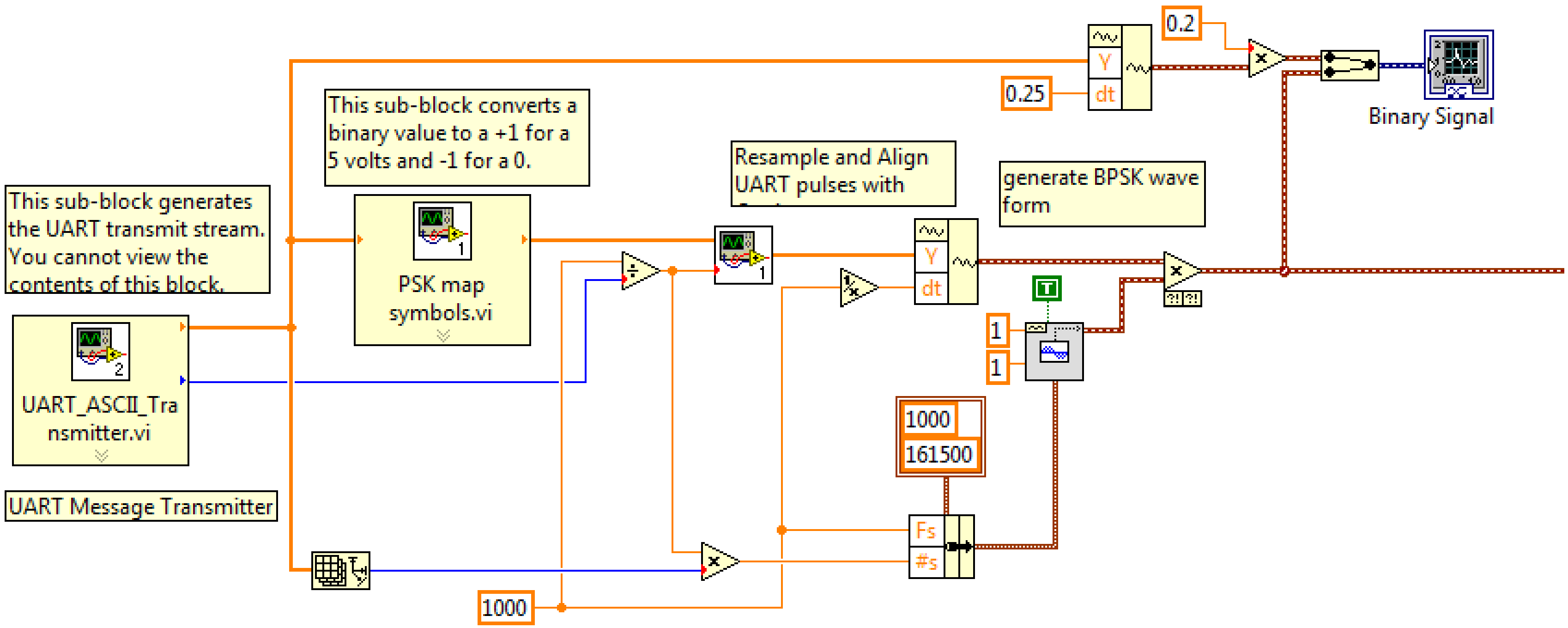
In PSK (Phase Shift Keying), the phase of a carrier is changed between two values according to the binary signal level[3]. The information about the bit stream is contained in the phase changes of the transmitted signal.

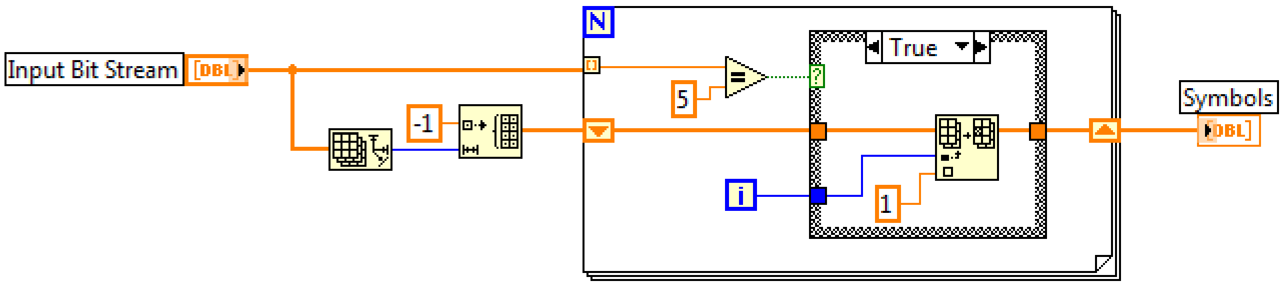


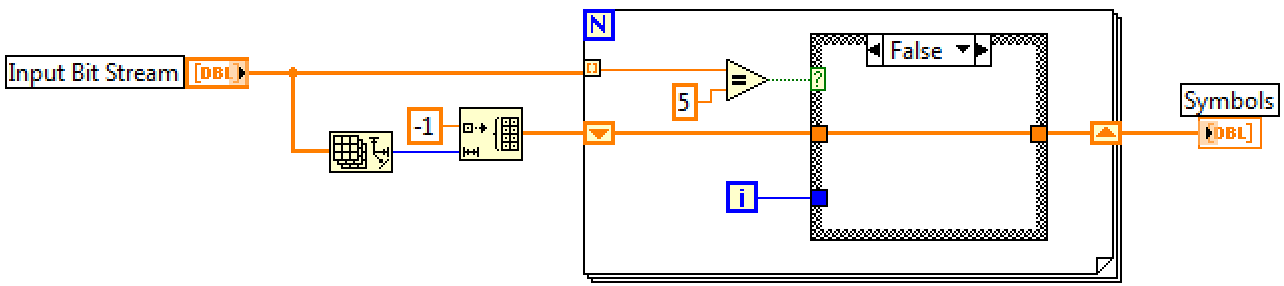
Implementation

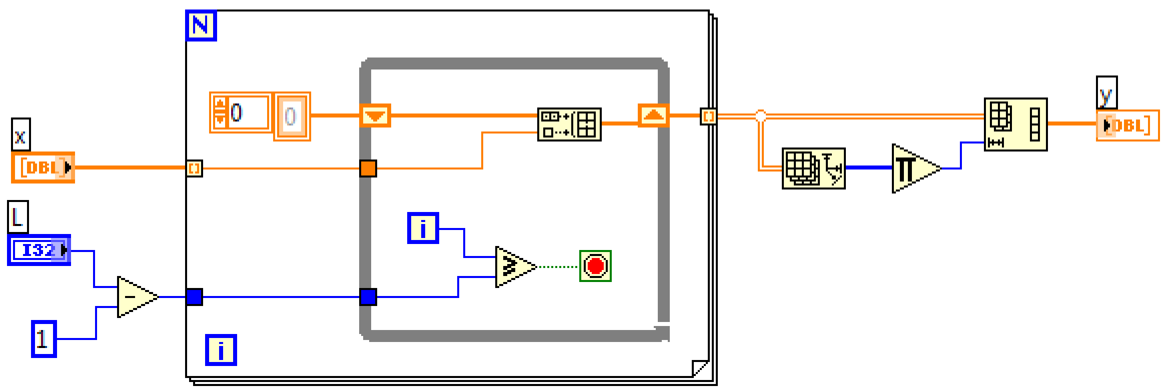


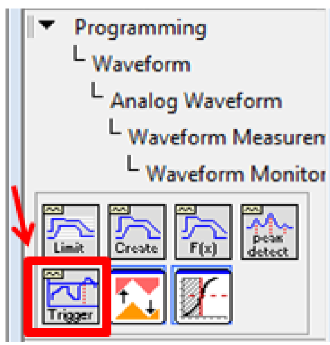
Printed on 9/1/2014


0.2
0.25
Y
dt
Binary Signal
This sub-block converts a binary value to a +1 for a 5 volts and -1 for a 0.
Resample and Align UART pulses with
generate BPSK wave form
This sub-block generates the UART transmit stream. You cannot view the contents of this block.
PSK map symbols.vi
UART_ASCII_Transmitter.vi
UART Message Transmitter
T
1
1
1000
161500
Fs
#s
1000

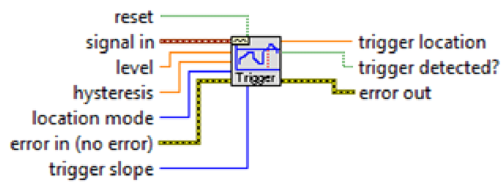

Input Bit Stream
DBL
N
-1
5
=
True
1
i
Symbols
DBL


Input Bit Stream
DBL
N
-1
5
=
False
i
Symbols
DBL

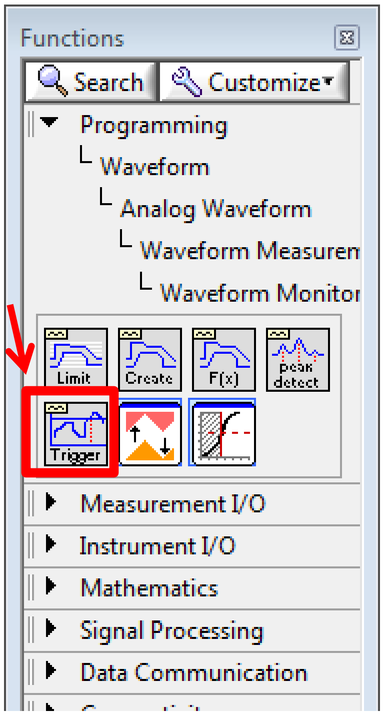

N
x
DBL
L
I32
1
0
0
i
i
y
DBL



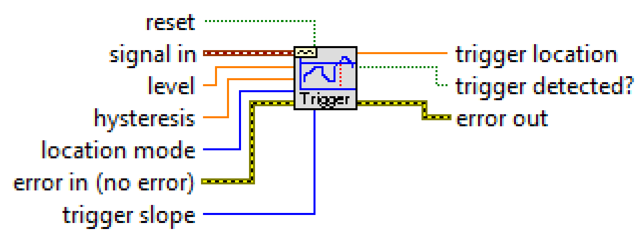
Basic Level Trigger Detection.vi



Finds the first level-crossing location in a waveform. You can retrieve the trigger location as an index or as a time. The trigger conditions are specified in terms of threshold **level**, **slope**, and **hysteresis**. Wire data to the **signal in** input to determine the polymorphic instance to use or manually select the instance.



Basic Level Trigger Detection.vi



Finds the first level-crossing location in a waveform. You can retrieve the trigger location as an index or as a time. The trigger conditions are specified in terms of threshold **level**, **slope**, and **hysteresis**. Wire data to the **signal in** input to determine the polymorphic instance to use or manually select the instance.