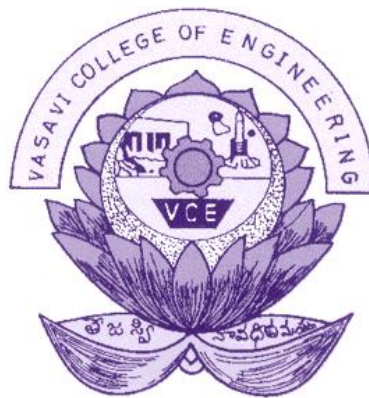


*VASAVI COLLEGE OF ENGINEERING
(AUTONOMOUS),
IBRAHIMBAGH, HYDERABAD – 31.*



*DEPARTMENT OF ELECTRICAL & ELECTRONICS
Fundamentals of LabVIEW*

*By
Sreenivasulu Meda
&
Ravi Ponnala*

Table of Contents

1.	INTRODUCTION	1
1.1	Introduction	1
1.2	Opening Labview	1
1.3	Creating a New VI	2
1.4	Saving a VI	4
1.5	Executing a VI	5
1.6	Highlight Execution of a VI	5
1.7	Error Handling in a VI	5
1.8	Control Palette and Functions Palette	6
1.8.1	Numerical Controls and Indicators	6
1.8.2	Boolean Controls and Indicators	7
1.8.3	String Controls and Indicators	7
1.9	Data Types	7
1.9.1	Numeric Data Type	7
1.9.1.1	Floating – Point Data Type	8
1.9.1.2	Fixed – Point Data Type	8
1.9.1.3	Integer Data Type	8
1.9.1.4	Complex Number Data Type	9
1.9.2	Boolean Data Type	9
1.9.3	String Data Type	11
1.9.4	Enum Data Type	12
1.10	Building a Simple VI	13
2.	Basic Examples	15
2.1	Average of Three Numbers	15
2.2	Pressure and Temperature indicator	16
2.3	Inrange and Coerce	17
3.	Exercises on Structures	19
3.1	Introduction	19
3.2	Case Structure	19
3.2.1	Numerical Case	19
3.2.2	Boolean Case	22
3.2.3	Enum Case	23

3.2.4	Case Structure with String	25
3.2.5	Case Structure with Ring.....	25
3.3	While Loop	27
3.3.1	Fill water in a tank up to a desired level using while loop.....	27
3.4	Flat Sequence Structure.....	28
3.5	Sequence Structure with Stacking	30
3.6	Structure Tunnels.....	34
1.11	For Loop	35

1. INTRODUCTION

1.1 Introduction

LabVIEW is expanded as Laboratory Virtual Instrument Engineering Workbench

LabVIEW is graphical programming language that allows for instrument control, data acquisition, and signal processing.

LabVIEW programs are called virtual instruments or simply VIs, because their appearance and operation imitates physical instruments, such as oscilloscopes and multimeters.

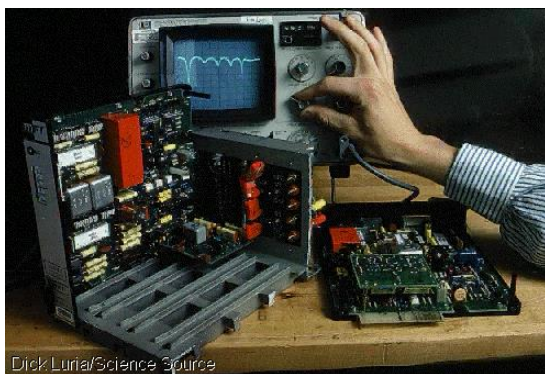
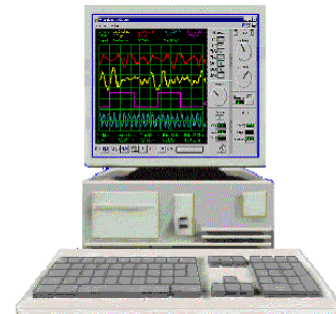


Figure 1.1: (a). Physical instruments



(b). Virtual instruments

1.2 Opening Labview

Press windows icon in the system and search for “Labview 2017 (32 bit)” and click it. The following screen appears.

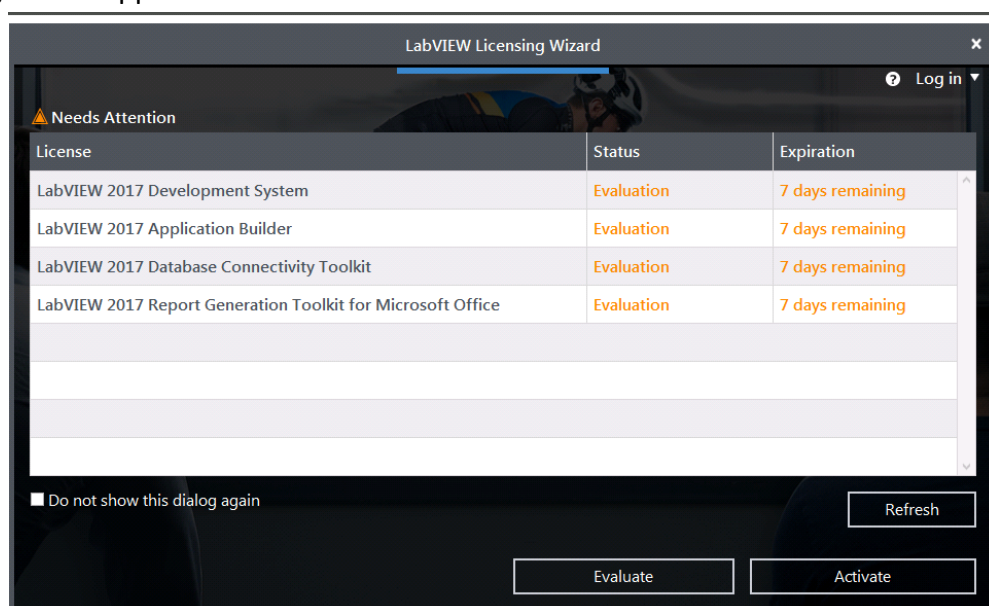


Figure 1.2: Labview Licensing Wizard

Select **Evaluate>>Extend valuation>>No**. Labview opens and the following window appears.

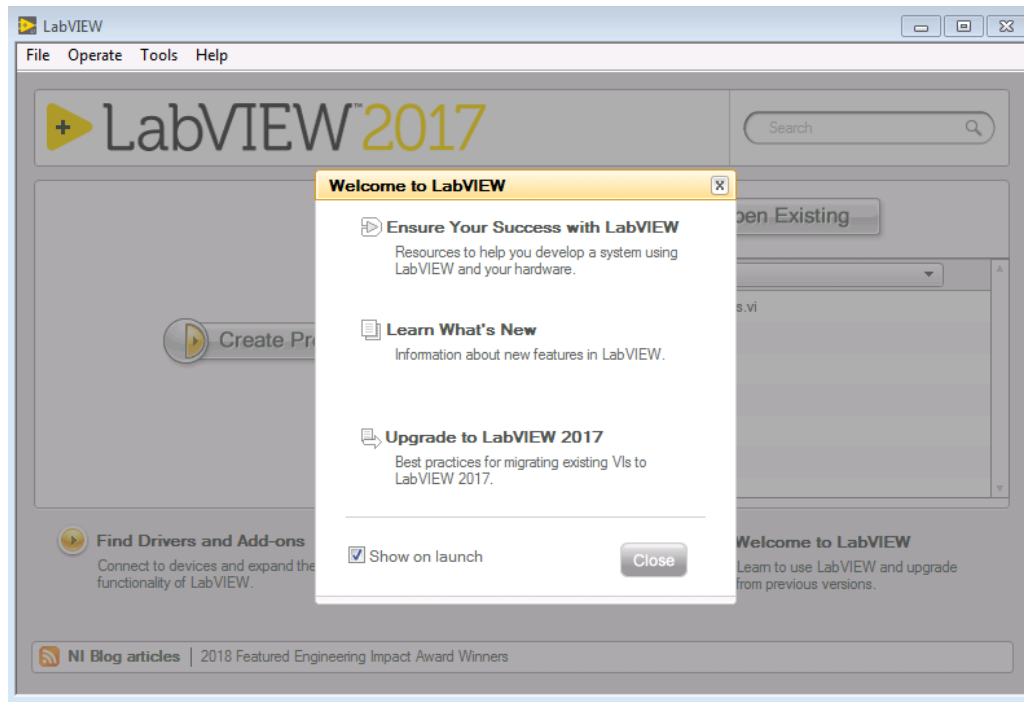


Figure 1.3: Main screen

Click on **Close**.

1.3 Creating a New VI

Select **File>>New VI**. Labview VI consists of two parts namely front panel window and block diagram. After clicking **New VI**, by default, front panel of Labview appears.

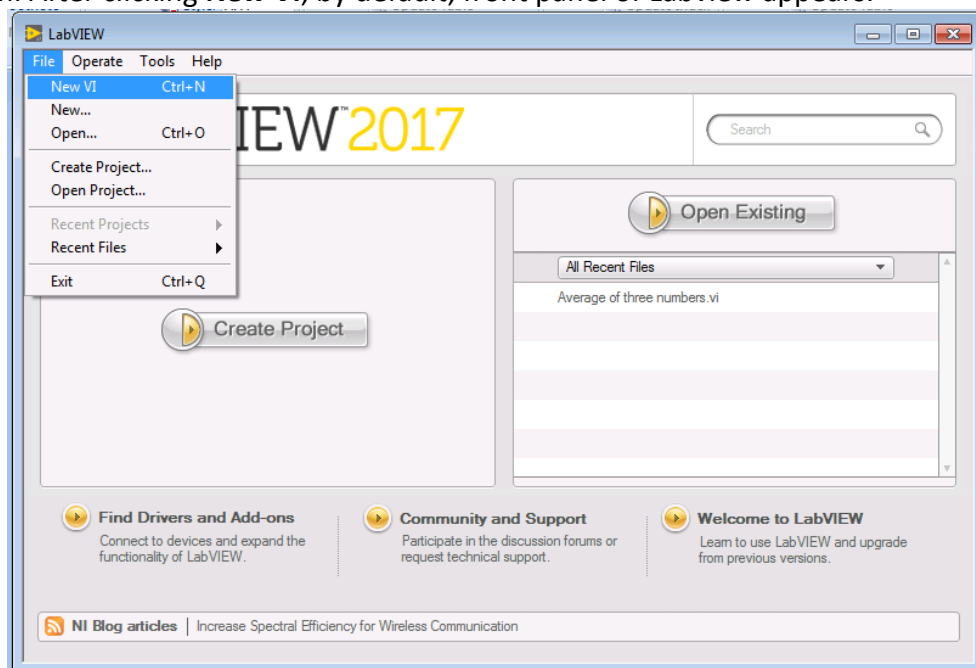


Figure 1.4: Creating a new VI

Press “Ctrl+T”. Both front panel and block diagram appears side by side as shown below:

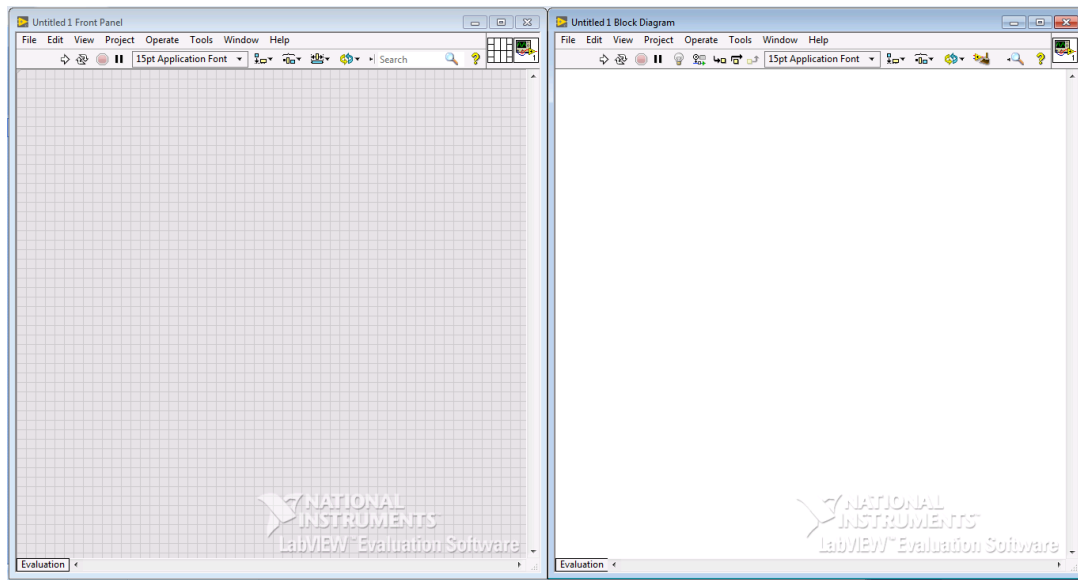


Figure 1.5: Empty front panel and block diagram of VI

Front panel window is window where user interacts either by giving an input or views the result of a VI as an output. The user interacts in the front panel using controls and indicators. Controls act as input to the block diagram and will be in the form of push buttons, knobs, dialers etc. Indicators act as output and will be in the form of graphs, charts, LEDs and other displays. An example view of front panel is as shown below:

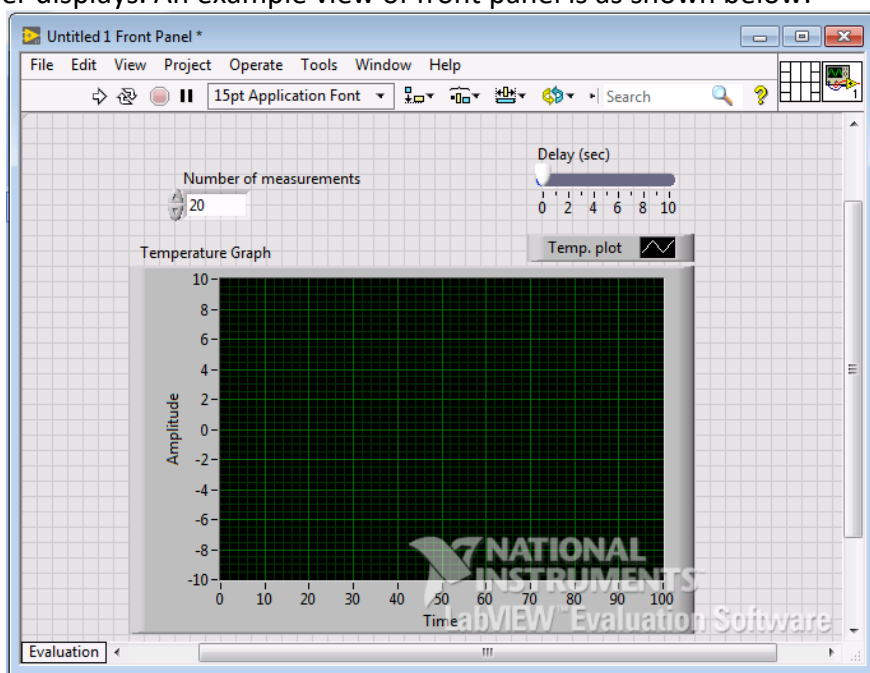


Figure 1.6: Front panel of VI

Block diagram is the window where code is added to control the front panel objects. The front panel objects appear as terminals in the block diagram. The code is in the form of graphical representations of functions. An example view of block diagram is as shown below:

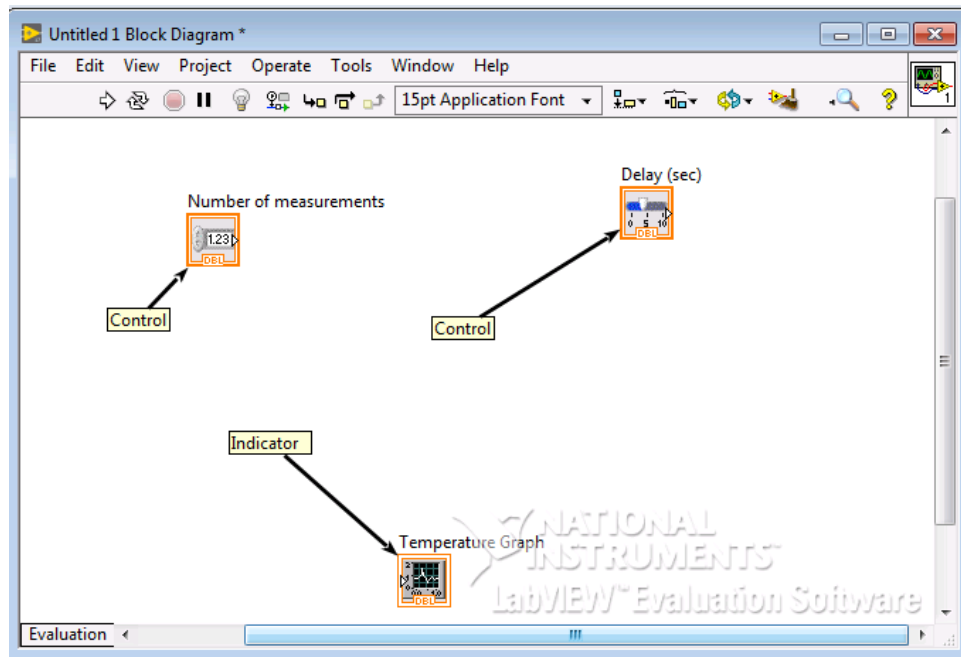


Figure 1.7: Block diagram of VI

1.4 Saving a VI

To save a VI, select **File>>Save**. To save an already saved file, select **File>>Save As**. A dialog box as shown below appears. From the dialog box, select appropriate option and select **Continue**.

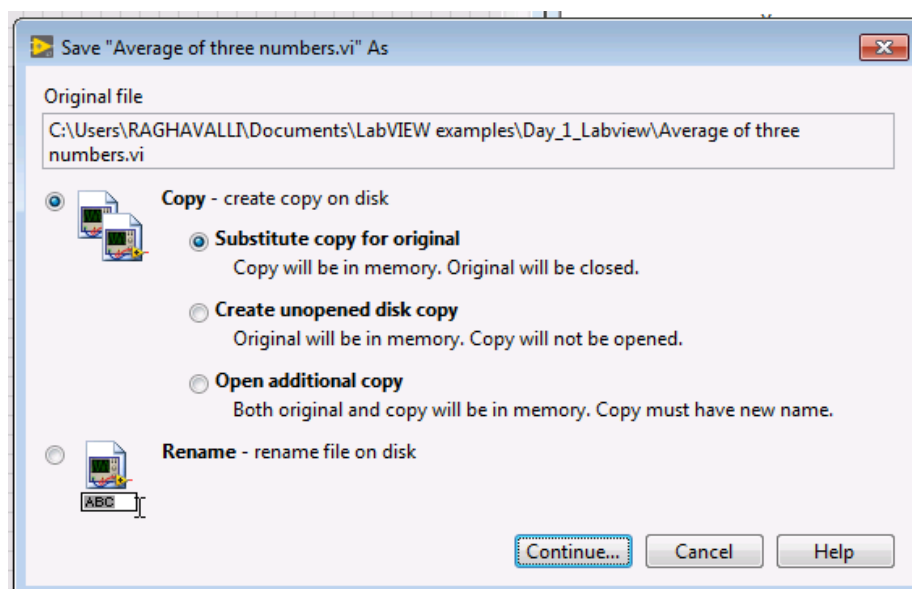


Figure 1.8: Saving a VI

1.5 Executing a VI




To execute or run a VI, click on the icon  in the front panel toolbar or in the block diagram toolbar.




Figure 1.9: Front panel toolbar in a VI

To run continuously, click on the icon  in the toolbar.

To abort execution, click on the icon  in the toolbar. However, this is not recommended and instead of this design VIs using stop button.

1.6 Highlight Execution of a VI

Highlight execution is used to observe the flow of data in the block diagram. To enable highlight execution, click on the icon  in the block diagram toolbar of a VI. To disable highlight execution, again click on the same icon.

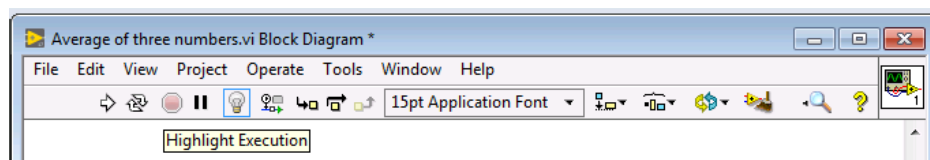



Figure 1.10: Block diagram toolbar in a VI

1.7 Error Handling in a VI

If any errors exist in the designed VI, then the run icon appears as broken  in the front panel or block diagram. Click this button and an **Error list** window appears.

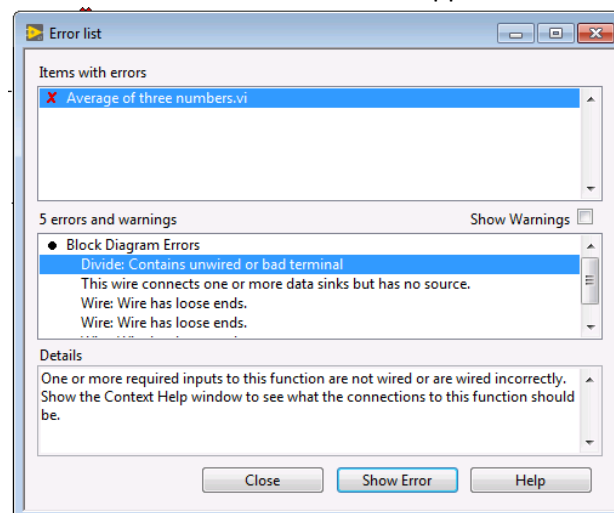


Figure 1.11: Error list window of a VI

The **Error list** window displays the list of errors. Click on any error and the details of error are displayed. Rectify the errors and then run the VI again.

1.8 Control Palette and Functions Palette

Controls palette contains the controls and indicators that are used to model the front panel. **Controls** palette can be accessed from the front panel by selecting **View>>Controls Palette** or just by right clicking in the front panel. **Functions** palette contains the VIs, functions and constants that are used to model the block diagram. **Functions** palette can be accessed from the block diagram by selecting **View>> Functions Palette** or just by right clicking in the block diagram. View of **Controls** palette window and **Functions** palette window is as shown below:

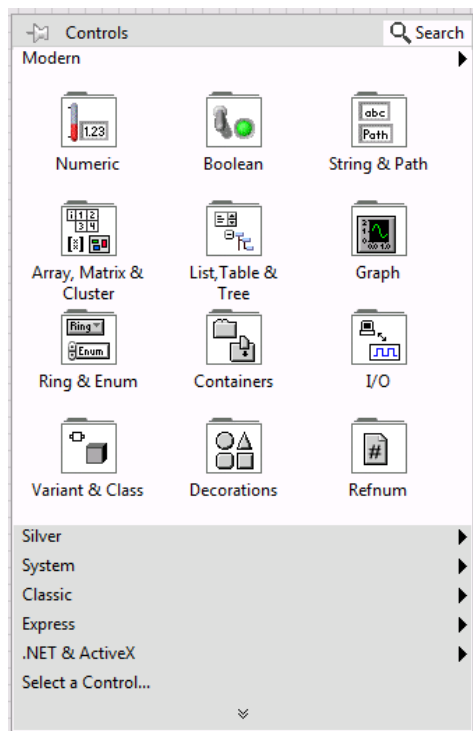
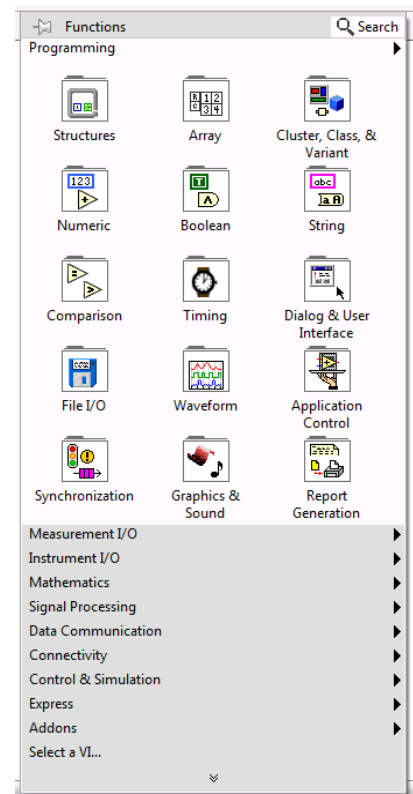


Figure 1.12: (a) Controls palette



(b) Functions palette

The **Controls** palette has different types of controls and indicators namely numeric, Boolean, string etc.

1.8.1 Numerical Controls and Indicators

Numerical controls and indicators have numeric data types and an example is shown below:

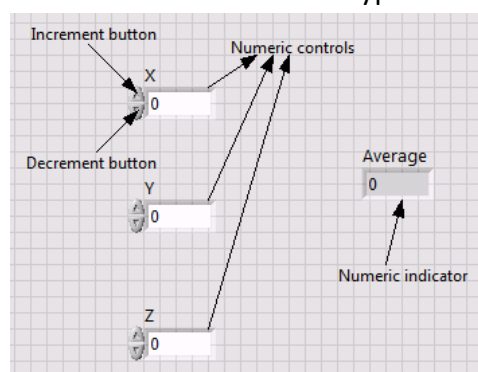


Figure 1.13: Numerical controls and indicator

1.8.2 Boolean Controls and Indicators

Boolean controls and indicators can have only two states or values namely TRUE and FALSE or ON and OFF. Examples of boolean controls and indicators are toggle switches, push buttons, LEDs etc.

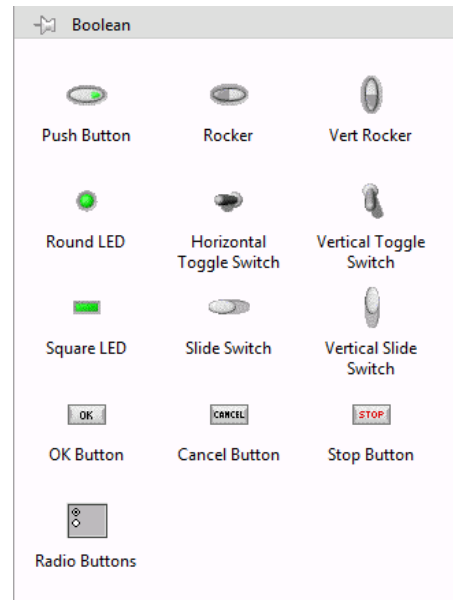


Figure 1.14: Boolean controls and indicator

1.8.3 String Controls and Indicators

String controls and indicators consist of ASCII characteristics. Examples of string data is name of a location or building etc.

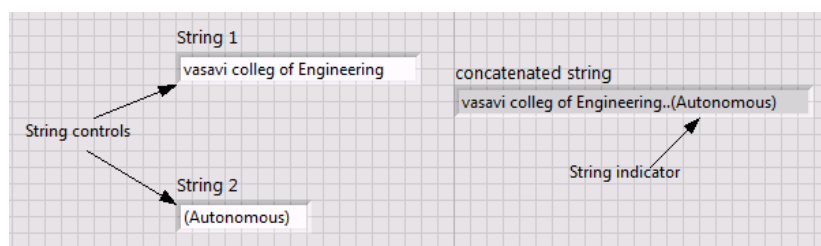


Figure 1.15: String controls and indicator

1.9 Data Types

Data exists in the form of various data types such as numeric, boolean, string, enumerated, dynamic etc.

1.9.1 Numeric Data Type

Numeric data type is used to represent numbers. This representation may be floating – point, fixed – point, integer and complex number.

1.9.1.1 Floating – Point Data Type

Floating data type is used to represent fractional numbers. In Labview, all integers are represented using orange colour. The following are the floating point representations used in Labview.

Single – precision (SGL): Single – precision, floating point number has 32 bit IEEE single – precision format. To save memory and to avoid overflowing range of numbers, this representation is preferred.

Double – precision (DBL): Double – precision, floating point number has 64 bit IEEE double – precision format. This is the default method of numeric representation in Labview.

Extended – precision (EXT): In memory, the size and precision of extended – precision numbers vary depending on the platform. In windows, they have 80 bit IEEE extended – precision format.

1.9.1.2 Fixed – Point Data Type

This is a numeric data type that represents a set of rational numbers using binary digits, or bits. To represent a rational number using fixed – point data type, the denominator of the rational number must be a power of 2, because the binary number system is a base – 2 numbersystem. This type of representation is preferred when there is no need of dynamic functionality of floating point representation.

1.9.1.3 Integer Data Type

Integer represents whole numbers. They may be signed or unsigned integers. Signed integers can take positive or negative values. Unsigned integers are those who values are only positive. In Labview, all integers are represented using blue colour. The following are the integer representations used in Labview.

Byte signed integer number (I8) – Allocates 8 bits of storage and number ranges from -2^7 to 2^7-1 . (-2^{n-1} to $2^{n-1}-1$, where 'n' is no. of bits)

Word signed integer number (I16) – Allocates 16 bits of storage and number ranges from -2^{15} to $2^{15}-1$.

Long signed integer number (I32) – Allocates 32 bits of storage and number ranges from -2^{31} to $2^{31}-1$.

Quad signed integer number (I64) – Allocates 64 bits of storage and number ranges from -10^{19} to 10^{19} .

Byte unsigned integer number (U8) – Allocates 8 bits of storage and number ranges from 0 to 2^8-1 .

Word unsigned integer number (U16) – Allocates 16 bits of storage and number ranges from 0 to $2^{16}-1$ (0 to 2^n-1).

Long unsigned integer number (U32) – Allocates 32 bits of storage and number ranges from 0 to $2^{32}-1$.

Quad unsigned integer number (U64) – Allocates 64 bits of storage and number ranges from 0 to $2e19$.

1.9.1.4 Complex Number Data Type

Complex number data type consists of real part and imaginary part. This type of data is also represented with orange colour in Labview. The following are the complex number representations used in Labview.

Complex Single (CSG): Complex single – precision, floating – point number takes real and imaginary values in 32 bit IEEE single – precision format.

Complex Double (CDB): Complex double – precision, floating – point number takes real and imaginary values in 64 bit IEEE double – precision format.

Complex Extended (CXT): Complex extended – precision, floating – point number takes real and imaginary values in IEEE extended – precision format.

1.9.2 Boolean Data Type

In Labview, boolean data is stored as an 8 – bit value. If this value is zero, then boolean value is false. If this value is non zero, then boolean value is true. This type of data is represented with green colour in Labview. Boolean value has two types mechanical actions namely switch and latch. These are further classified as follows:

Switch when pressed: Changes the control value every time the operating tool is clicked.

Switch when released: Changes the control value only after the mouse button is released within the graphical boundary of control.

Switch until released: Changes the control value when operating tool is clicked and retains the new value until the mouse button is released.

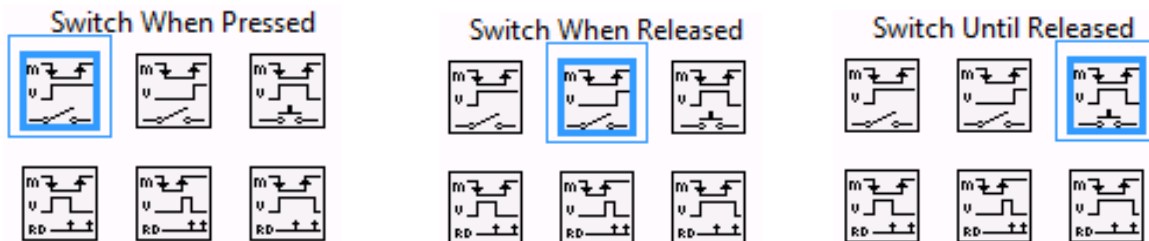


Figure 1.16: Switching actions of boolean value

Latch when pressed: Changes the control value when the operating tool is clicked. However, the new value is retained only till the VI reads it once. After the VI reads the value, the control reverts to its default value even though the operating tool is still clicked.

Latch when released: Changes the control value only after the mouse button is released within the graphical boundary of control. However, the new value is retained only till the VI reads it once. After the VI reads the value, the control reverts to its default value.

Latch until released: Changes the control value when operating tool is clicked and retains the new value until the VI reads it once or the mouse button is released, depending on which value occurs last.

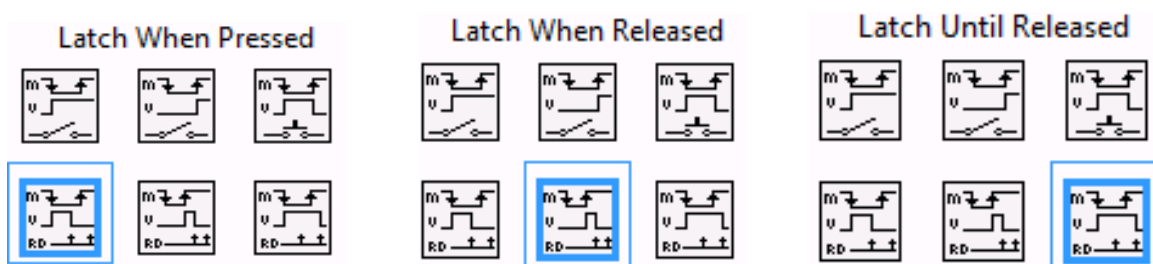


Figure 1.17: Latching actions of boolean value

To change the mechanical action, right click on the boolean switch, select mechanical action

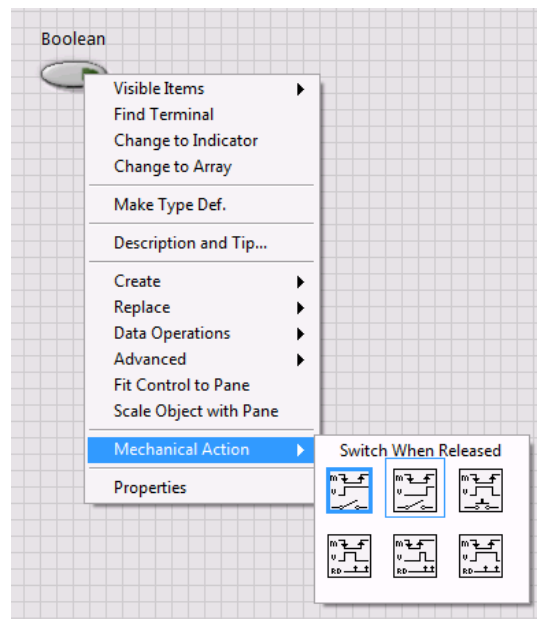


Figure 1.18: Editing the switching actions of boolean value

1.9.3 String Data Type

String data type consists of ASCII characteristics. In Labview, strings are represented with pink colour. String control or indicator has four display types.

Display Type	Description	Message
Normal Display	Printable characters are displayed same as they are entered in control.	Vasavi College of Engineering
'\Codes Display	All non-displayable characters are displaced as backlash	Vasavi\College\of\Engineering
Password Display	All characters are displayed as “*”	*****
Hex Display	ASCII value of each character is displayed instead of the character enter.	5661 7361 7669 2043 6F6C 6C65 6765 206F 6620 456E 6769 6E65 6572 696E 67

Table 1.1: Types of string display

To change the type of display, right click on string control or indicator and select required

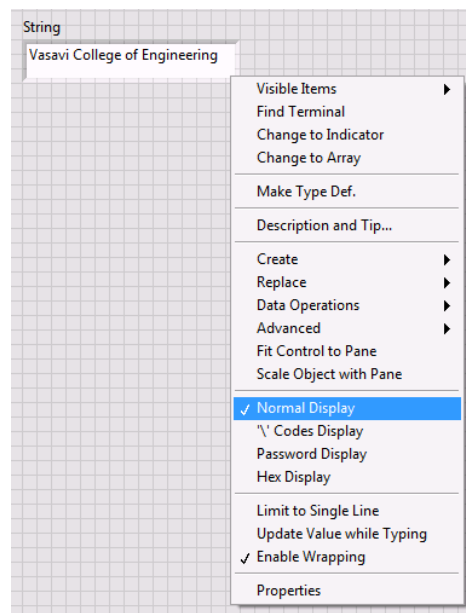


Figure 1.19: Changing the type of string display

1.9.4 Enum Data Type

Enum is a combination of data types. It represents a pair of values, a string and a numeric where enum can be one of a list of values. An example of enum is mathematical operation, where the possible value pairs for the mathematical operation can be Add -0, Sub-1, Mul-2 and Div-3. Enum is preferred as it is easier to operate on numbers on the block diagram rather than strings. In Enum, the values are fixed and starts from "0".

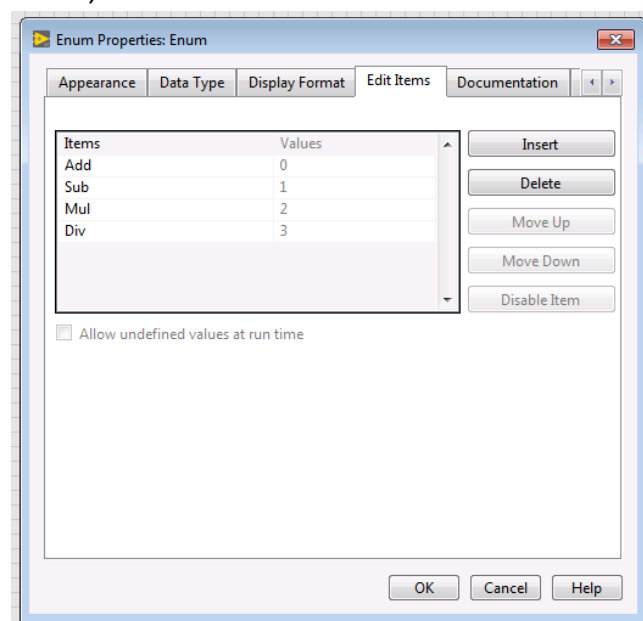


Figure 1.20: Enum

1.10 Building a Simple VI

Let us create a simple VI that converts distance from kilo meters to meters. Open a new blank VI. Press **Ctrl+T**.

Right click in front panel. Select **Numeric>>Numeric Control** and drop it in front panel.

Right click in front panel. Select **Numeric>>Numeric Indicator** and drop it in front panel.

Double click on the name of control and indicator and change the names of control as kilo meter and indicator as miles.

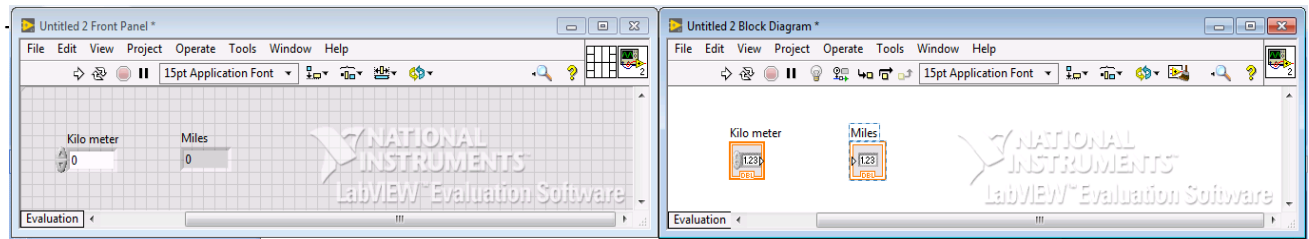


Figure 1.21: Front panel and block diagram without coding

To convert kilo meters to meters, we need to multiply kilo meters with 0.62131. Hence, this task should be performed in the block diagram by the following steps.

Step 1: Right click in block diagram. Select **Numeric>>Multiply** and drop it in block diagram.

To display the name of the function, right click on the function and select **Visible Items>>Label**.

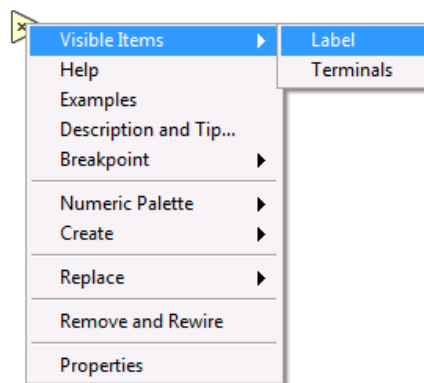



Figure 1.22: Label visibility of a function

Step 2: Right click in block diagram. Select **Numeric>> Numeric Constant** and drop it in block diagram. To display the name of the function, right click on the function and select **Visible Items>>Label**. Now enter the name for the constant.

Step 3: Connect all the blocks using wiring tool as shown in the block diagram below. Move the cursor to any of the block diagram and when the cursor changes to the icon  wiring tool is in operation. When the mouse hovers over the exit or entry point of a terminal or

over a wire, wiring tool is accessed by the cursor. To wire blocks together, pass the wiring tool over the first terminal, click, pass the cursor the second terminal, and click again.

It can be observed that the function “Multiply” has a dot at the bottom entry terminal. This is because one terminal of the function “Multiply” is of data type “DBL” and the other terminal is of data type “I32”. The function “Multiply” coerces the smaller representation “I32” to wider representation “DBL” before execution and Labview places a coercion dot on the terminal where the conversion takes place. This is inefficient programming.

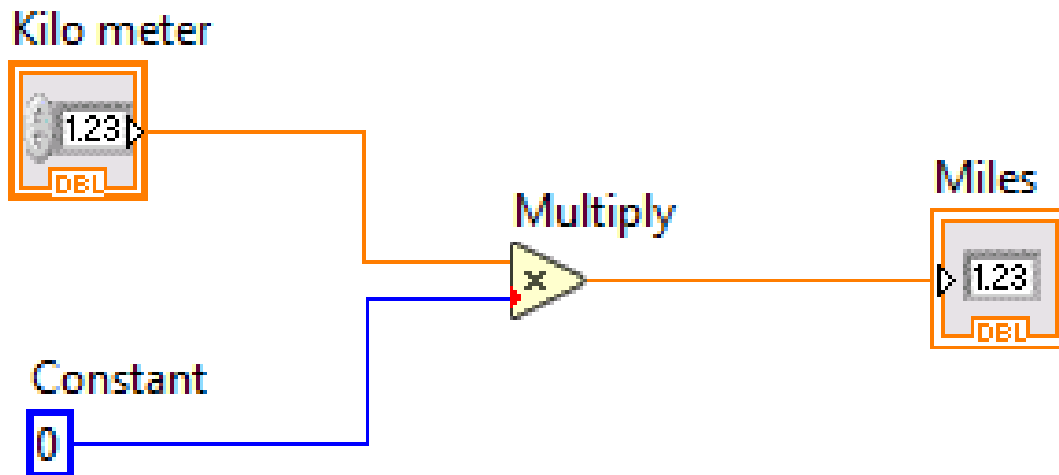


Figure 1.23: Block diagram with coercion error

Step 4: To increase the efficiency of programming, convert the constant from “I32” to “DBL” by right clicking on function “constant” and selecting Representation>>DBL.

Step 5: Assign the value “0.621371” to the constant by double clicking on the value and editing it.

Step 6: Save the VI.

Step 7: Enter a value in the control “kilo meter” and run VI. The corresponding value of miles is displayed in the indicator “Miles”. The front panel and block diagram appears as shown below:

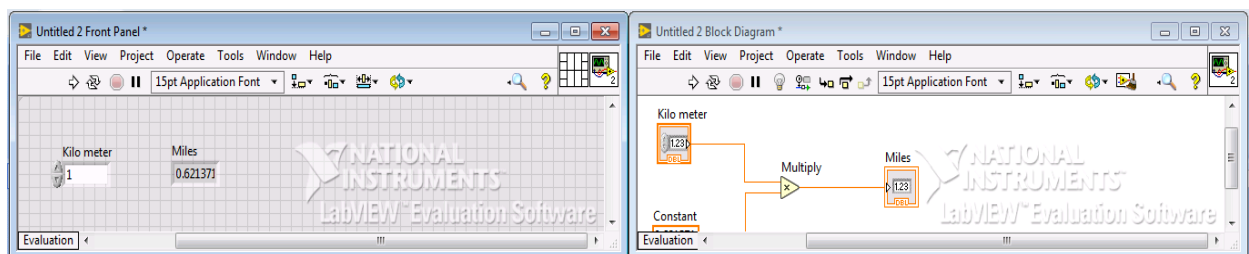


Figure 1.24: Conversion of kilometers to meters

2. Basic Examples

2.1 Average of Three Numbers

Task: Compute the average of three given numbers.

Open a new VI. Press **Ctrl+T**.

Right click in front panel. Select **Numeric>>Numeric Control** and drop it in front panel.

Repeat this and drop two more **Numeric Control** in the front panel.

Right click in front panel. Select **Numeric>>Numeric Indicator** and drop it in front panel.

Right click in block diagram. Select **Numeric>>Compound Arithmetic** and drop it in block diagram.

Right click in block diagram. Select **Numeric>>Numeric Constant** and drop it in block diagram.

Right click in block diagram. Select **Numeric>>Divide** and drop it in block diagram.

Connect all the blocks as shown below using wiring tool.

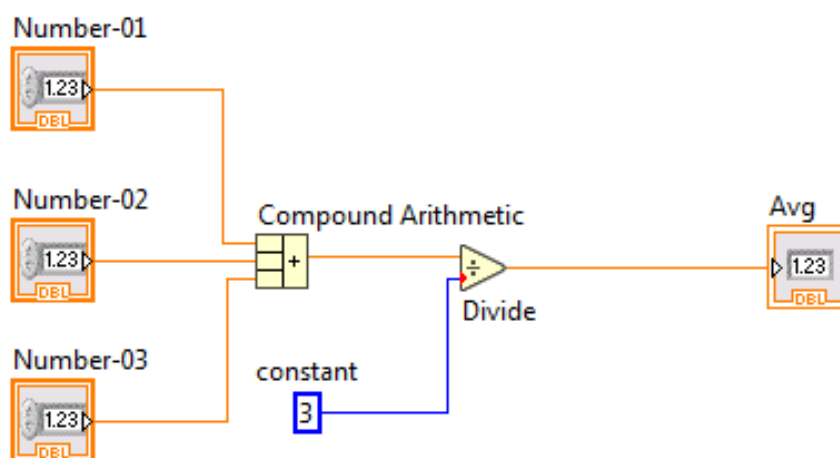


Figure 2.1: Average of three numbers with coercion dot (block diagram)

In the front panel, enter the input values for which average value is to be found.

Execute the VI.

Average of the three numbers will be displayed in the indicator. However, coercion dot appears at the input terminal of the **Divide** block in the block diagram. This coercion dot appeared because the constant was of "I32" representation and sum of the three numbers is of data type "DBL". To avoid this coercion dot, Right click in block diagram. Select **Numeric**

>> **Conversion >> To Double Precision Float** and drop it in block diagram. Connect this block in between constant and divide block as shown below:

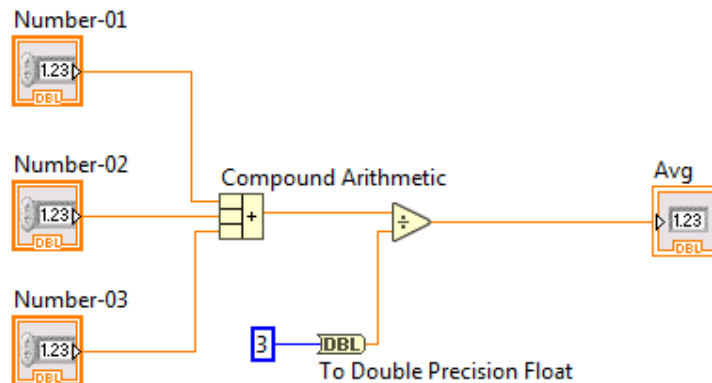


Figure 2.2: Average of three numbers without coercion dot (block diagram)

Click **Run Continuously**.

The average of the three numbers is displayed in the indicator.

2.2 Pressure and Temperature indicator

Task: One LED should glow when pressure is greater than 70 and another LED should glow when the temperature is between 20 and 80.

Open a new VI. Press **Ctrl+T**.

Right click in front panel. Select **Numeric>>Numeric Control** and drop two numeric controls in front panel.

Right click in front panel. Select **Boolean>>Round LED** and drop two LEDs in front panel.

Right click in block diagram. Select **Comparison>> Greater?** and drop it in block diagram.

Select **Comparison>> Greater Or equal?** and drop it in block diagram. Select **Comparison>> Less Or equal?** and drop it in block diagram.

Right click in block diagram. Select **Numeric>> Numeric Constant** and drop 3 constants in block diagram.

Right click in block diagram. Select **Boolean>> And** and drop logic gate in block diagram.

Connect all the blocks as shown below using wiring tool. Rename the controls, indicators and blocks. The front panel and block diagram appear as shown below:

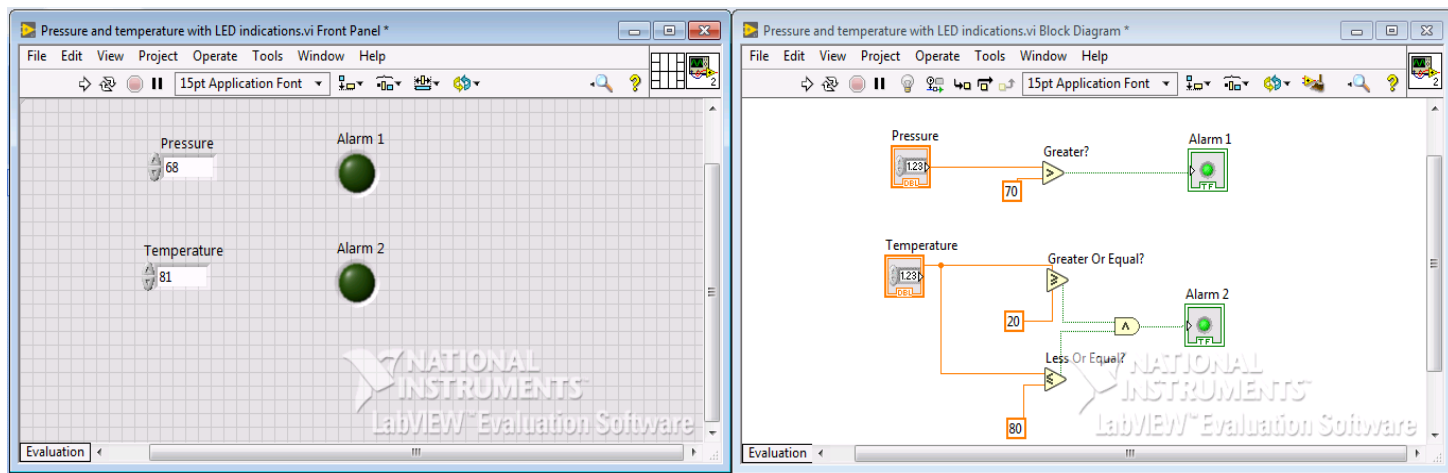


Figure 2.3: Pressure and Temperature indicator

In the front panel, enter the input values.

Give values to the controls and execute the VI.

The alarm1 glows only when the pressure entered is above 70 and the alarm2 glows only when the temperature is between 20 and 80.

2.3 Inrange and Coerce

Task: To glow LED when the number inputted is in a specified range. Also coerce the input to the value within limits, when the inputted value is out of specified range.

Open a new VI. Press **Ctrl+T**.

Right click in front panel. Select **Numeric>>Numeric Control** and drop two numeric controls in front panel.

Right click in front panel. Select **Boolean>>Round LED** and drop LED in front panel.

Right click in front panel. Select **Numeric>>Numeric Indicator** and drop control in front panel.

Right click in block diagram. Select **Comparison>> In Range and Coerce** and drop it in block diagram.

Right click in block diagram. Select **Numeric>> Numeric Constant** and drop two numeric constants in block diagram.

Connect all the blocks as shown below using wiring tool. Rename the controls, indicators and blocks. The front panel and block diagram appear as shown below:

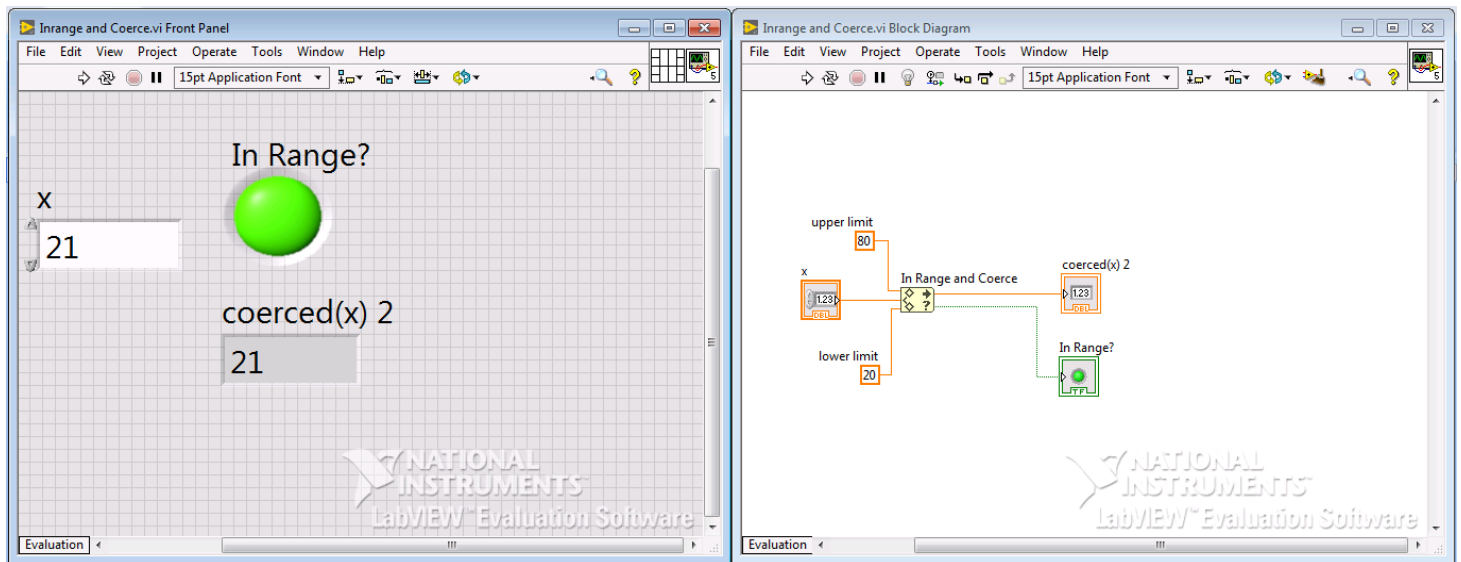


Figure 2.4: Inrange and Coerce

In the front panel, enter the input values.

Give values to the controls and execute the VI.

The LED glows when the value entered in the control **X** is between 20 and 80 and the indicator shows the value entered in the control **X**. However, if the value is out of the range, then the LED doesn't glow and the indicator shows the lower limit or upper value.

3. Exercises on Structures

3.1 Introduction

Structures contain the graphical code and they control the code inside the structure i.e., structures control how and when the code inside the structure is to be run. Basic structures in LabVIEW are Case Structures, While Loop and For Loop.

3.2 Case Structure

Case structures are used to selectively execute a code based on a condition.

3.2.1 Numerical Case

Task: To classify the inputted value of frequency based on pre defined criteria.

Open a new VI. Press **Ctrl+T**.

Right click in front panel. Select **Numeric>>Numeric Control** and drop it in front panel.

Right click in front panel. Select **Numeric>>Numeric Indicator** and drop control in front panel.

Double click in front panel and write down the criterion for classifying the frequency, which is as follows:

- <100Hz - Low frequency
- 101 to 1000Hz - Medium frequency
- >1001 - High frequency

Right click in block diagram. Select **Structures>>Case Structure**. Keep mouse holding and then drag the mouse to create a structure in the block diagram. Rename the control and indicator.

The front panel and block diagram appear as shown below:

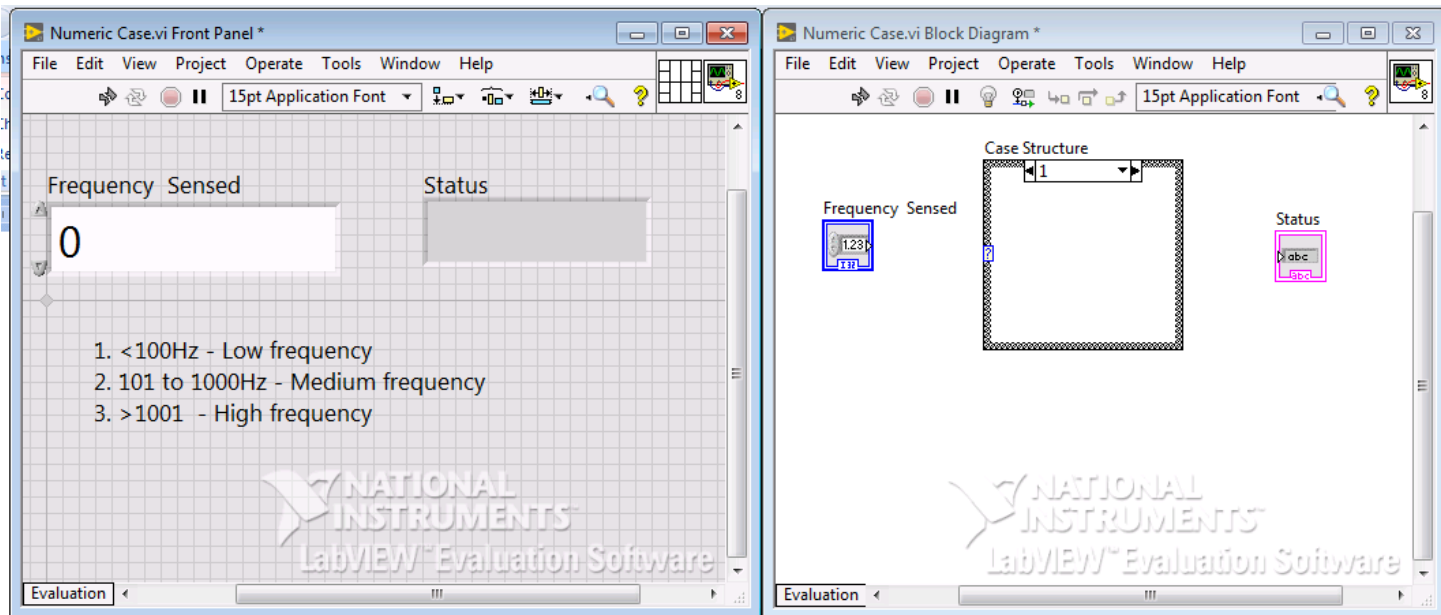


Figure 3.1: Preview of front panel and block diagram before coding and wiring

Now connect the **Numeric Control** to the input terminal of the case structure (figure 3.2 a) using wiring tool at the position where “question mark” is present. Click on the icon ▼ in **selector label** (figure 3.2 b) and select **0, Default**. Click inside the **selector label**, delete **0, Default** and type **..100**

Right click in block diagram, select **String>>String Constant** and drop it in inside the case structure. Type **Low frequency** inside the **String Constant** and connect it to the case structure as shown in figure 3.2 c.

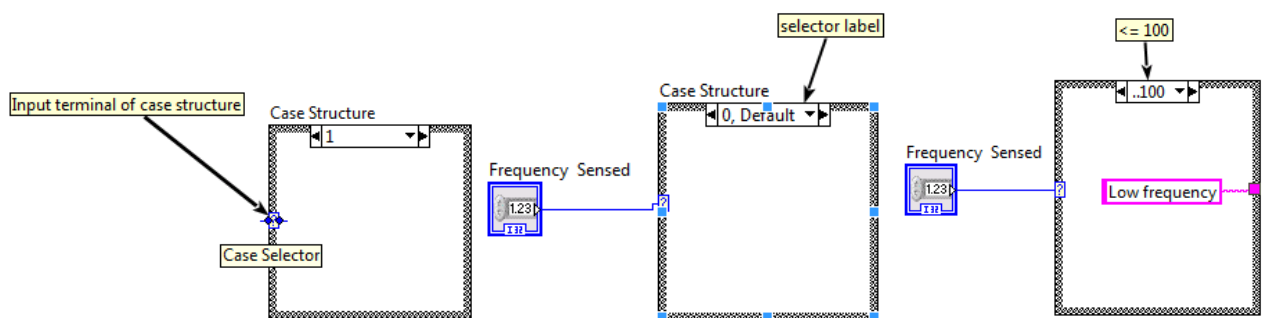


Figure 3.2: a). Case selector b) Selector label c) Logic for case “Low frequency”

Next, right click on the icon ▼ and select **Add Case After** and type **101..1000** in the **Selector label**. Right click in block diagram, select **String>>String Constant** and drop it in inside the case structure. Type **Medium frequency** inside the **String Constant** and connect it to the case structure as shown in figure 3.3.

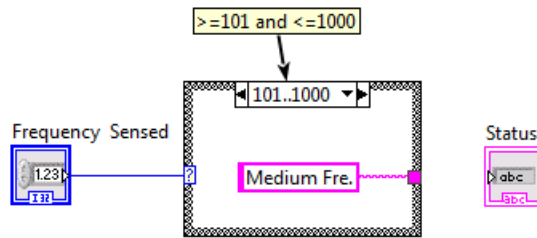


Figure 3.3: Logic for case “Medium frequency”

Again, right click on the icon ▼ and select **Add Case After** and type **1000..** in the **Selector label**. Right click in block diagram, select **String>>String Constant** and drop it in inside the case structure. Type **High frequency** inside the **String Constant** and connect it to the case structure as shown in figure 3.4.

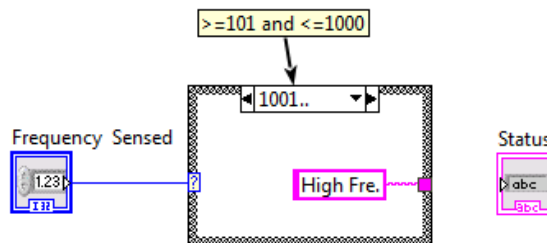


Figure 3.4: Logic for case “High frequency”

Connect the input block of the string indicator to the case structure and the final block diagram appears as shown in figure 3.5.

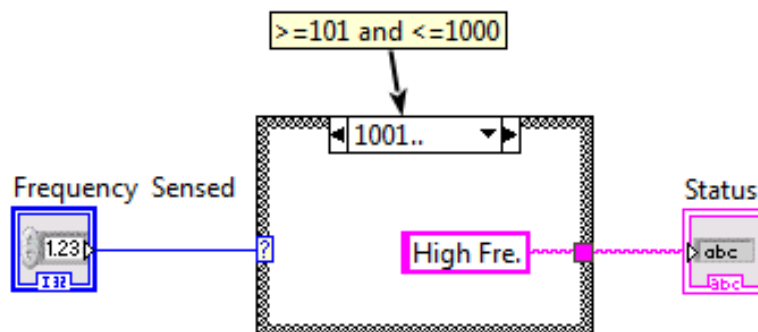


Figure 3.4: Block diagram for numerical case

Enter any value of frequency in the numerical control and run VI. The corresponding category of frequency will be displayed in the string indicator.

3.2.2 Boolean Case

Task: To perform addition or subtraction of any two numbers.

Open a new VI. Press **Ctrl+T**.

Right click in front panel. Select **Numeric>>Numeric Control** and drop two controls in front panel.

Right click in front panel. Select **Numeric>>Numeric Indicator** and drop it in front panel.

Right click in front panel. Select **Boolean>>Push Button** and drop it in front panel.

Add **Case Structure** in the block diagram.

The front panel and block diagram appear as shown below:

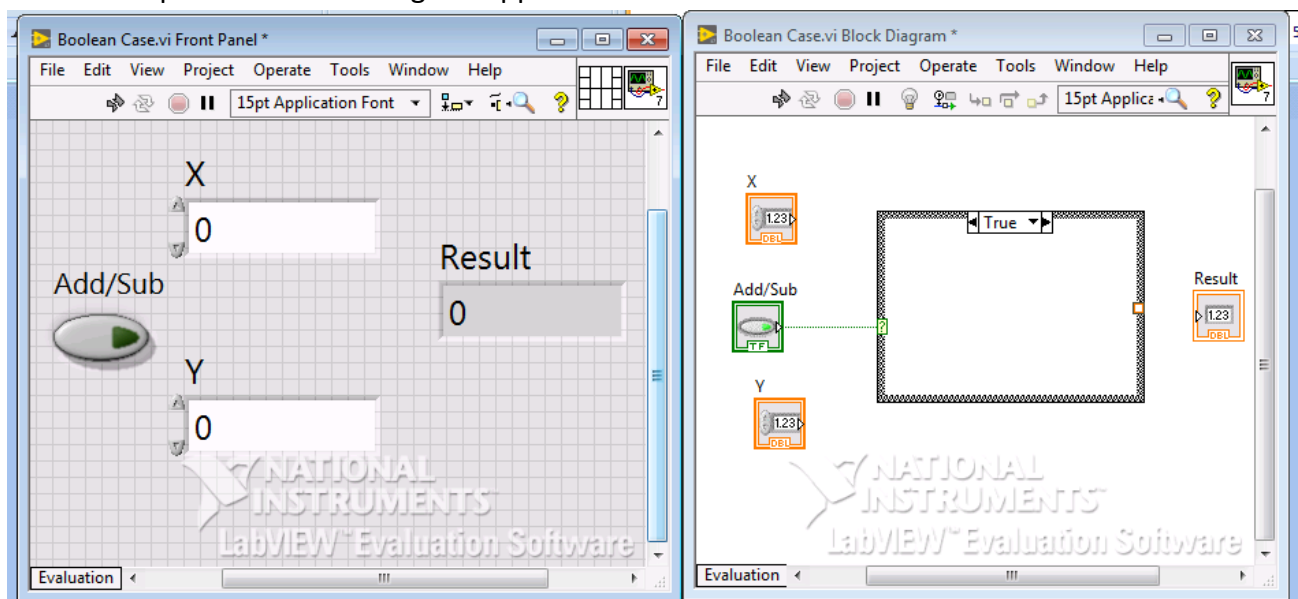


Figure 3.5: Front panel and block diagram of Boolean case before wiring

Now write the logic for case “SUB” as shown in figure 3.6

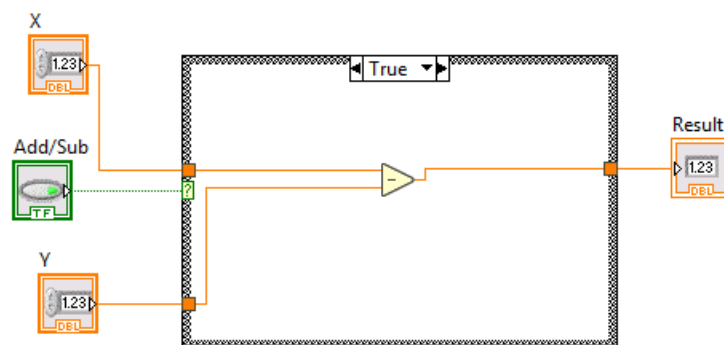


Figure 3.6: Logic for “SUB” case

Now write the logic for case “ADD” as shown in figure 3.7

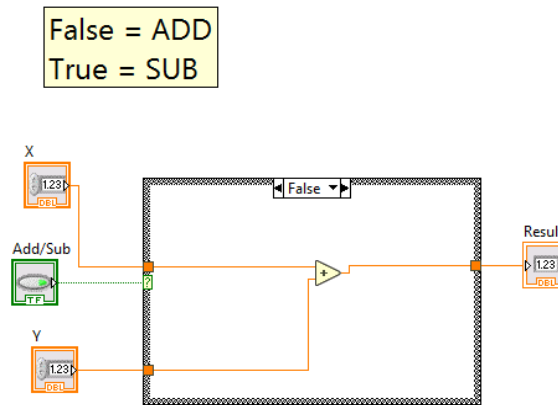


Figure 3.7: Logic for “ADD” case

Run VI. When the push button is pressed, subtraction of two numbers is performed and the result will be displayed in the indicator. When the push button is not pressed, addition of two numbers is performed and the result will be displayed in the indicator.

3.2.3 Enum Case

Task: To perform addition, subtraction, multiplication or division of any two numbers using Enum.

Open a new VI. Press **Ctrl+T**.

Right click in front panel. Select **Numeric>>Numeric Control** and drop two controls in front panel.

Right click in front panel. Select **Numeric>>Numeric Indicator** and drop it in front panel.

Right click in front panel. Select **Ring & Enum>>Enum** and drop it in front panel.

Right click on Enum. Select **Edit items...** and the window shown in figure 3.8 appears.

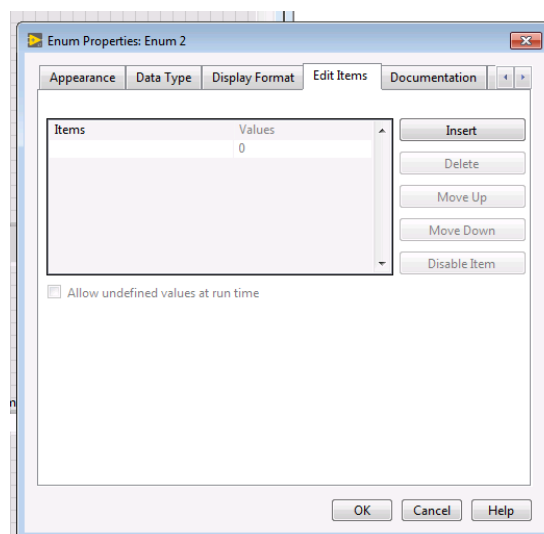


Figure 3.7: Enum properties

Click **Insert**. Type **ADD** under “items” and press enter. Repeat the step and insert **SUB**, **MUL** and **DIV**. Press OK.

The front panel and block diagram appear as shown in figure 3.8:

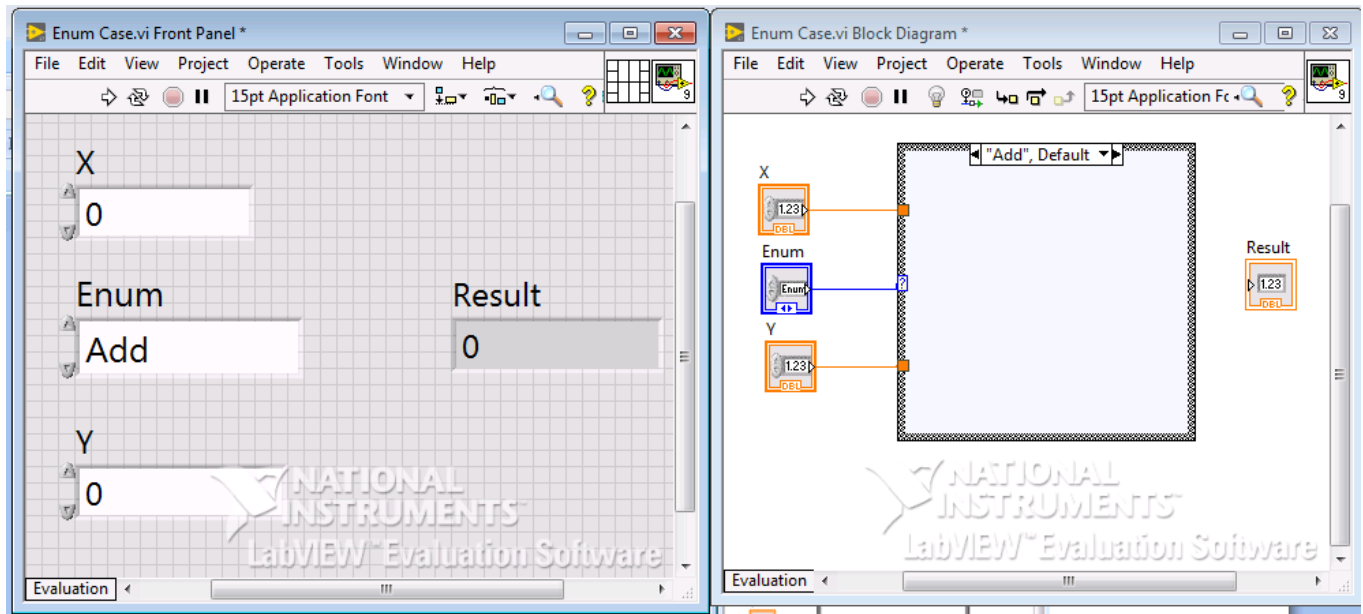


Figure 3.8: Front panel and block diagram for Enum case without logic

Now, write logic for “ADD” case as shown in figures 3.9. Similarly, add logic for cases **SUB**, **MUL** and **DIV**.

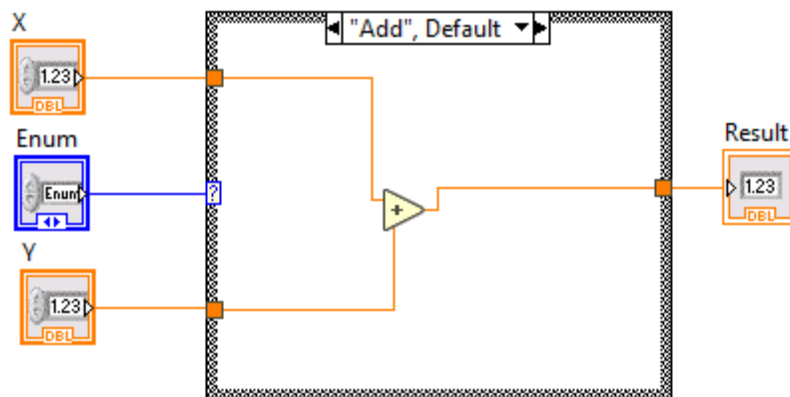


Figure 3.9: Logic for Enum “ADD” case

Select the required arithmetic operation in Enum and run VI. The corresponding arithmetic operation is performed and the result will be displayed in the indicator.

3.2.4 Case Structure with String

Task: To perform addition, subtraction, multiplication or division of any two numbers by using a command in the String.

Open a new VI. Press **Ctrl+T**.

Right click in front panel. Select **Numeric>>Numeric Control** and drop two controls in front panel.

Right click in front panel. Select **Numeric>>Numeric Indicator** and drop it in front panel.

Right click in front panel. Select **String & Path>>String Control** and drop it in front panel.

Complete logic for all the arithmetic operations. The front panel and block diagram with MUL case appear as shown in figure 3.10:

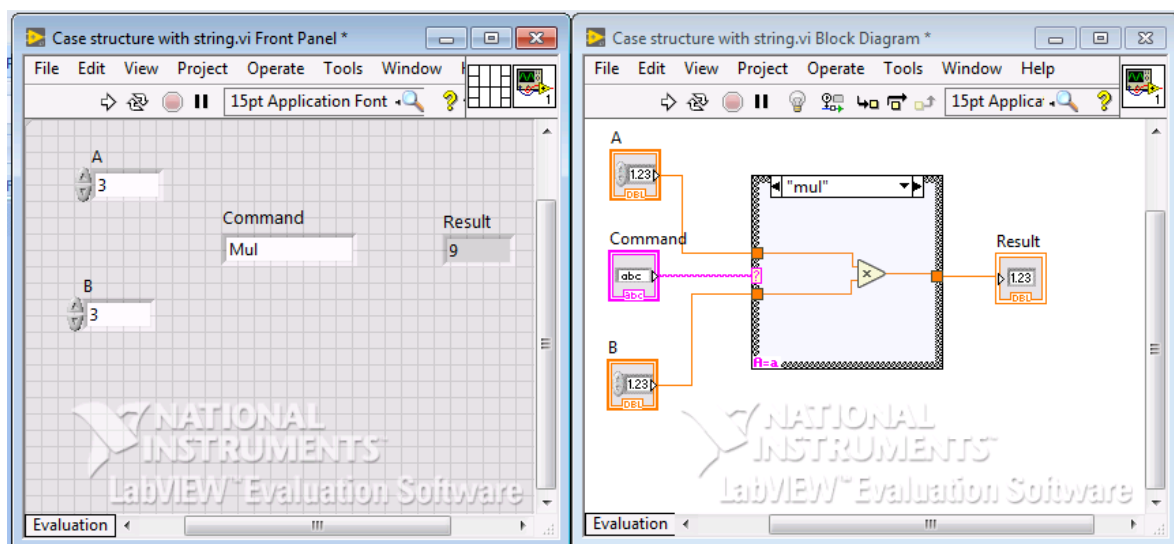


Figure 3.10: Logic for “MUL” case with string

3.2.5 Case Structure with Ring

Task: To perform addition, subtraction or multiplication of any two numbers using **Ring**.

Open a new VI. Press **Ctrl+T**.

Right click in front panel. Select **Numeric>>Numeric Control** and drop two controls in front panel.

Right click in front panel. Select **Numeric>>Numeric Indicator** and drop it in front panel.

Right click in front panel. Select **Ring & Enum>>Text Ring** and drop it in front panel.

Right click on Ring. Select **Edit items...** and the window shown in figure 3.11 appear.

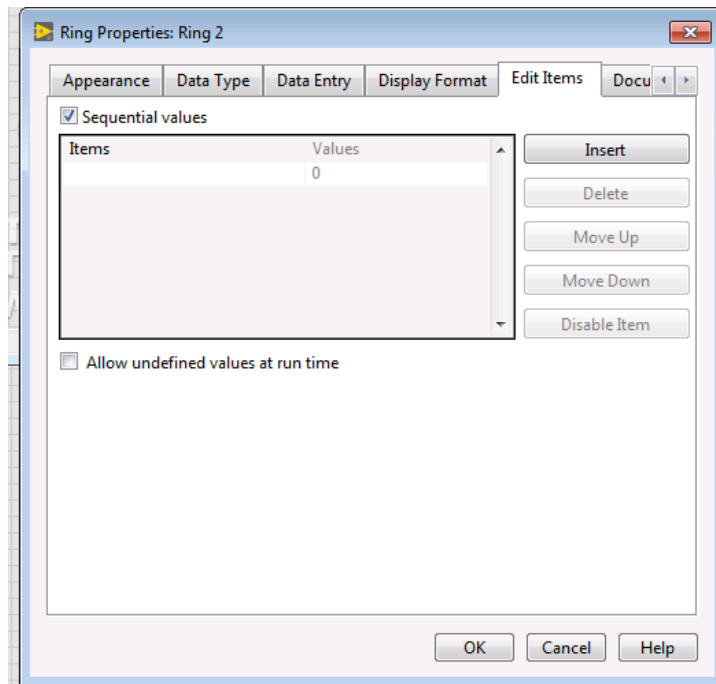


Figure 3.11: Ring properties

Unlike Enum which has fixed sequential values, Ring can have random values.

In this example, arithmetic operation “ADD” is given value “100”, “SUB” is given value “200” and “MUL” is given value “300”.

Click **Insert**. Type **ADD** under “items”, assign value “100” to item and press enter.

Repeat the step of and insert **SUB** and **MUL**. Press OK.

Now, write logic for “ADD” case as shown in figures 3.12. Similarly, add logic for cases **SUB**, and **MUL**.

The front panel and block diagram for MUL case appear as shown in figure 3.12:

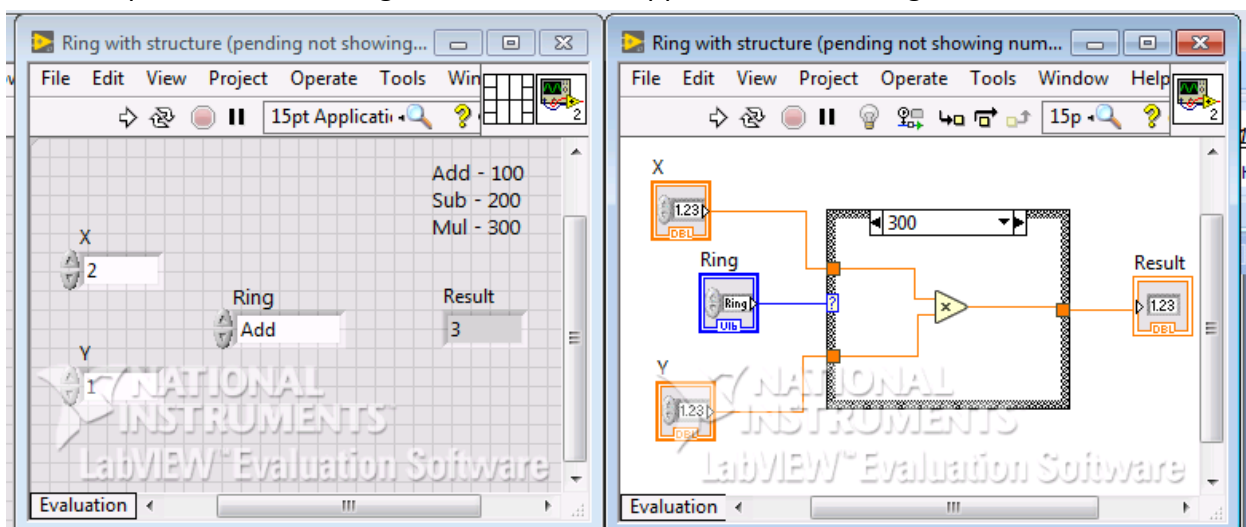


Figure 3.12: Front panel and block diagram for “MUL” case using Ring

Type the required arithmetic operation in Ring and run VI. The corresponding arithmetic operation is performed and the result will be displayed in the indicator.

3.3 While Loop

While loops are used to repeat a code until a certain condition is met or start a code after certain condition is met.

3.3.1 Fill water in a tank up to a desired level using while loop

Open a new VI. Press **Ctrl+T**.

Right click in front panel. Select **Numeric>>Tank** and drop it in front panel. To change the value of level in tank, double click on the highest value and enter required value.

Right click in block diagram. Select **Structures>>While Loop**. Keep mouse holding and then drag the mouse to create a **while loop** in the block diagram.

The front panel and block diagram appear as shown in 3.13:

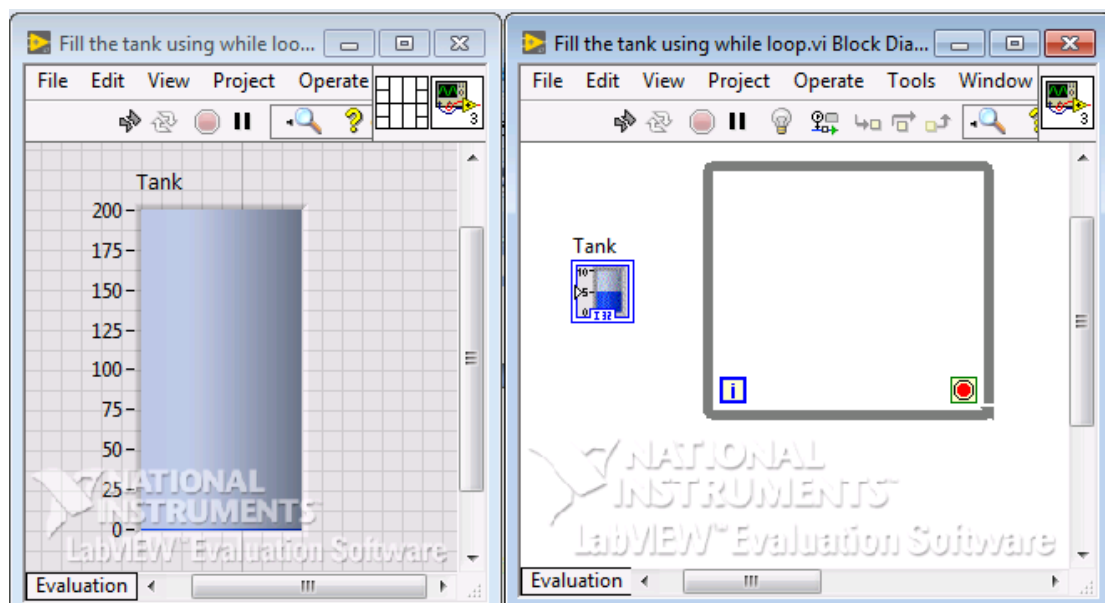


Figure 3.13: Front panel and block diagram for the water tank model without logic

In block diagram, drag the **Tank** block into **While** loop.

Select **Comparison>> Greater Or equal?** and drop it in **While** loop. Select **Timing>>Wait** and drop it in **While** loop. Select **Numeric>> Numeric Constant** and drop two numeric constants in **While** loop. Assign values to **Numeric Constant** and make connections as shown in figure 3.14 using wiring tool. In this example, the value of one **Numeric Constant** is set to “200”,

which is the time delay between each iteration of while loop. If this delay is not considered, then LabVIEW consumes high CPU usage. The value of another **Numeric Constant** is set to “151” and this is the tank level at which the while loop stops and filling of water in the tank is stopped.

If coercion dot appears on the tank block, then convert tank block to “I32” by right clicking on tank block and selecting **Representation>>I32**.

If coercion dot appears on the **Numeric Constant**, then convert **Numeric Constant** to “U32” by right clicking on **Numeric Constant** and selecting **Representation>>U32**.

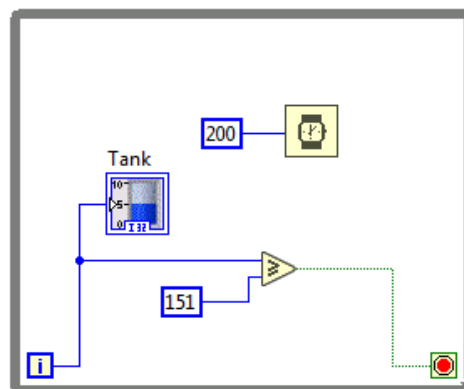


Figure 3.14: Block diagram for the water tank model with logic

Run VI. It can be observed that water is filled in the tank till the value “150” is reached and thereafter the while loop stops and filling of water in the tank is stopped.

3.4 Flat Sequence Structure

In certain cases, a function need to be executed first and after the execution of the first function, then only second or another function should be executed. For such cases, sequence structure is preferred. Flat sequence structure consists of several frames that are executed in order from left to right.

Task: To perform arithmetic operations on two numbers such that sum of the two numbers to be executed first, then subtraction and then multiplication.

Right click in front panel. Select **Numeric>>Numeric Control** and drop two numeric controls in front panel.

Right click in front panel. Select **Numeric>>Numeric Indicator** and in front panel.

Right click in block diagram. Select **Structures>>Flat Sequence**. Keep mouse holding and then drag the mouse to create a **Flat Sequence** in the block diagram. Drag the **Numeric Indicator** into the **Flat Sequence**.

Select **Timing>>Wait** and drop it in **Flat Sequence**. Select **Numeric>> Numeric Constant** and drop numeric constant in **Flat Sequence**. Assign value to **Numeric Constant**. Complete logic for “ADD” case. Make connections using wiring tool. The front panel and block diagram appear as shown in 3.15:

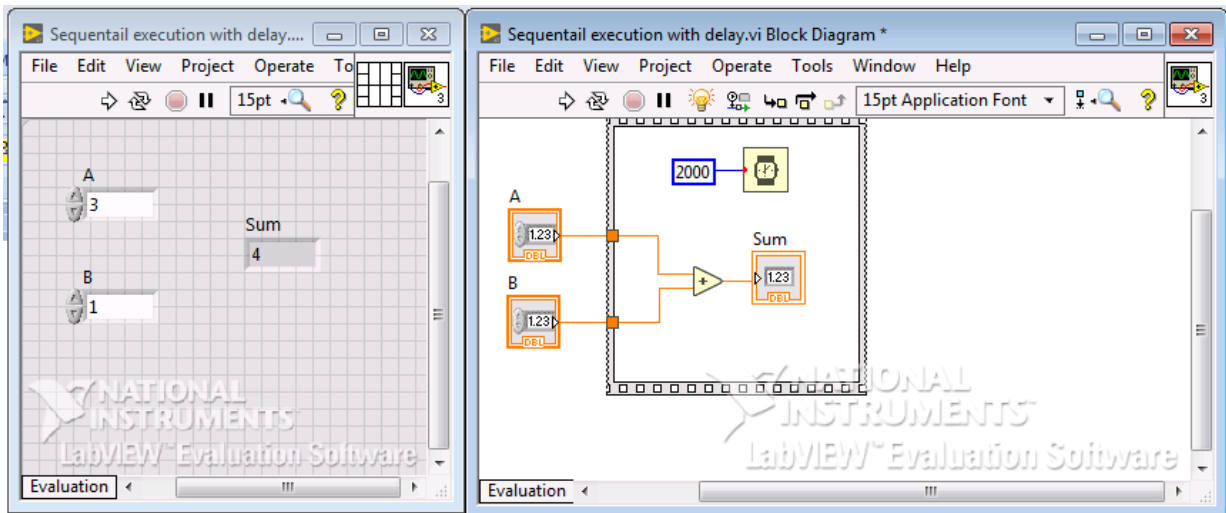


Figure 3.15: Front panel and block diagram with single in frame flat sequence

Right click on **Flat Sequence** and select **Add Frame After**. Resize the frame and add logic for “SUB” case using the same **Numeric Controls**. The front panel and block diagram appear as shown in 3.16:

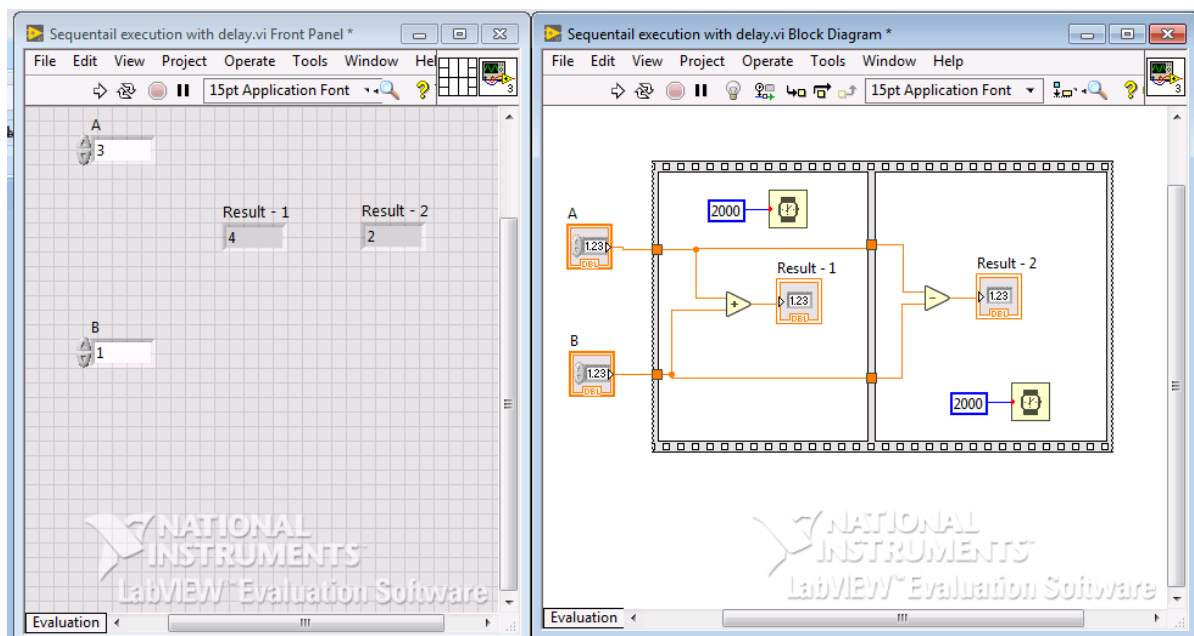


Figure 3.16: Front panel and block diagram with two frames in flat sequence

Right click on **Flat Sequence** and again select **Add Frame After**. Resize the frame and add logic for “MUL” case using the same **Numeric Controls**. The front panel and block diagram appear as shown in 3.17:

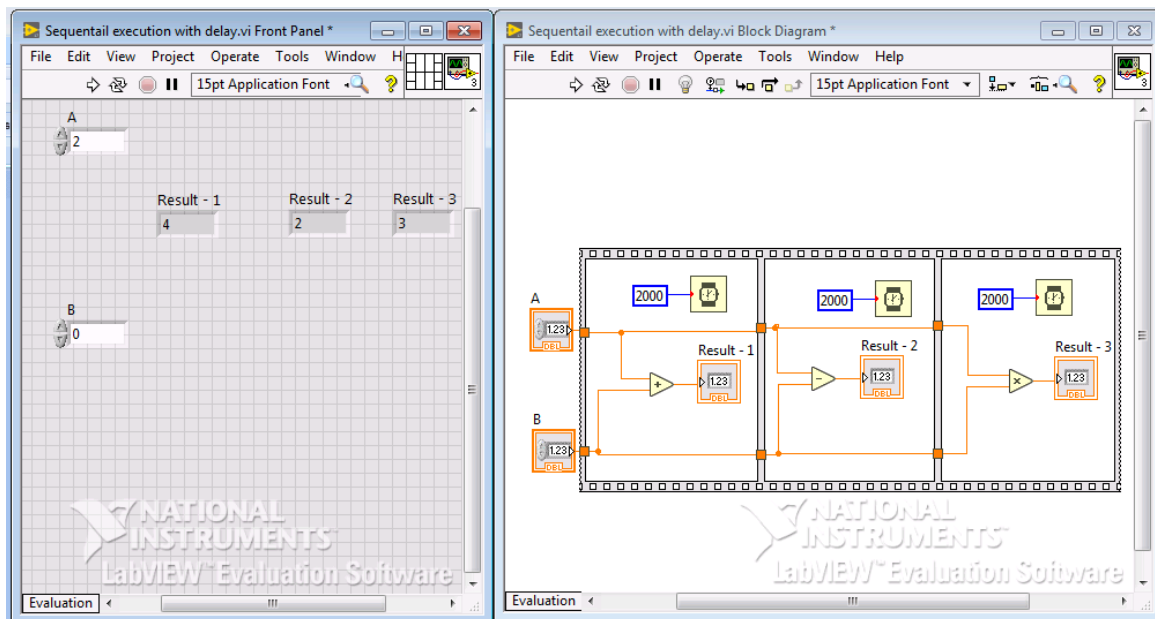


Figure 3.17: Front panel and block diagram with three frames in flat sequence

Assign values to the **Numeric Controls**. Run VI. It can be observed that the sum of the two numbers displays first, then subtraction and finally multiplication.

3.5 Sequence Structure with Stacking

In sequence structure, to reduce the amount of space consumed in the block diagram stacking concept is used. In this concept, multiple frames can be created in a single frame just by giving number to the frame in **Stacked Sequence Structure**.

Task: To perform arithmetic operation $[(X+Y)*Z]-A$ where X, Y, Z and A are the inputted values. The task should be performed using **Stacked Sequence Structure**.

Right click in front panel. Select **Numeric>>Numeric Control** and drop two numeric controls in front panel.

Right click in block diagram. Select **Structures>>Flat Sequence**. Keep mouse holding and then drag the mouse to create a **Flat Sequence** in the block diagram. Drag the two **Numeric Controls** into the **Flat Sequence**.

Right click inside the **Flat Sequence Structure** and select **Numeric>>Add** and drop it inside the **Flat Sequence Structure**.

Right click on the **Flat Sequence Structure** and select **Replace With Stacked Sequence** as shown in figure 3.18. Now the **Flat Sequence Structure** is converted to **Stacked Sequence Structure**.

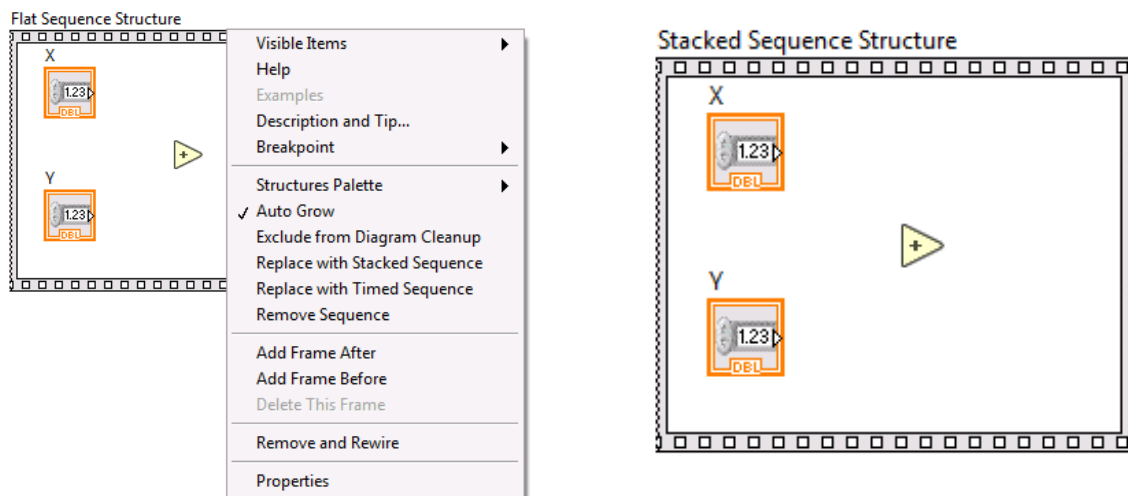


Figure 3.18: Conversion of Flat Sequence Structure to Stacked Sequence Structure

Connect **Numeric Controls X** and **Y** to function **ADD** using wiring tool.

Now to pass the value of (X+Y) to another frame, right click on the **Stacked Sequence Structure** and select **Add Sequence Local** as shown in figure 3.19.

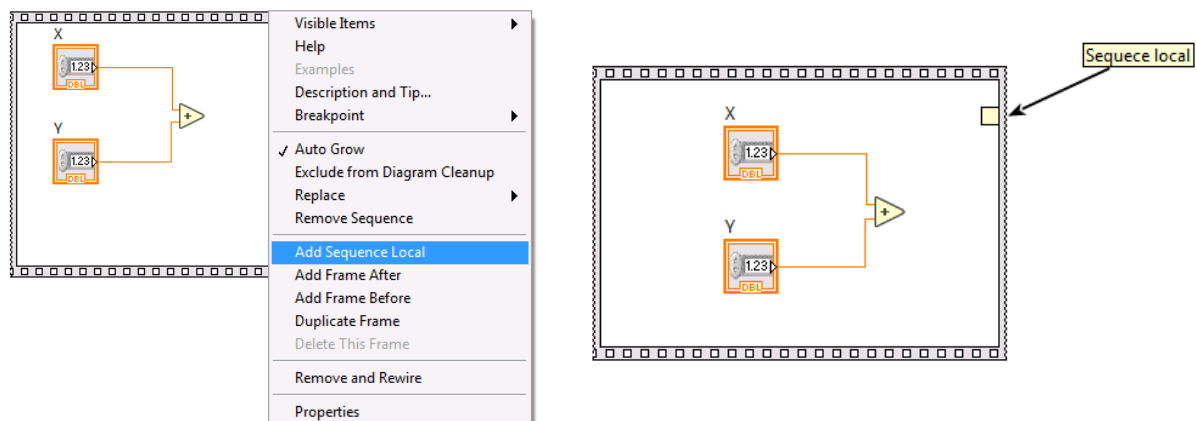


Figure 3.19: Adding “sequence local” to a frame in Stacked Sequence Structure

Connect the output terminal of **ADD** function to the **Sequence Local**. Double click and comment “0th frame” just next to **Sequence local** in block diagram so as to identify the frame to which it belongs. The completed **Stacked Sequence Structure** for **ADD** case (0th frame) is completed and is shown in figure 3.20.

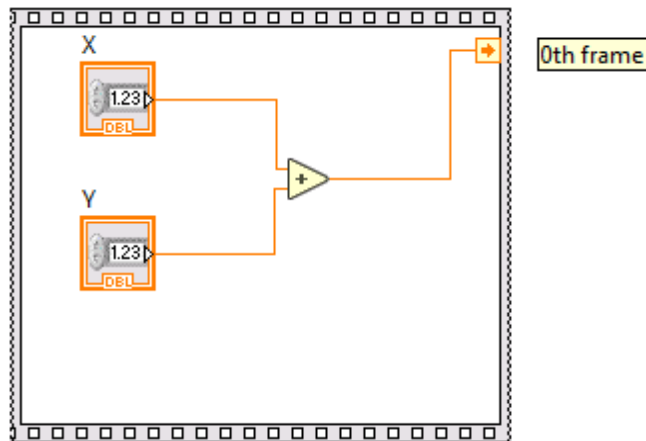


Figure 3.20: Stacked Sequence Structure in 0th frame

Next, right click on **Stacked Sequence Structure** and select **Add Frame After**. Frame 1 appears. Right click in front panel, select **Numeric>>Numeric Control** and drop it in front panel. Rename it as “Z”. Drop Numeric Control “Z” in the 1st frame. Right click on the **Stacked Sequence Structure** and select **Add Sequence Local**. Multiply the output of 0th frame to the Numeric Control “Z”. Complete connections using wiring tool. The output signal $\{(X+Y)*Z\}$ is connected to **Sequence local**. Double click and comment “1st frame” just next to **Sequence local** in block diagram so as to identify the frame to which it belongs. The block diagram in 1st frame appears as shown in figure 3.21.

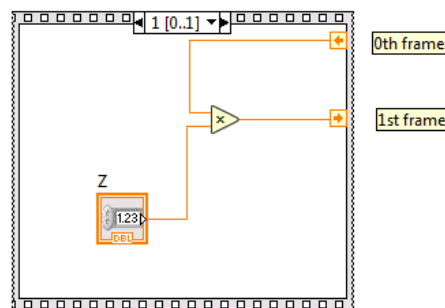


Figure 3.21: Stacked Sequence Structure in 1st frame

Again, right click on **Stacked Sequence Structure** and select **Add Frame After**. Frame 2 appears. Right click in front panel, select **Numeric>>Numeric Control** and drop it in front panel. Rename it as “A”. Drop Numeric Control “a” in the 2nd frame. Right click on the **Stacked Sequence Structure** and select **Add Sequence Local**. Subtract Numeric Control “A” from the output of 1st frame. Right click in front panel, select **Numeric>>Numeric Indicator** and drop it in front panel. Rename it as “Result”. Drop Numeric Indicator “Result” in the 2nd frame.

Complete connections using wiring tool. The output signal $[(X+Y)*Z]-A$ is now connected to Numeric Indicator “Result” as shown in figure 3.22.

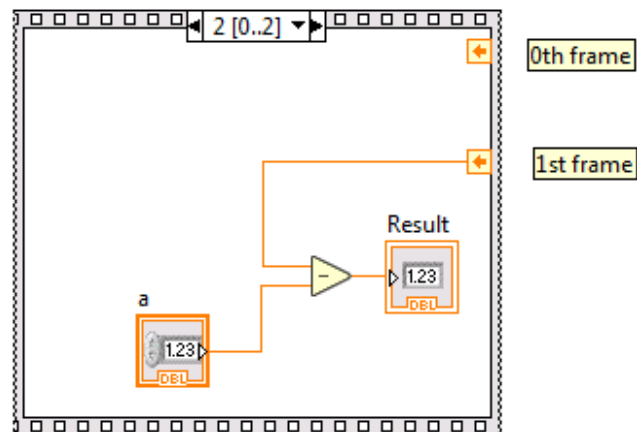


Figure 3.22: Stacked Sequence Structure in 2nd frame

The front panel and block diagram now appears as shown in figure 3.23.

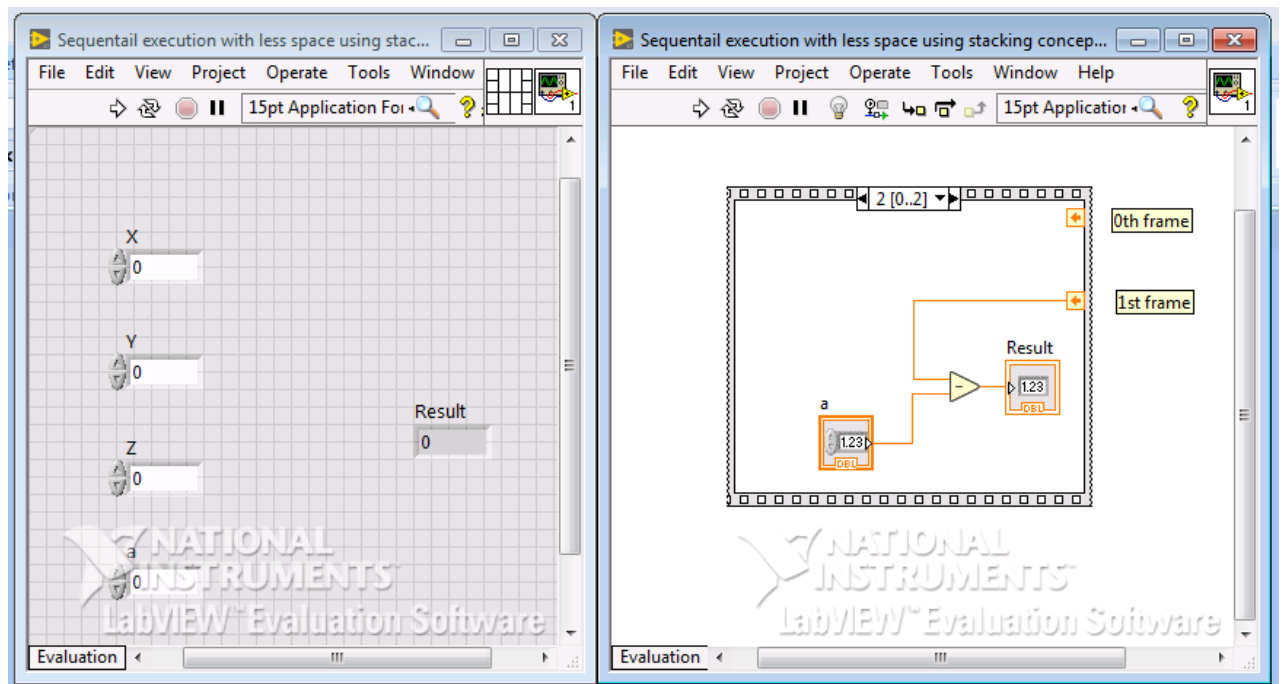


Figure 3.23: Front panel and block diagram of stacked sequence structure

Assign values to numeric controls X, Y, Z and A. Run VI. The result of the arithmetic operation $[(X+Y)*Z]-A$ is displayed in the numeric indicator “Result”.

3.6 Structure Tunnels

Tunnels pass data into or out of structures. Tunnel appears as a solid block on the border of the structure. The colour of the block is same as the colour of the data type wired into the tunnel. Whenever a data into a loop using tunnel, the loop will be executed only after data is arrived at the tunnel.

Task: To understand the behaviour of input and output tunnels let us design a simple VI in which the LED glows when the value of sensor is greater than specified threshold value.

Open a new VI. Press **Ctrl+T**.

Right click in front panel. Select **Numeric>>Numeric Control** and drop two numeric controls in front panel. Rename them as **Threshold** and **Sensor Data**.

Right click in front panel. Select **Boolean>>Round LED** and drop an LED in front panel.
Rename it as **Value Crossed?**

Right click in front panel. Select **Boolean>>Stop Button** and drop it in front panel.

Create while loop in the block diagram.

In while loop, right click and select **Numeric>>Numeric Constant** and drop it in the loop.

Assign a value to the **Numeric Constant**.

In while loop, right click and select **Timing>>Wait** and drop it in the loop.

In while loop, right click and select **Comparison>>Greater?** and drop it in the loop.

Make connections as shown in figure 3.24 using wiring tool.

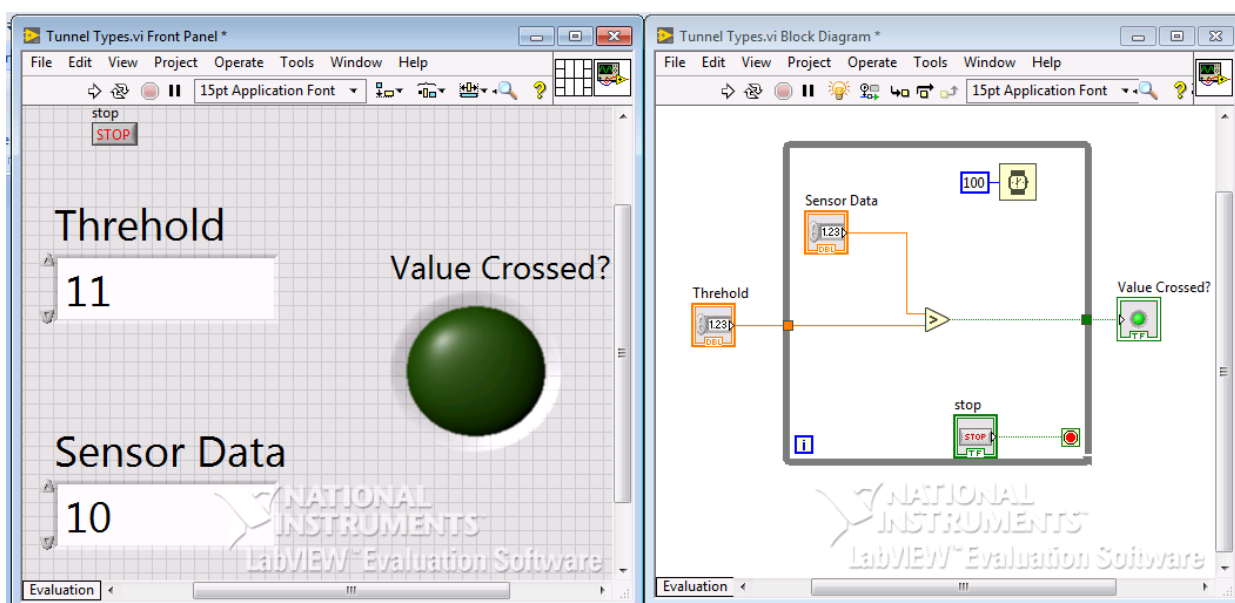


Figure 3.24: Structure tunnel

Assign value to the **Threshold** and click Run. Select **Highlight Execution**. Assign a value to **Sensor Data** which is less than **Threshold**. Now increase value in **Sensor Data**. As the value of **Sensor Data** crosses **Threshold** value the LED will not glow because of output tunnel effect. The LED glows only after the while loop is stopped using **Stop Button**. Also, once the execution is started any change made in the **Threshold** value will not get reflected in the control diagram because of input tunnel effect.

3.7 Loops and Structures

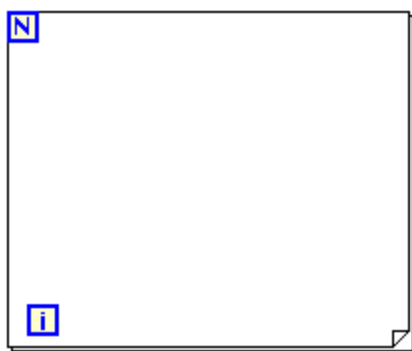
Generally for condition/decision making purpose following two things very useful

- I. LOOPS
 - i) FOR LOOP
 - ii) WHILE LOOP
- II. STRUCTURES
 - i) CASE STRUCTURE
 - ii) Sequence Structure
 - iii) Event Structure

The different Loops and Structures available are located in the “*Structures*” sub palette in the Functions palette on the Block Diagram.

i) FOR LOOP

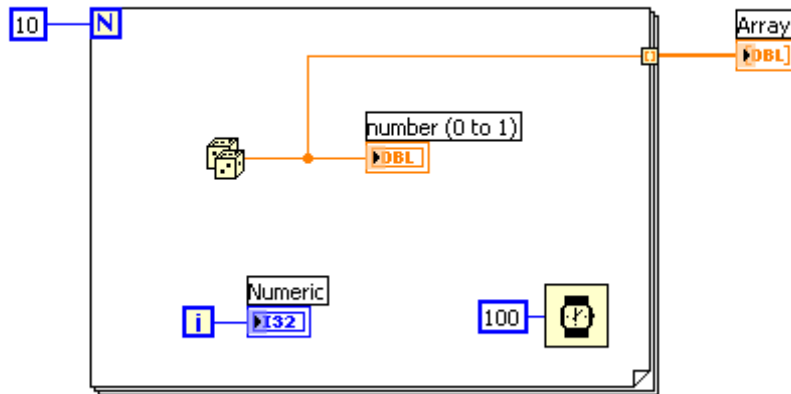
A For Loop executes a sub diagram a set number (N) of times. The Figure below shows an empty For Loop in LabVIEW.



A For loop executes its sub diagram n times, where N is the value wired to the count (N) terminal. The iteration (I) terminal provides the current loop iteration count, which ranges From 0 to N-1.

Iteration count value can be controlled in the loop by using delay time function. Default time delay function takes it time value in milli seconds.

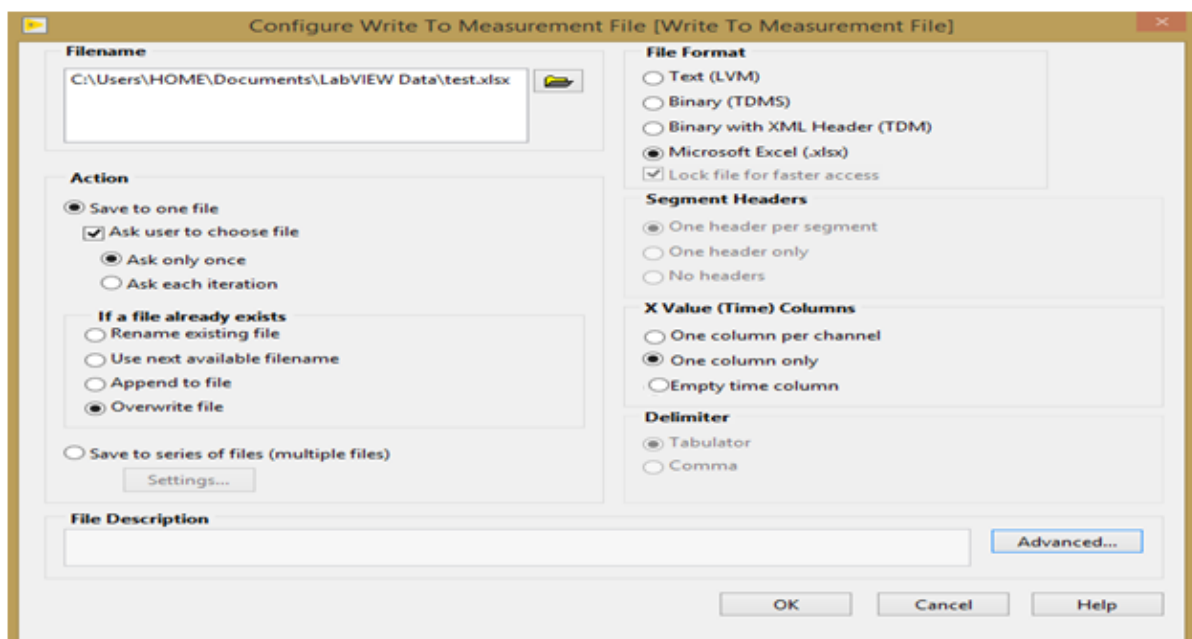
The following example uses a For Loop in order to create an array with 10 elements and fill it with random numbers.



3.8 File I/O operation:

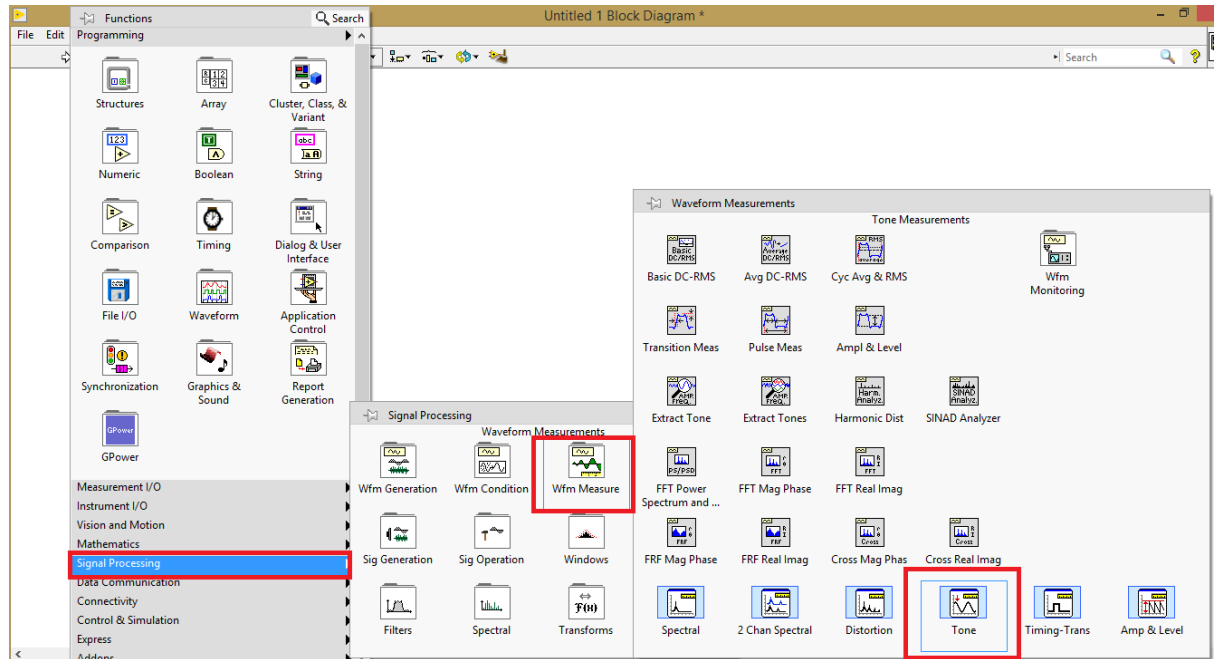
Select from block diagram R.C>> File I/O >> Write to measurement file.

In any specified program if any data is generated then that data can be stored by using File I/O write to measurement file. In this data can store in different formats. Like Microsoft Excel or TDMS or TEXT format. Below figure shows the selection of file to store the data in Excel format.



3.8 Tone Measurement

This block is useful to find the magnitude of the waveform, phase angle and frequency of the AC signal. In real time any AC signal from DAQ assistant is fed to tone measurement it will give the information (RMS value, phase and frequency). It can be select from block diagram R.C >> Signal Processing >> Waveform Measurements (Wfm Measure) >> Tone Measurement.



3.9 Charts and Graphs

In order to represent the data in graphically two methods are available

- 1) Charts
- 2) Graphs

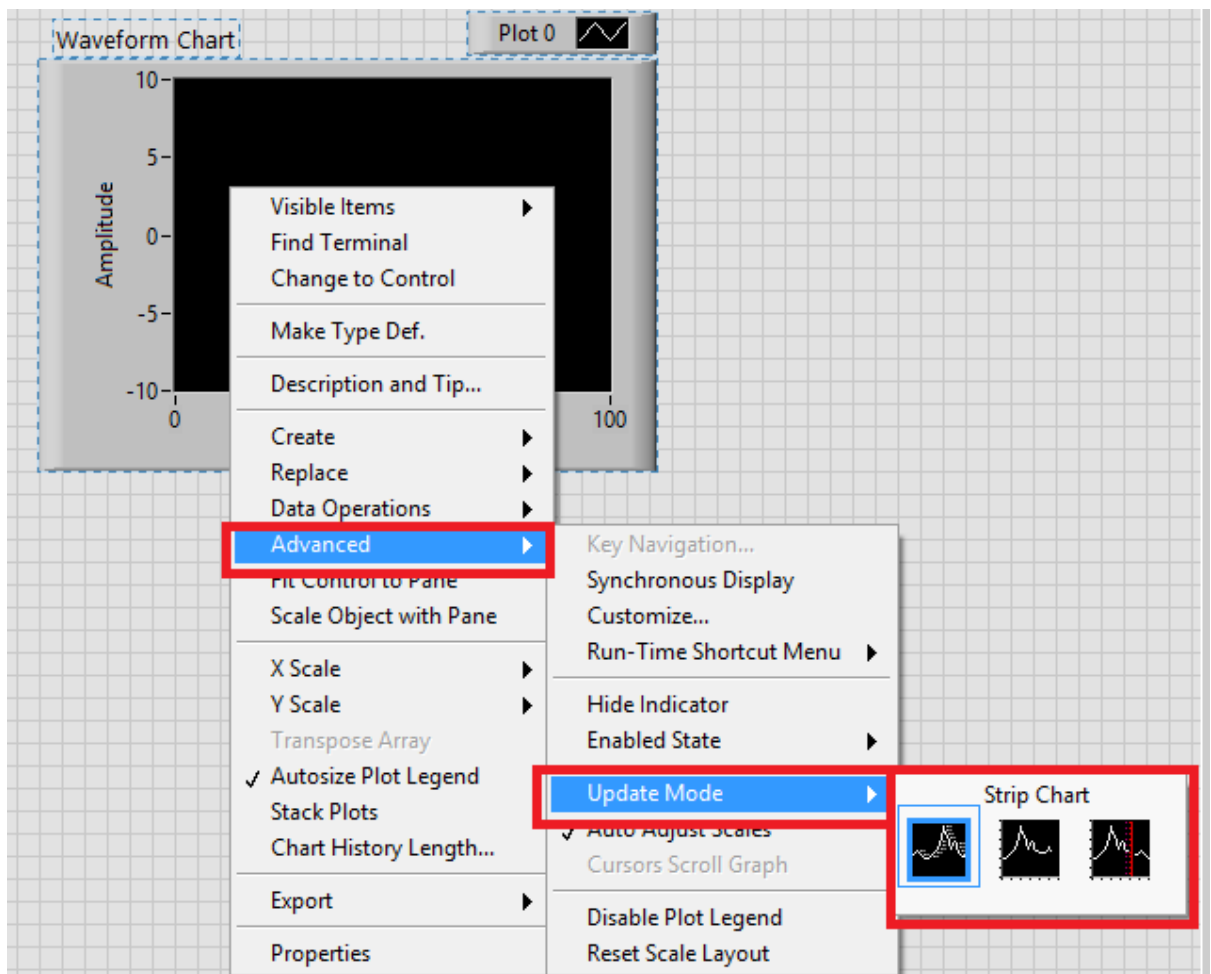
WAVEFORM CHART

It is a special type of numeric data indicator. It stores the previous data also. In order to indicate single value also it is useful.

This chart is available in front panel R.C >> Graph >> waveform Chart

In this three types of charts are there they are

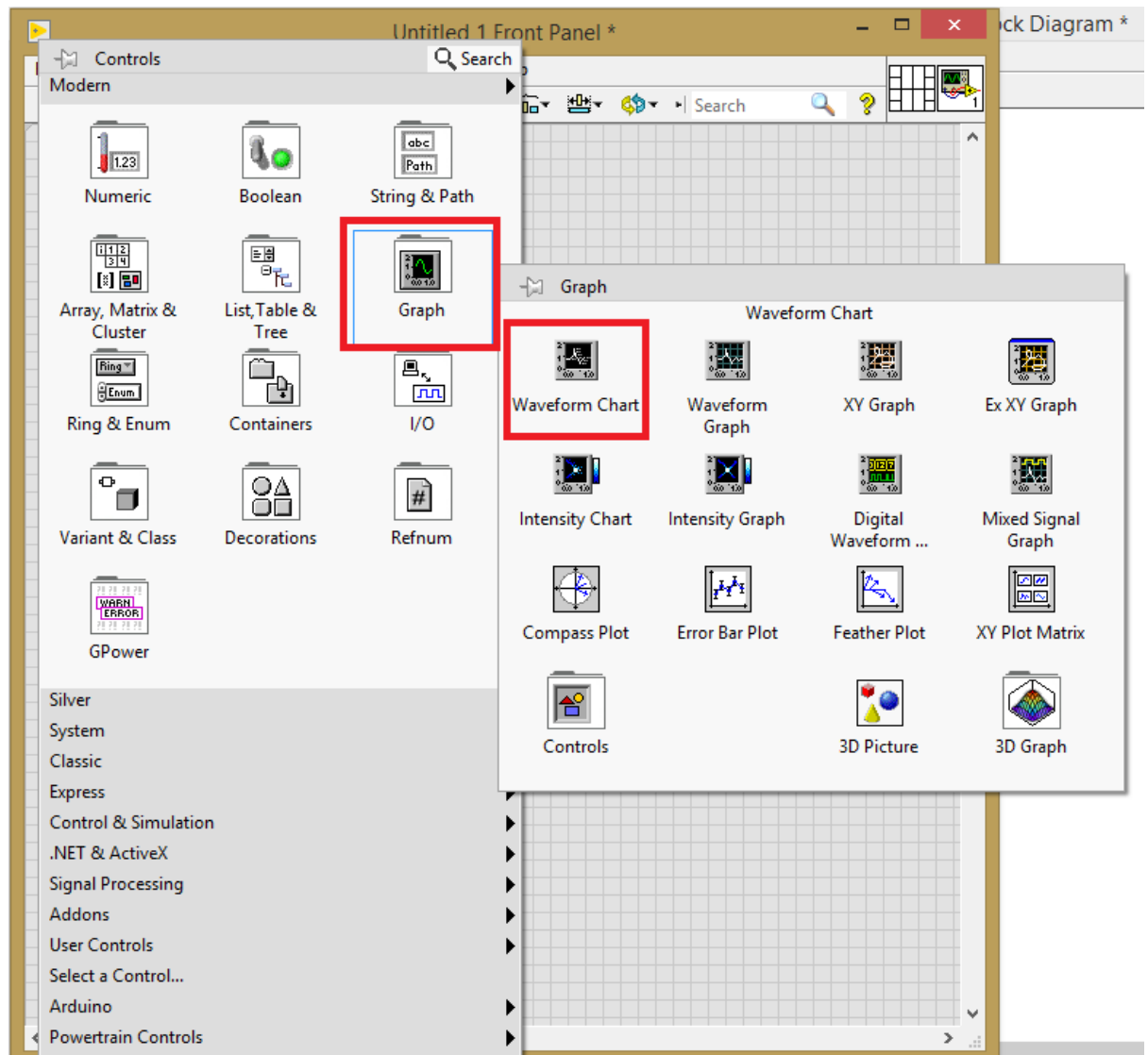
- A) Strip chart: Data moving left to right. After scale it shows moving at right.
- B) Scope chart: Moving left to right. After scale again starts from left side.
- C) Sweep chart: After one cycle moving left to right with updating the waveform shows with red symbol.



Waveform Graph

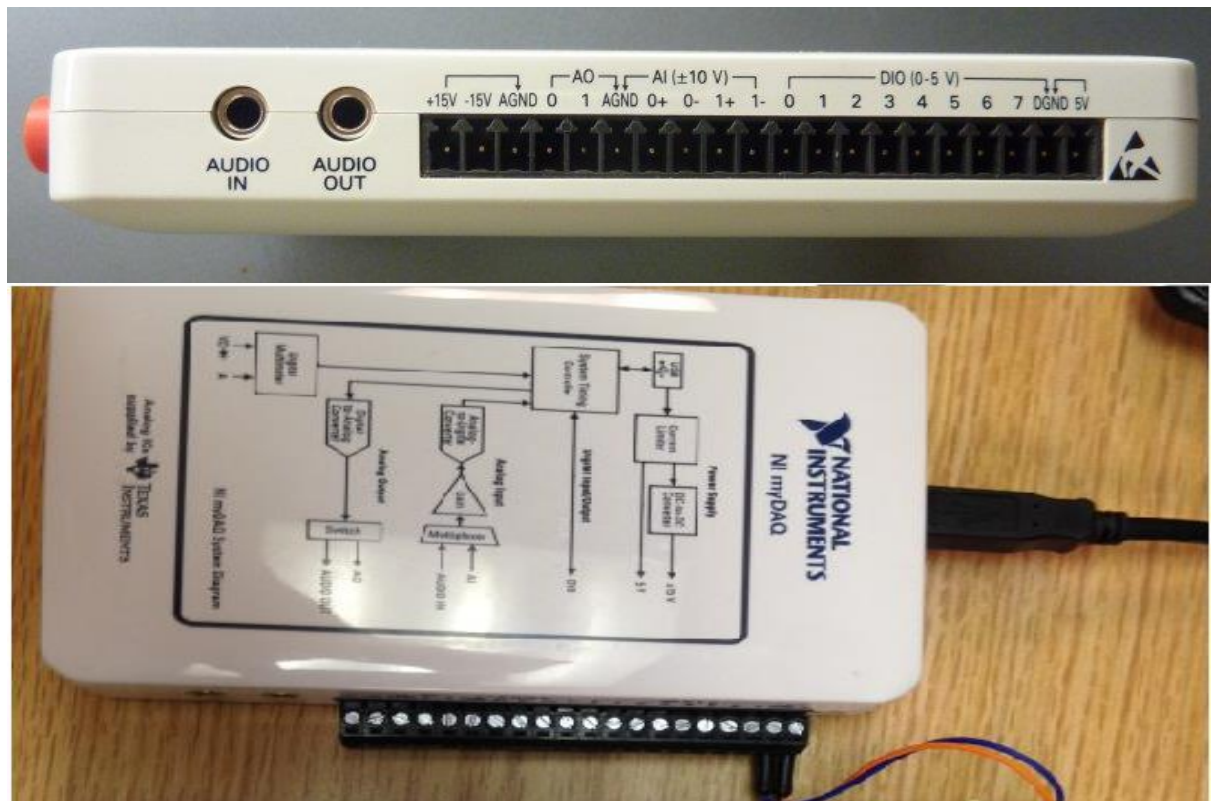
It cannot maintain the history. Remaining thing is same as charts. In order to represent the data at least it needs 1-D array.

It can be select from front panel R.C >> Graph >> Waveform Graph

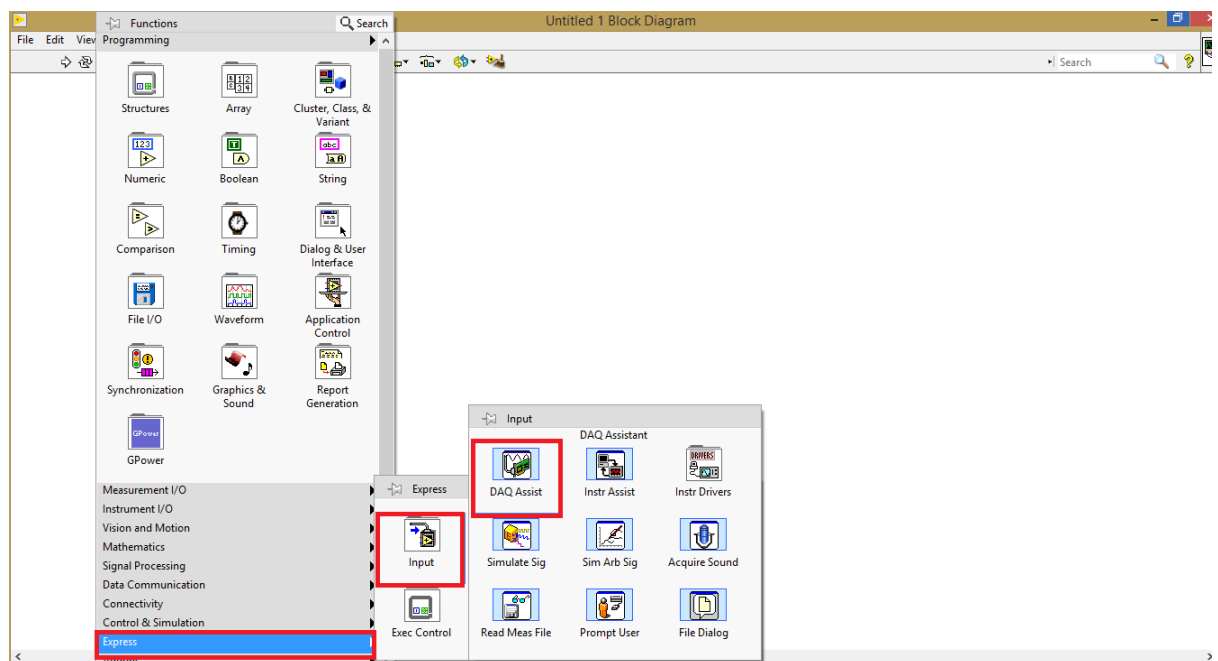


4 NI myDAQ

NI myDAQ(Data Acquisition) is a device which is used to applied the real signal to the computer. It has 2 analog input terminals, 2 analog outputs and 8 digital output pins are available. Along with this 5V & 15V supplies are also available. This voltage can be used for general electronics or to supply the power to small power devices.



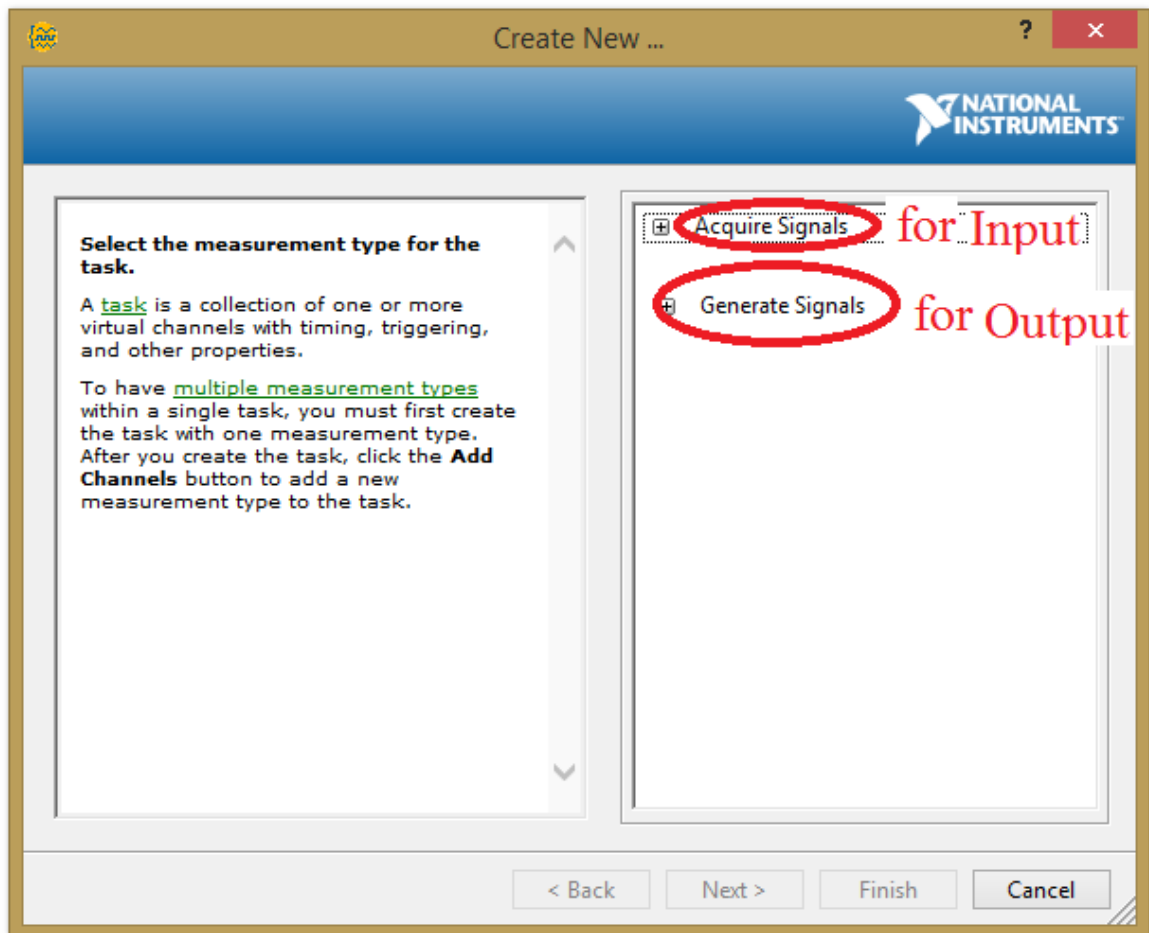
In order to get the DAQ interfacing to the computer in the LabVIEW software myDAQ drivers should be installed. Once it is installed it is available in block diagram R.C >> Express >> Input>> DAQ Assistant. This is the software tool which is used to interface any data acquisition card like myRIO, CRIO, USB-6008, USB-6009 etc.. to the LabVIEW.

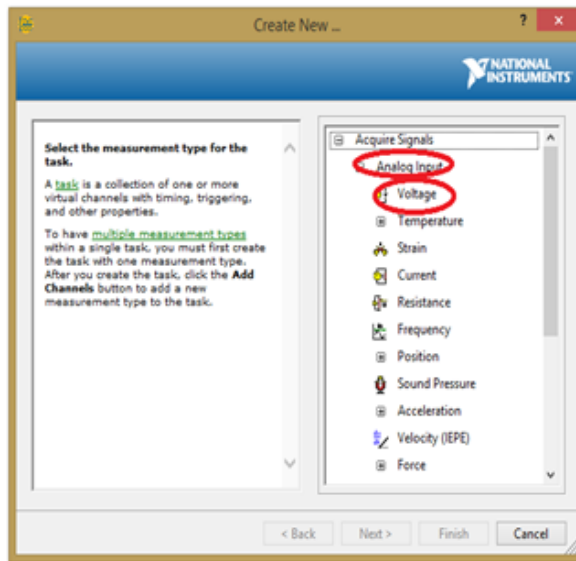


Once it is selected it ask about which type input/output signal

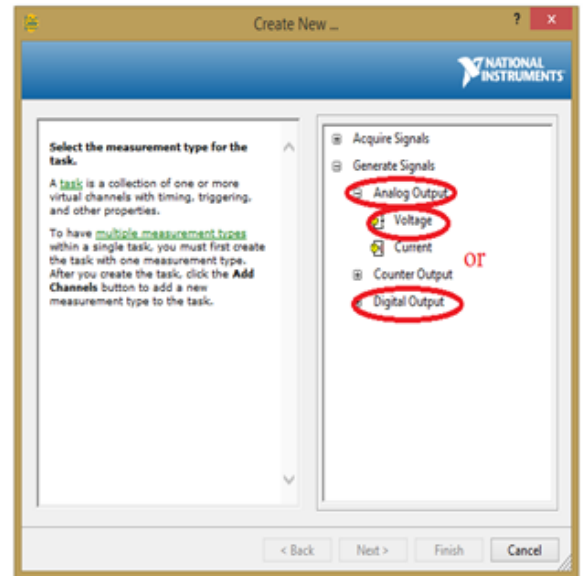
For Input –Acquire signals

For output-Generate Signals





For input giving
signal to LabVIEW



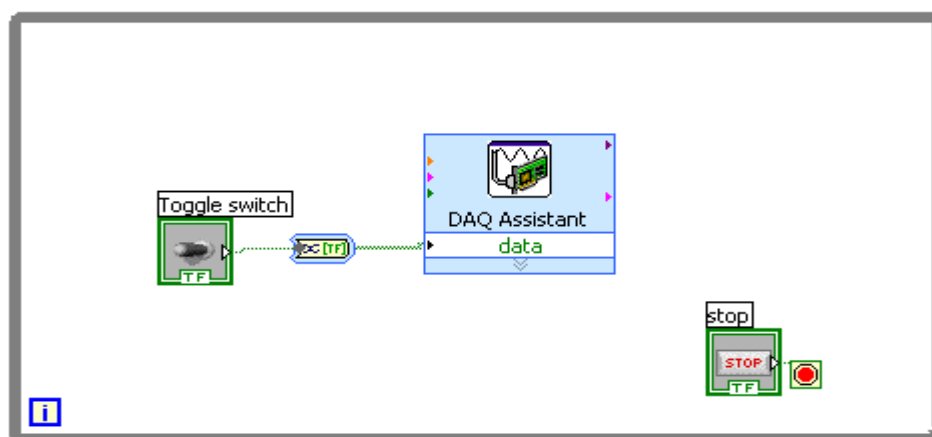
For out put
generating signal

If it is analog input signal then specify No. Of samples to read (this many samples show in LabVIEW at any time) and sampling rate(no.of samples per second).

Similarly if it is analog output then select continuous samples also specify the no. of samples and sampling rate. But if it is Digital output signal then select 1 sample on demand.

DAQ Assistant tool is used to give the analog input to the LabVIEW or to take out the Analog/Digital output from the LabVIEW.

Example: Turning ON/OFF “A Bulb”



For any queries, kindly contact

M. Sreenivasulu,
Asst. Prof.,
EEE Department,
VCE (A).

Email:

[m.srinivasulu@staff.vce.ac](mailto:m.srinivasulu@staff.vce.ac.in)
[.in](mailto:m.srinivasulu@staff.vce.ac.in)

Mobile: 7337063538

RAVI PONNALA

Asst. Prof.,

EEE Department,

VCE (A).

Email: [ravi.ponnala237@g](mailto:ravi.ponnala237@gmail.com)

[mail.com](mailto:ravi.ponnala237@gmail.com) or

[ravi.ponnala@staff.vce.ac.](mailto:ravi.ponnala@staff.vce.ac.in)

[in](mailto:ravi.ponnala@staff.vce.ac.in)

Mobile: 9989600881