MÜHENDİSLİK FAKÜLTESİ
FACULTY OF ENGINEERING
ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ BÖLÜMÜ
DEPARTMENT OF ELECTRICAL AND
ELECTRONICS ENGINEERING

ORTA DOĞU TEKNİK ÜNİVERSİTESİ
MIDDLE EAST TECHNICAL UNIVERSITY

DUMLUPINAR BULVARI 06800
ÇANKAYA ANKARA/TURKEY
T: +90 312 210 23 02
F: +90 312 210 23 04
ee@metu.edu.tr
www.eee.metu.edu.tr

INTRODUCTION TO AGILENT VEE

I. Introduction

The Agilent **V**isual **E**ngineering **E**nvironment (VEE) is a graphical data flow programming language from Agilent Technologies (Keysight) for automated test, measurement, data analysis and reporting. This tutorial aims at giving a basic understanding of VEE Pro and presents various examples.

In this tutorial, we will not use real instruments, instead we focus on developing some intuition about how VEE Pro works.

Introduction to Agilent VEE by Özlem Tuğfe Demir and Barış Bayram © 2015
All Rights Reserved. (e-mail: ozlemtugfedemir@gmail.com)

II. Understanding Pins of an Object in VEE

In this part, we will examine the pins of the objects in VEE. As an example, consider the formula object given in Fig. 1.

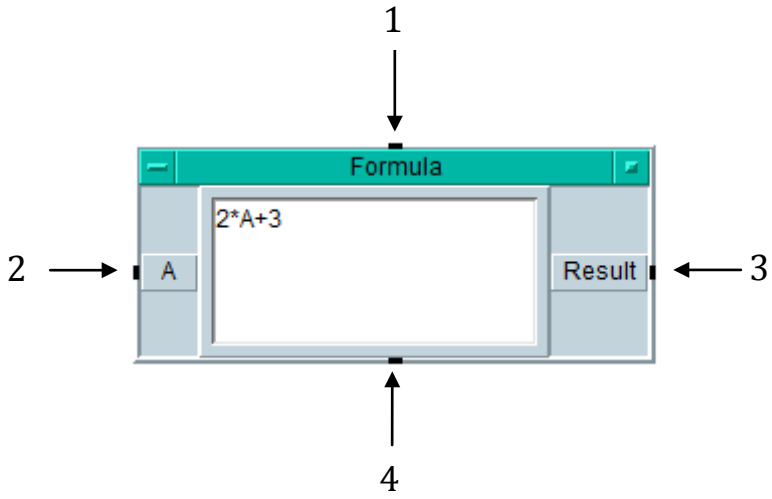

Fig. 1. The pins of an object

1) The small black box corresponding to 1 is the **sequence input pin**. If it is connected, the object will not operate until it receives a message through this pin. Note that the sequence input pin does not have to be connected.
2) The small black box corresponding to 2 is the **data input pin**. In Fig. 1, there is only one data input. We can add other inputs if needed. All data inputs should have data before the object operates. Hence, all data input pins should be connected.
3) The small black box corresponding to 3 is the **data output pin.** There may me more than one data output. An object performs its task after receiving all data inputs and sends the results to the data output pins.
4) The small black box corresponding to 4 is the **sequence output pin.** The sequence output pin is not fired until all objects connected to the data output pins have received the output data.

III. Execution Order of the Objects in VEE

As a VEE program runs, the objects execute in the following order.

- **Start** objects operate first. In a VEE program, we connect the Start objects to the sequence input pins of other objects to make them execute first. In Fig. 2, you see a Formula object with a Start object connected to its sequence input pin. When the program runs, firstly the Formula object performs its task.

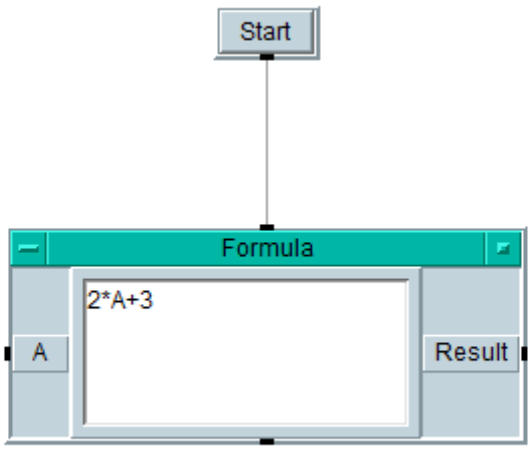

Fig. 2. A Formula object connected with a Start object

- Objects without any data input executes after the objects connected with Start object.

- Objects with data inputs execute after their sequence input pins are fired (if connected) and they receive the input data through their data input pins.

IV. Lessons for VEE Using Virtual Instruments

Lesson 1: Generating and Displaying a Waveform

In this lesson, you will learn how to generate and display a waveform using virtual instruments.

In Fig. 3, you see the **Menu Bar** placed in the upper part of the work area.

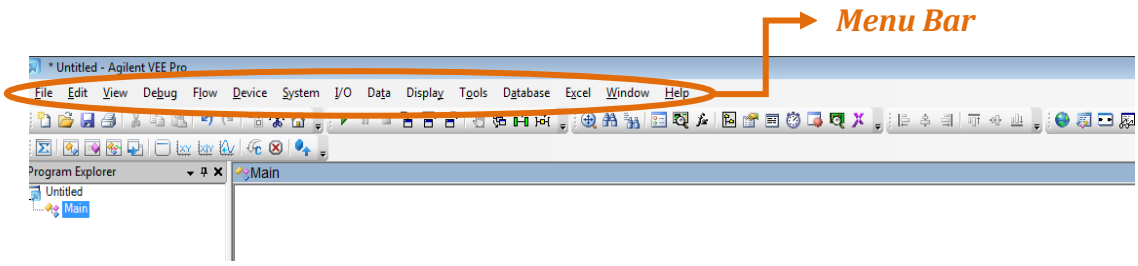

Fig. 3. Menu Bar

- In order to create a virtual function generator, select **Menu Bar → Device → Virtual Source → Function Generator**. In Fig. 4, you can see how a function generator is created and a **Function Generator** object. You can adjust the type of function (cosine, triangular, etc.) and other parameters of it from the dropdown menus. The rectangle labeled as **Func** is the data output pin of the Function Generator object and it is the generated waveform.

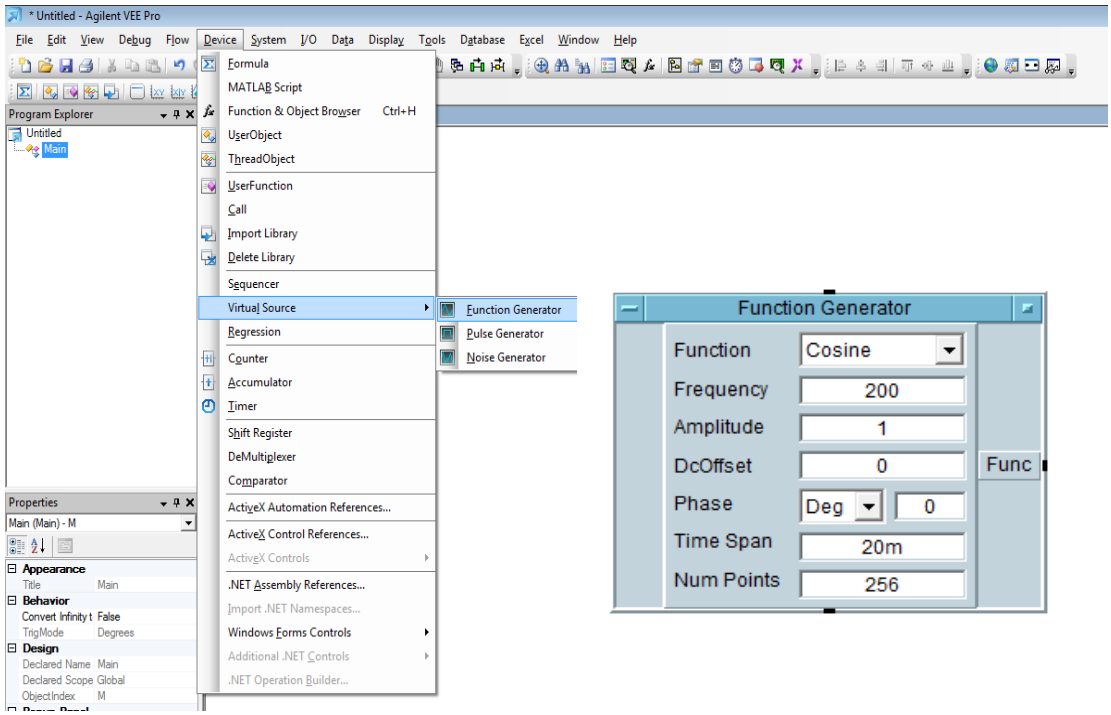

Fig. 4. Function Generator

- Select **Menu Bar → Display → Waveform (Time)** for plotting the generated waveform in time domain.

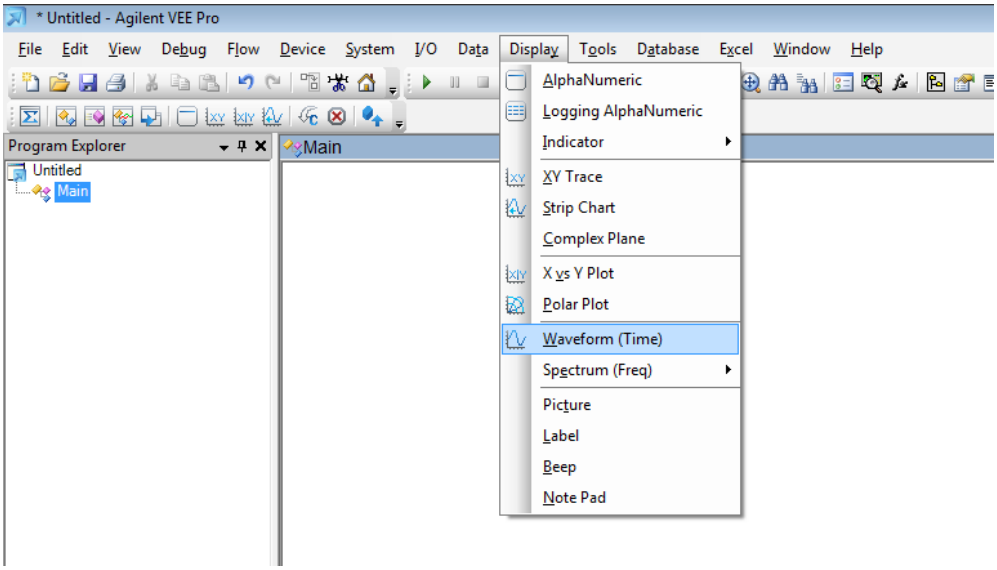

Fig. 5. Adding a Waveform (Time) object

- Adjust the settings of the function generator as shown in Fig. 6 and connect the data output pin of the function generator (**Func**) to the data input pin of the Waveform (Time) labeled as **Trace1**. In order to run the program, press **Run** button placed below the Menu Bar. After the program runs, the waveform in Fig. 6 is displayed.

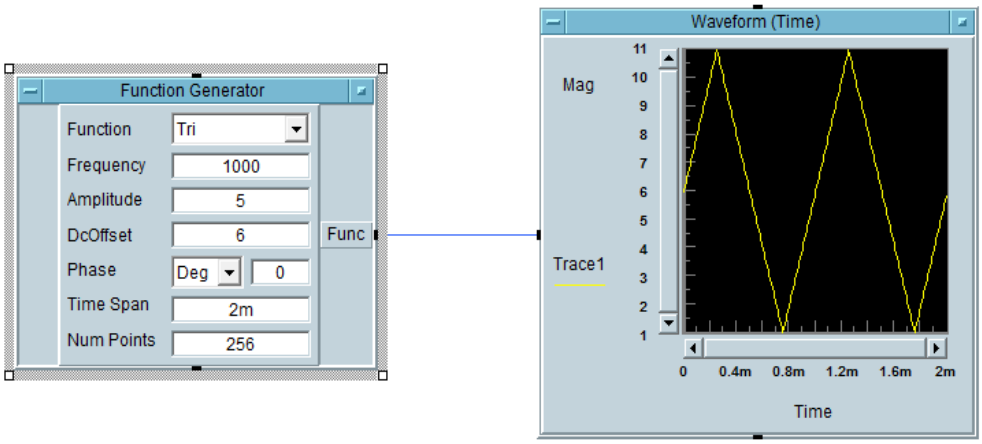

Fig. 6. Displaying a triangular waveform

- Now, let's change the color and the line type of the displayed waveform to magenta and dash respectively by pressing **Trace1** on the Waveform object. Moreover, let's press **Mag** and change the minimum and maximum values to 0 and 15. The following figure in Fig. 7 is seen after these modifications.

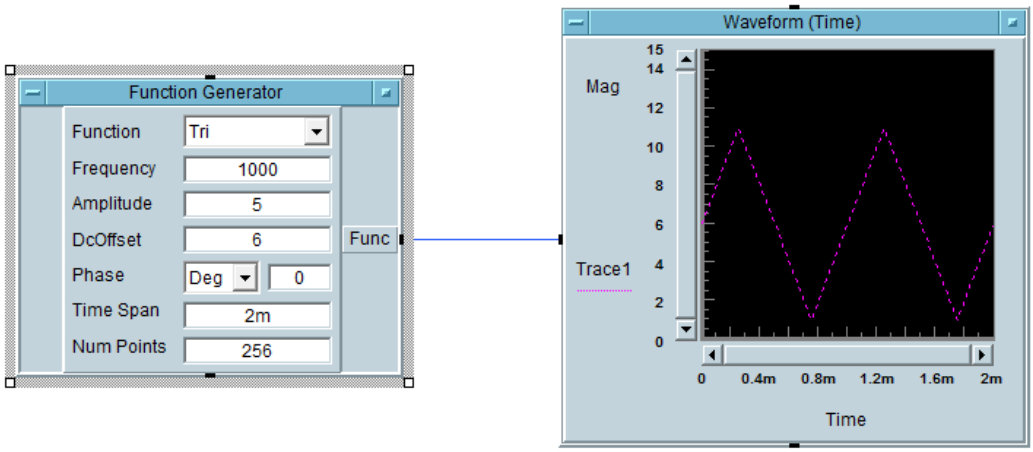

Fig. 7. Displaying a triangular waveform with magenta and dash Line

Lesson 2: Generating and Displaying a Noisy Waveform

In this lesson, we will generate virtual noise and display the sum of the original waveform and noise.

- Create a Function Generator and a Waveform (Time) object using the procedure in Lesson 1.

- Select **Menu Bar → Device → Virtual Source → Noise Generator** and place the object in the work area.

- At this point, we need an adder to sum the waveforms of the Function Generator and Noise Generator. There is not one way to do this. In this lesson, we will use **Function & Object Browser**. Using Function & Object Browser, it is possible to execute various arithmetic, logical, etc. operations. Select **Menu Bar → Device → Function & Object Browser**. You will see an opened box at the right side of the Work Area as in Fig. 8.

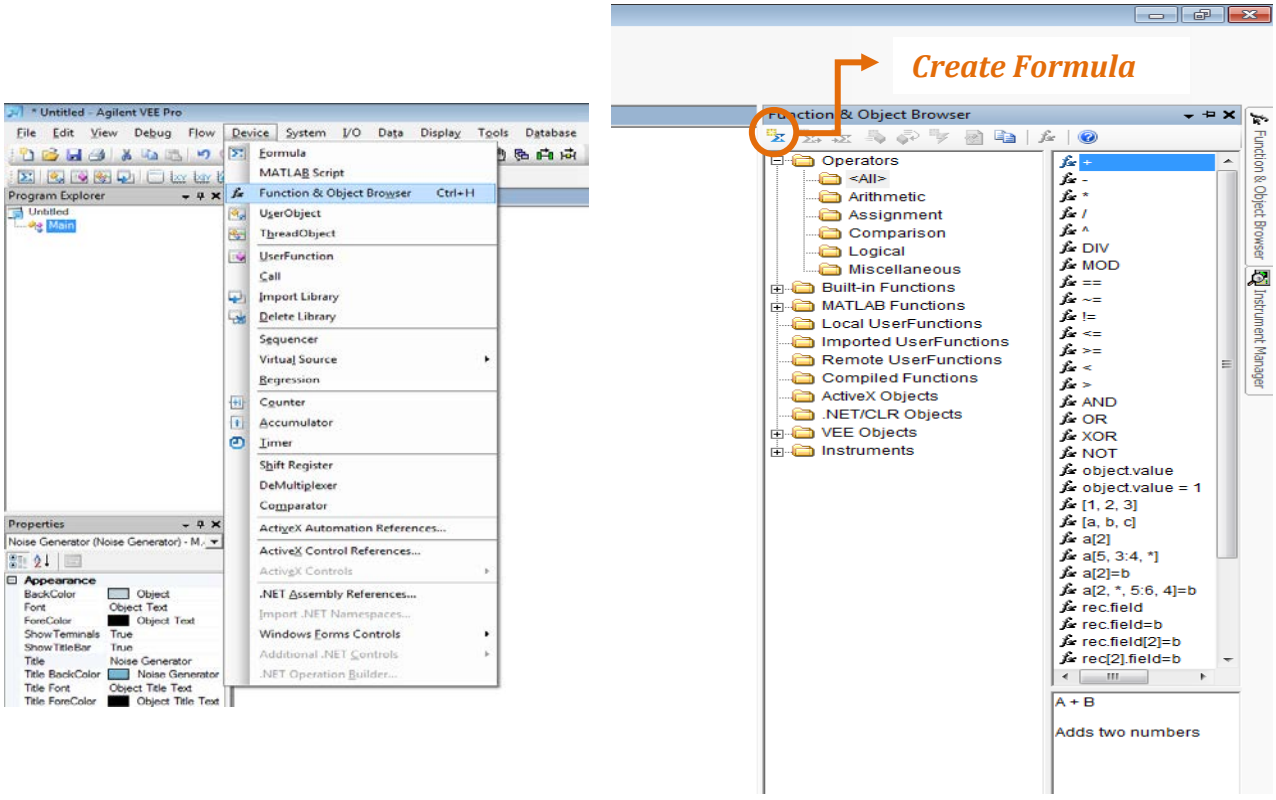

Fig. 8. Function & Object Browser

- On the opened box, expand **Operators** and select **<All>** and **+** function at the right. You can see the formula of the selected function at the right bottom corner of the box. Press the **Create Formula** button as in Fig. 8 and place the object in the work area.

- Connect the data output pins of Function Generator and Noise Generator, (**Func** and **noise WF**) to the data input pins of the **A+B**. Then connect the data output pin of the A+B (**Result**) to the data input pin of the Waveform (Time) as in Fig. 9 and run the program. You will display a cosine waveform which is distorted by some noise.

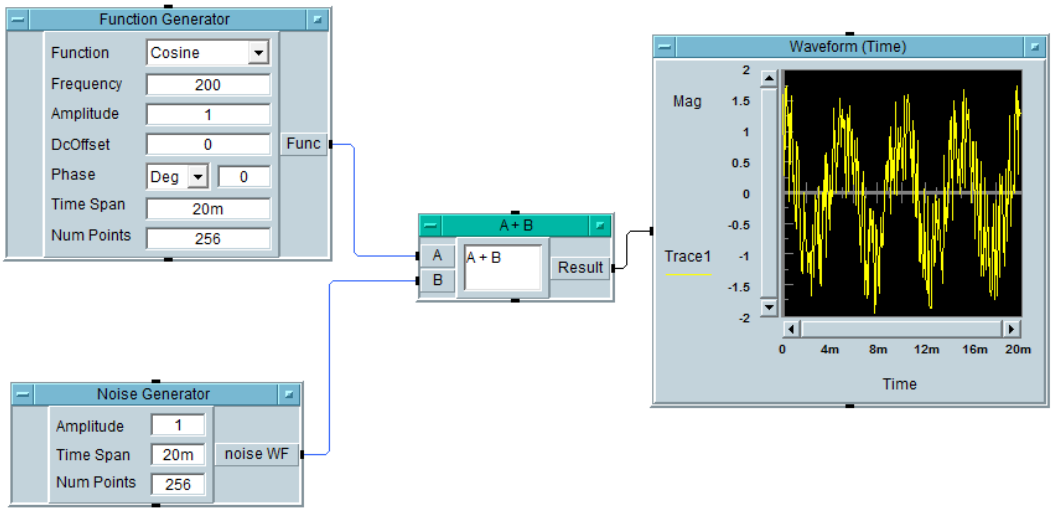

Fig. 9. Displaying a noisy waveform

Lesson 3: Adding a Data Input and Real64 Slider

In this lesson, we will learn how to adjust an input parameter using Real64 Slider object.

- Set up the configuration in Lesson 2.

- Our aim is to make the amplitude of the noise variable and adjust it. For this purpose, **right click** on **Noise Generator** object and select **Add Terminal → Data Input** and then select **Amplitude** from the opened menu and click **OK**. Now, a data input for the amplitude is created.

- In order to adjust the amplitude of the noise, we can use a slider. Select **Data → Continuous → Real64 Slider** and place it in the work area. Real64 Slider allows us to select a real input value with 3 digits after the decimal point. If you would like to select integer input values, you can choose **Int32 Slider** instead of Real64 Slider.

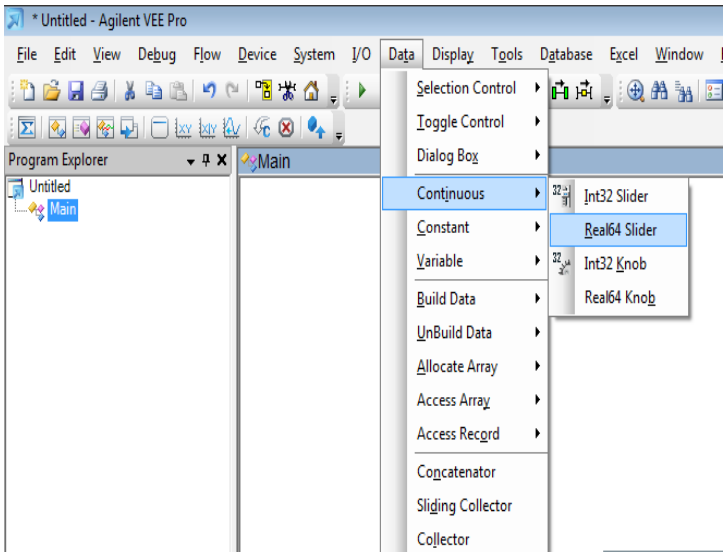

Fig. 10. Adding a Real64 Slider

- Connect the data output pin of Real64 Slider to the data input pin of Noise Generator (**Amplitude**) as in Fig. 11.

- Adjust the noise amplitude to 0.21 and run the program. You will observe a noisy cosine waveform with less noise than that in Lesson 2.

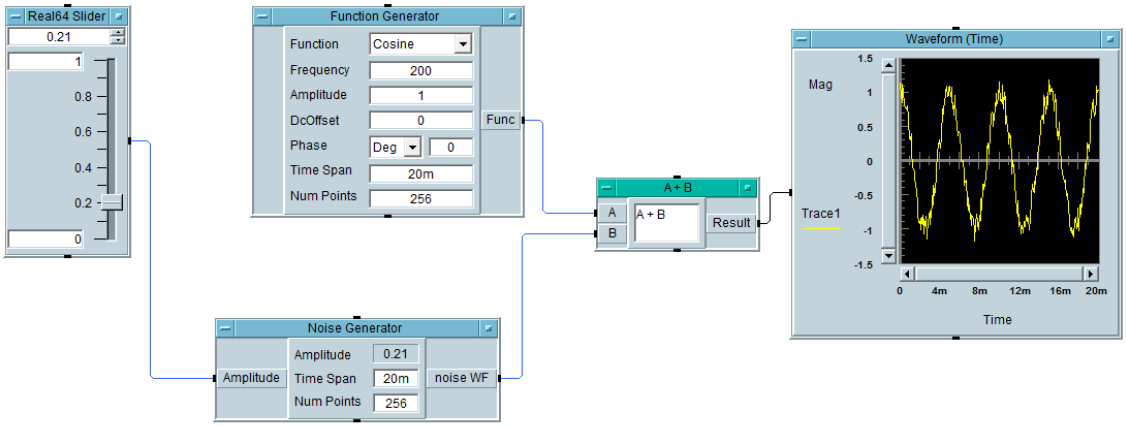

Fig. 11. Adjusting the noise amplitude with Real64 Slider

Lesson 4: Creating a Dialog Box for User Input

In this lesson, we will adjust the frequency of Function Generator with a **Dialog Box.**

- Set up the configuration in Lesson 2.

- Our aim is to make the frequency of the Function Generator variable and adjust it with **Dialog Box**. For this purpose, **right click** on **Function Generator** object and select **Add Terminal → Data Input** and then select **Frequency** from the opened menu and click **OK**. Now, a data input for the frequency is created.

- Select **Menu Bar → Data → Dialog Box → Int32 Input**.

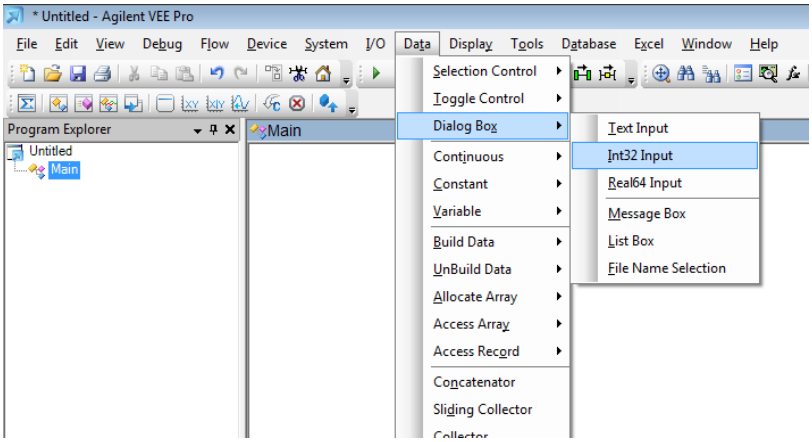

Fig. 12. Adding an Int32 Input Dialog Box

- You see an **Int32 Input Dialog Box** in Fig. 13.

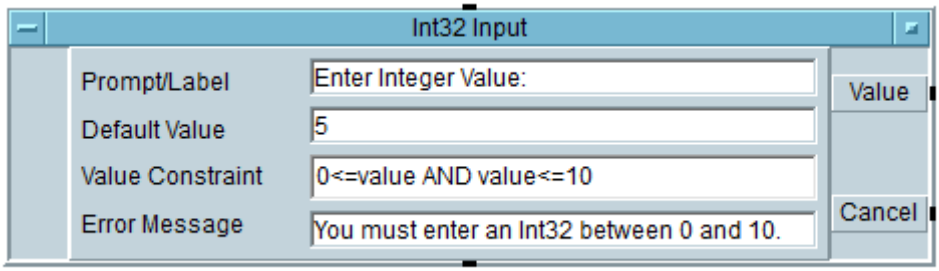

Fig. 13. Int32 Input Dialog Box

- Fill the dialog Box as in Fig. 14. For this example, only the integer valued frequencies between 1 and 1000 Hz is acceptable. Otherwise, the error message written is **Error Message** field is displayed.

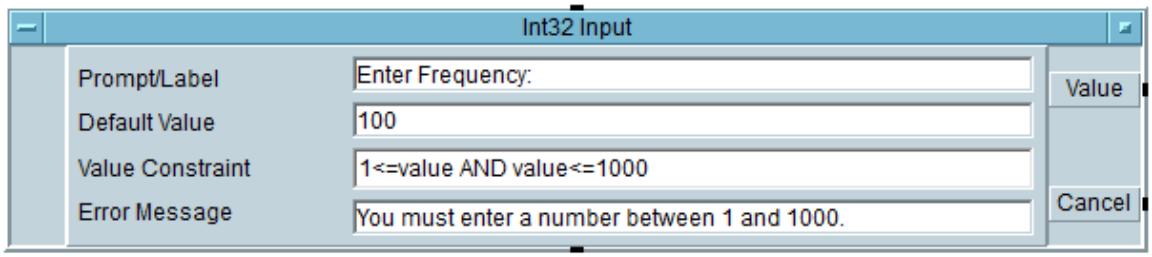

Fig. 14. Int32 Input Dialog Box for the frequency between 1 and 1000

- Connect the data output pin labeled as **Value** to the **Frequency** pin of Function Generator. A window will appear as you run the program as in Fig. 15. Enter the desired frequency into the opened window and click **OK**.

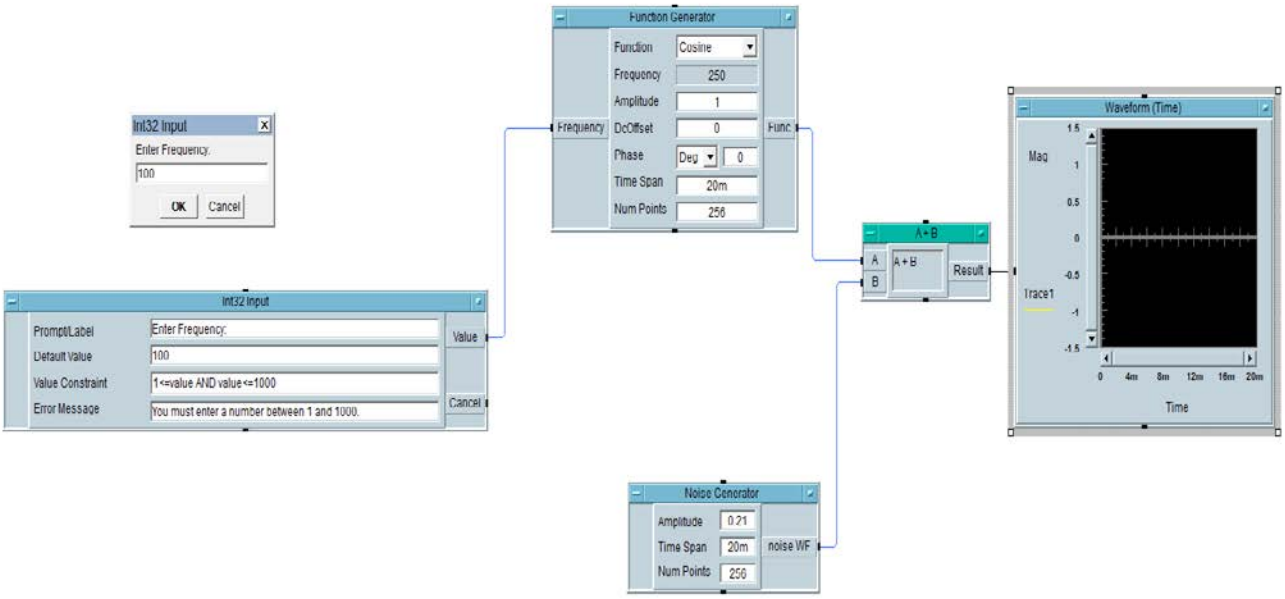

Fig. 15. The appearance of the work area after the program executes

- Enter 250 into the text box below **Enter Frequency:** and click **OK**. You will observe a waveform similar to the one in Fig. 16.

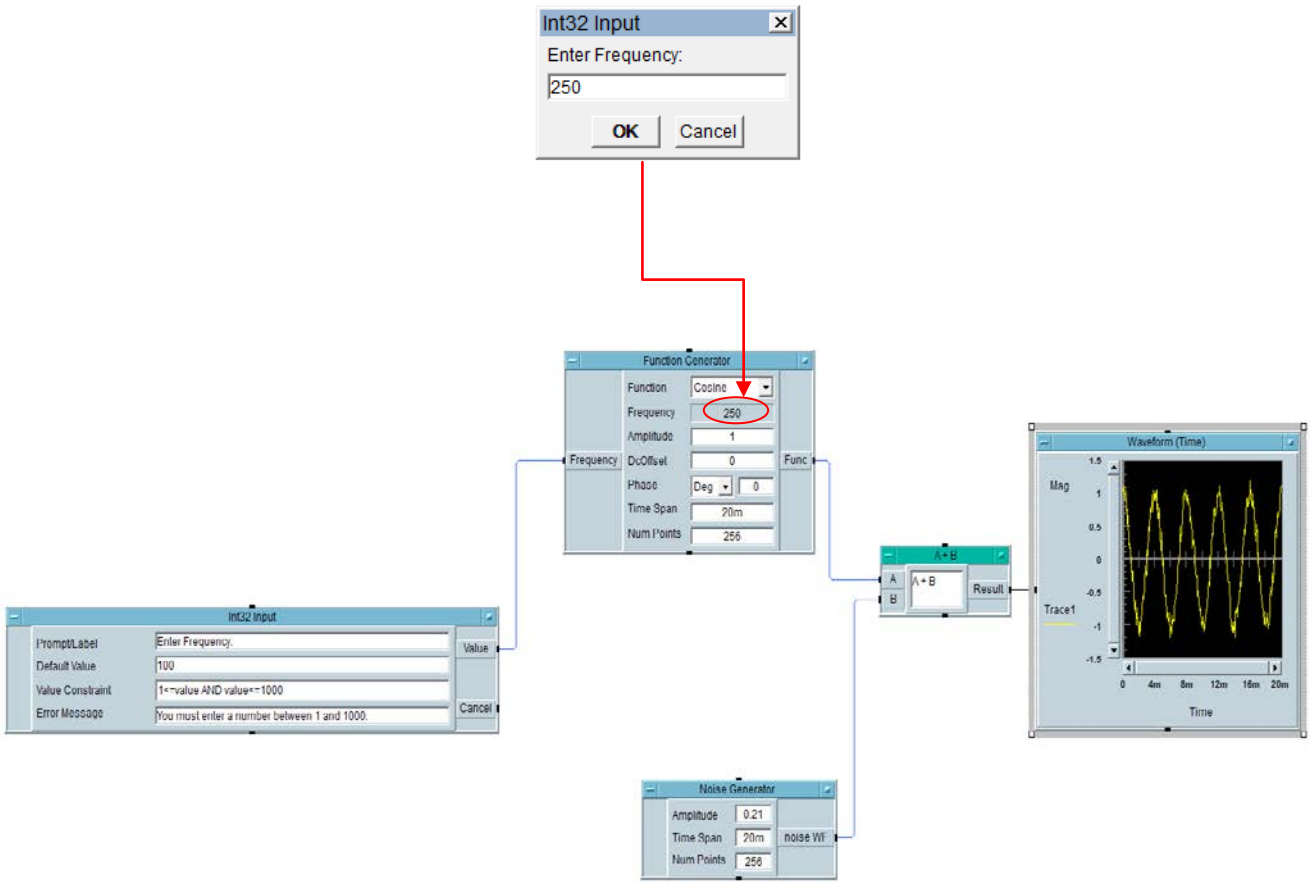

Fig. 16. Adjusting the frequency of the Function Generator with an Int32 Input Dialog Box

- In case of entering a non-integer frequency value, the program rounds the frequency to the nearest integer value.

- In case of entering a number out of the predefined range (1-1000 in our example), the error message is displayed as in Fig. 17.

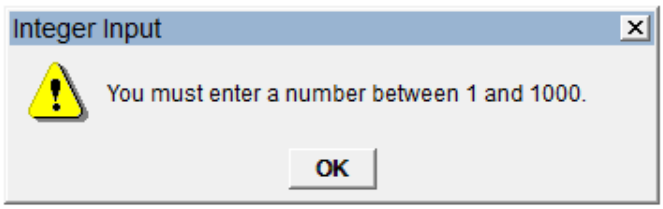

Fig. 17. The error message

Lesson 5: Mathematically Processing Data

In this lesson, we will learn how to add an **Formula** object and a **Real64 Constant**.

- Add a **Function Generator** and adjust it to generate a 100 Hz sinusoid.
- Add a **Waveform (Time)**.
- Select **Menu Bar → Constant → Real64** as shown in Fig. 18.
- Set the value of **Real64** object to **0.5**.

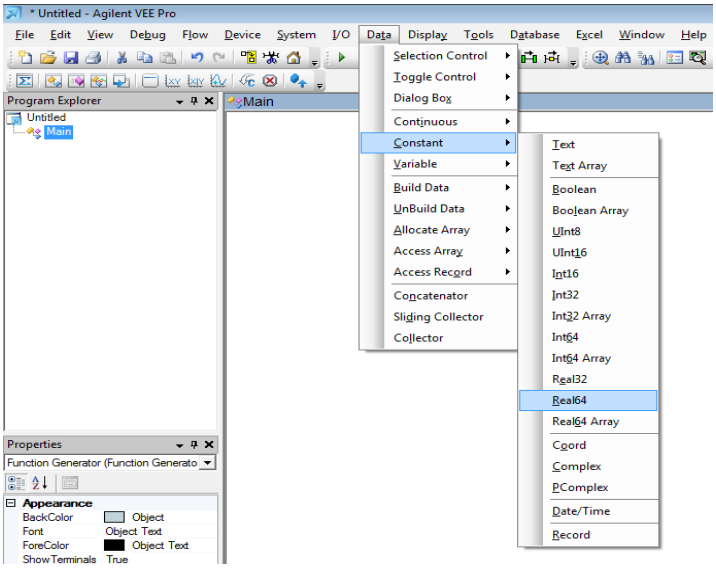

Fig. 18. Adding a Real64 Constant

- Our aim is to plot absolute value of the sine wave with a 0.5 DC offset. For this purpose, we will use **Formula** object. Select **Menu Bar → Device → Formula** as shown in Fig. 19.

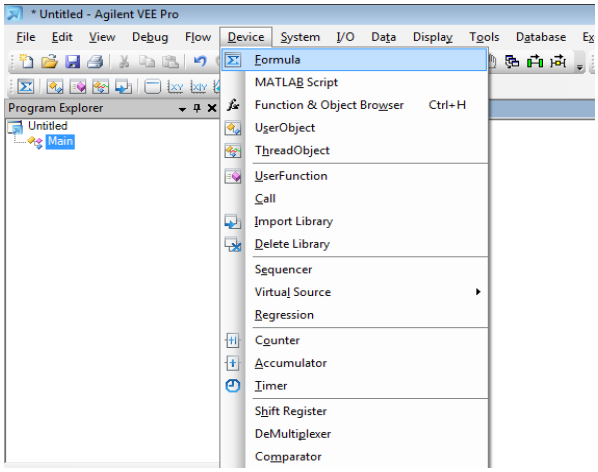

Fig. 19. Adding a Formula Object

- **Right click** on **Formula** object and select **Add Terminal → Data Input**. Now, a second data input labeled as **B** is created. Note that **A** and **B** stand for the output of the Function Generator and Real64 Constant respectively. Hence, connect the data output pins of the Function Generator and Real64 to A and B pins of the Formula object. Enter **abs(A)+B** to the text box in the Formula.

- Connect the data out pin of the Formula object (**Result**) to the data input pin of **Waveform (Time)** and run the program. You will observe a waveform as in Fig. 20.

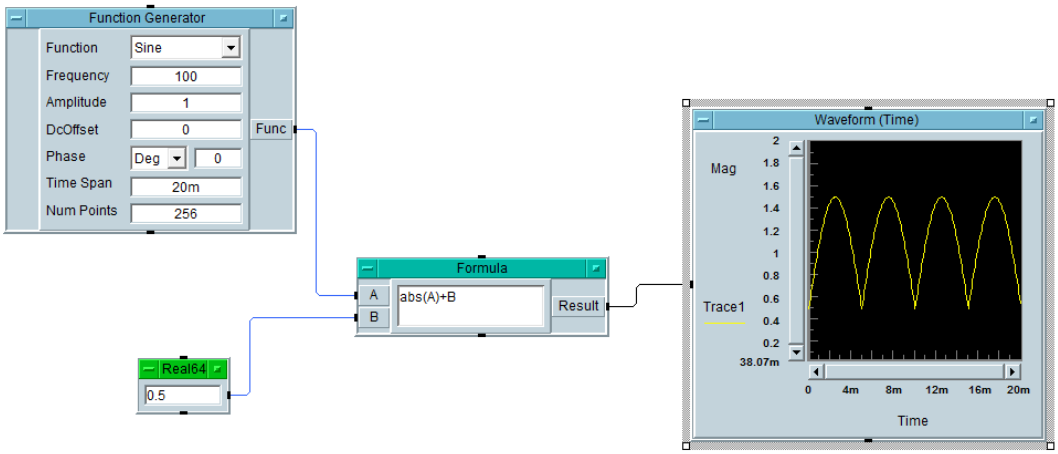

Fig. 20. Displaying the result of the Formula object

Lesson 6: Creating an Array for Test Results

In this lesson, we will learn how to create and array and displaying its content.

- Firstly, we will add a counter. Select **Menu Bar → Flow → Repeat → For Count** and set its value to 4. This means that it will count from 0 to 3.

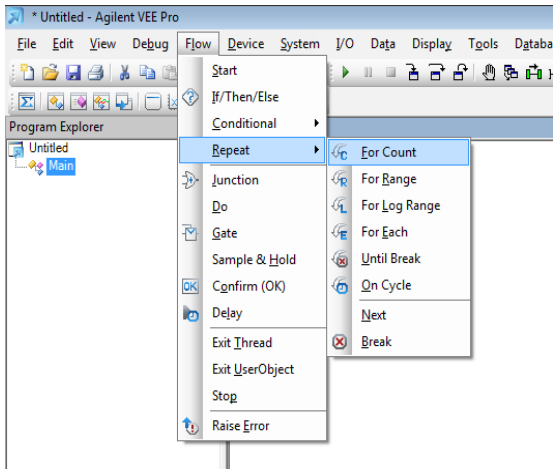

Fig. 21. Adding a For Count object

- In order to display the output value of For Count, select **Menu Bar → Display → AlphaNumeric**.

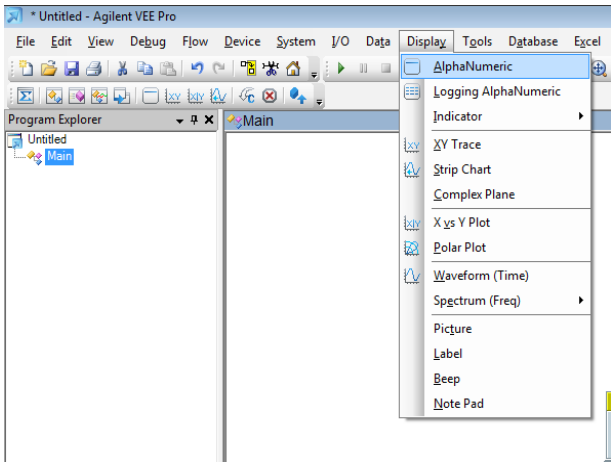

Fig. 22. Adding an AlphaNumeric object

- As you run the program, AlphaNumeric will display the value of For Count. Since it counts very rapidly, you will see only 3 as in Fig. 23.

Fig. 23. Displaying the current value of For Count with AlphaNumeric

- In order to see the all values of For Count with a certain period, we can use **Delay** object. To add a Delay object, select **Menu Bar → Flow → Delay** and set its value to 1.

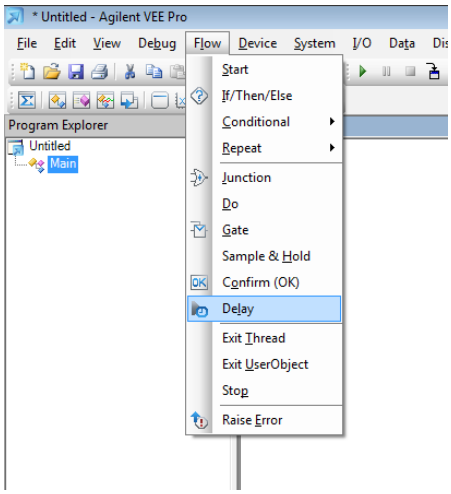

Fig. 24. Adding a Delay object

- Connect the **data output pin** of **For Count** to the **sequence input pin** of **Delay** and **sequence output pin** of **Delay** to the **sequence input pin** of **AlphaNumeric**. In this way, AlphaNumeric can only display the value of For Count after 1 second delay. When you run the program, AlphaNumeric will display 0, 1, 2 and 3 with an approximately 1 second period.

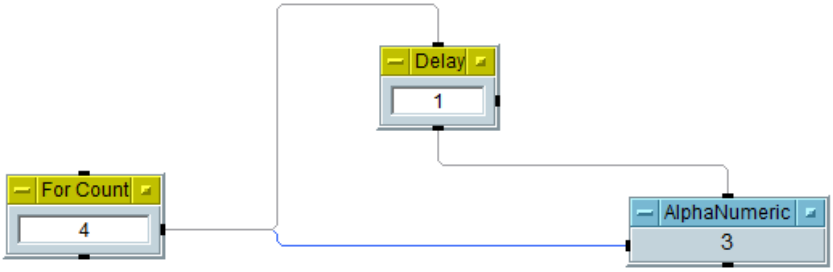

Fig. 25. Displaying the current value of For Count with 1 second delay

- Alphanumeric object only displays the current value of the objects. In order to display the values in sequence, we can use **Logging AlphaNumeric** object. Now, delete the **AlphaNumeric** object and add a **Logging AlphaNumeric** object by selecting **Menu Bar → Display → Logging AlphaNumeric**.

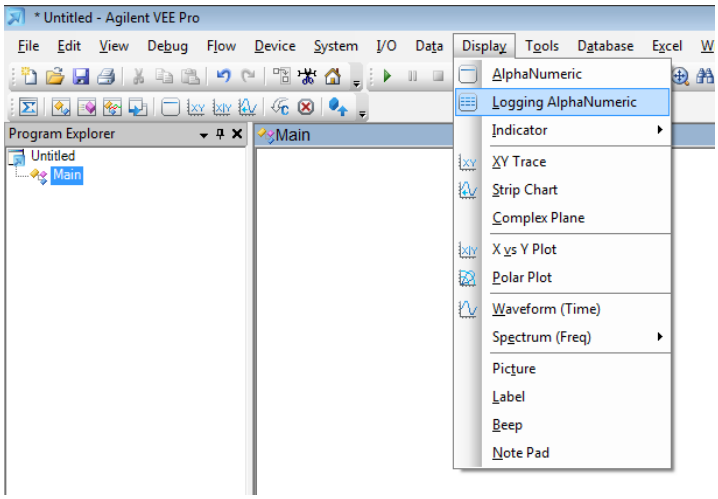

Fig. 26. Adding a Logging AlphaNumeric object

- As you run the program, Logging AlphaNumeric will display 0, 1, 2 and 3 in sequence with 1 second period as in Fig. 27.

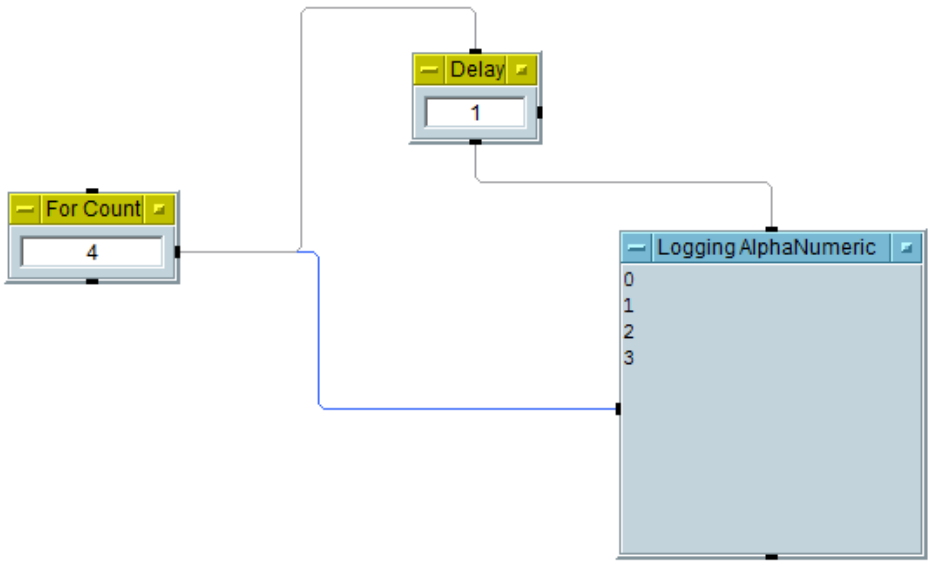

Fig. 27. Displaying all values of For Count in sequence with 1 second delay

- Now, clear the work area except For Count object and select **Menu Bar → Data → Collector**. This object will collect the output values of For Count in an array.

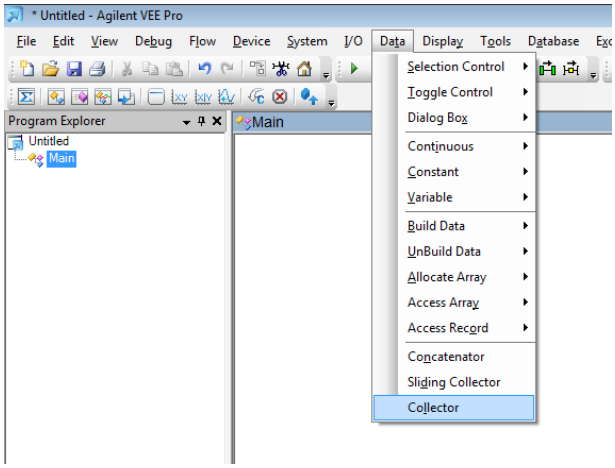

Fig. 28. Adding a Collector

- Connect the **data output pin** of **For Count** to **data input pin** of **Collector** labeled as **Data**.
- Connect the **sequence output pin** of **For Count** to the **control input XEQ**. **XEQ** is activated after For Count finishes counting. Collector will not send its data to its output pin unless XEQ is activated. Hence, in this way, the output of the Collector will contain all values of For Count (0, 1, 2, 3).
- Add two **Formula** objects and connect the **data output pin** of **Collector,** labeled as **Array.** Enter **A[0]** into one of the formula boxes and **A[1:3]** into the other. Here, **A** is the data array [0 1 2 3]. Hence, A[0]=0 and A[1:3]=[1 2 3].
- Add two **AlphaNumeric** objects and connect the **data output pins** of the **Formula** objects to the **data input pins** of them.
- When you run the program, you will observe the results as in Fig. 29.

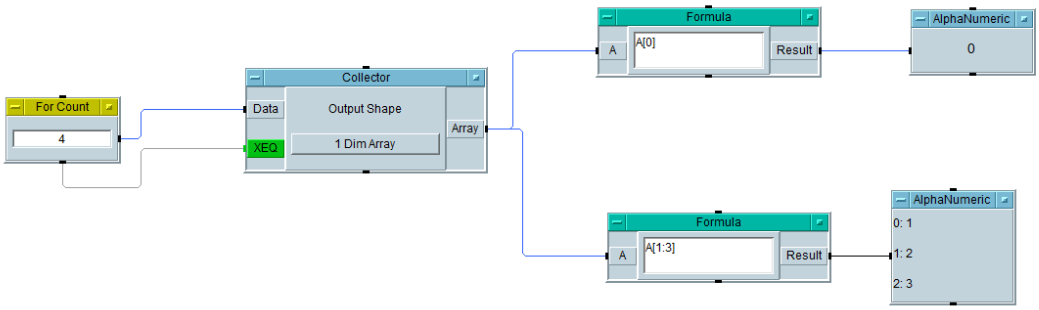

Fig. 29. Displaying the content of an array

- Now, clear the work area and add a **constant Text** object by selecting **Menu Bar → Data → Constant → Text** as shown in Fig. 30. Enter **deneme** into the text box.

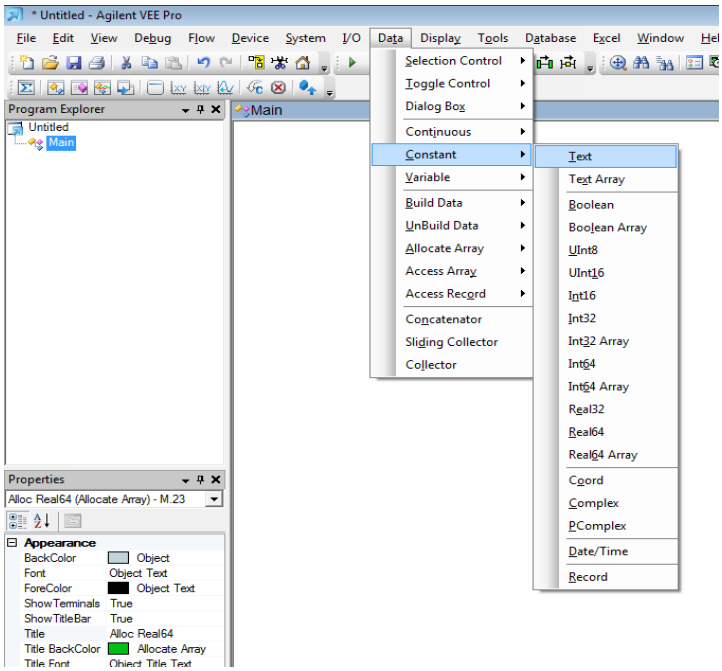

Fig. 30. Adding a Constant Text object

- Select **Menu Bar → Data → Allocate Array → Real64** and place it in the work area. This object creates several types of arrays with a predefined dimension.

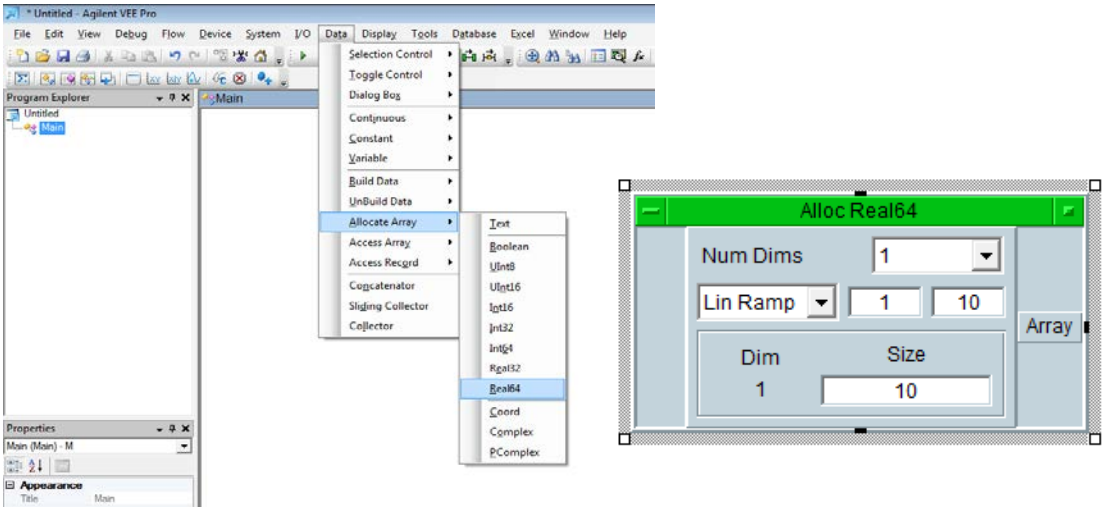

Fig. 31. Adding an Real64 Allocate Array object

- Select **Lin Ramp** array from **8** to **14** with size **4** in **Alloc Real64** object as shown in Fig. 32. This object will create the linear ramp array [8 10 12 14].

Fig. 32. Real64 Allocate Array object

- Now, we will concatenate the data in Text and Alloc Real64 objects. Select **Menu Bar → Data → Concatenator** as shown in Fig. 33.

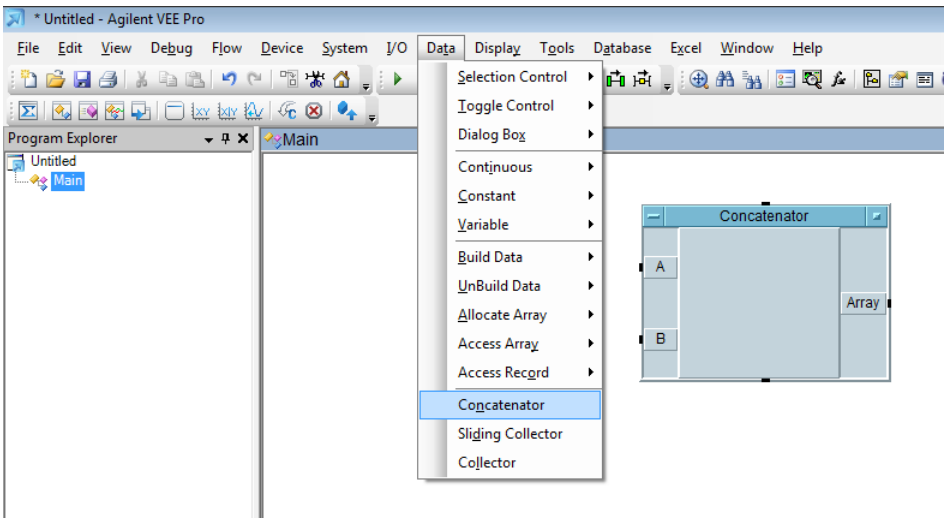

Fig. 33. Adding a Concatenator object

- Add an AlphaNumeric object into the work area and make the connections of the objects as shown in Fig. 34. When you run the program, AlphaNumeric object will display the concatenated data as shown in Fig. 34.

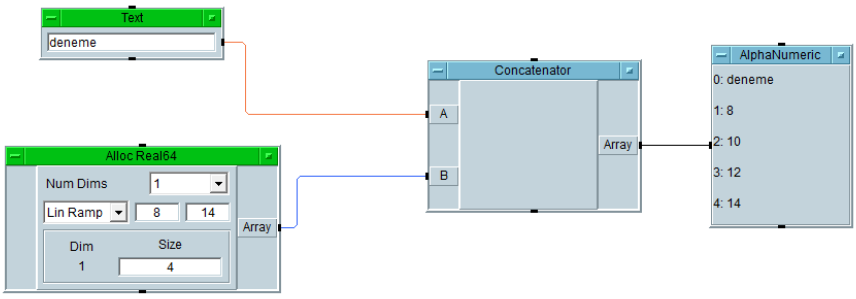

Fig. 34. Concatenating data

Lesson 7: X vs Y Plot

- Add three Function Generator objects in the work area. For each object, set 1 kHz cosine waveform with 1 milliseconds span time and 256 sample points. Adjust their amplitudes as 1, 2, and 3, respectively as shown in Fig. 35.

- Our aim is to plot these three waveforms on the same graph. We will do this using **X vs Y Plot object**. Select **Menu Bar → Display → X vs Y Plot**.

- Right click on X vs Y Plot object and select **Add Terminal → Data Input**. Now, another trace called **YData2** is created for the y axis. Repeat this once more.

- For the x axis, a same length array to the X vs Y Plot object should be given. For this purpose, add an Alloc Real64 object in the work area and choose Lin Ramp from 0 to 1m with size being equal to 256. Make the connections as shown in Fig. 35 and run the program. You will observe three cosine waveforms with different amplitudes on the same graph.

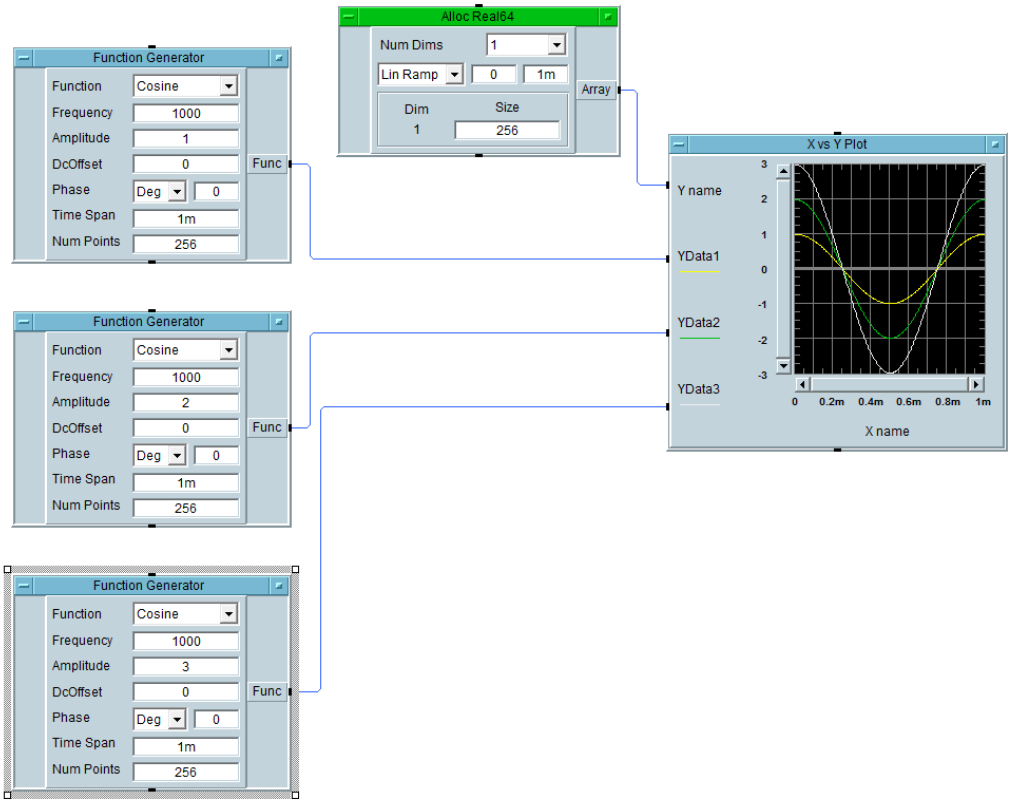

Fig. 35. Displaying using X vs Y Plot object

Lesson 8: For Range Object

- Select **Menu Bar → Flow → Repeat → For Range**. Enter From=0, Thru=9 and Step=3 as shown in Fig. 36. This object will give 0, 3, 6, and 9 as output in sequence.

- Add a Formula object and enter **2*A** to the formula text box.

- Add a X vs Y Plot object. Make the connections as in Fig. 36 and run the program. You will observe 2A versus A for A=0, 3, 6 and 9. Since line type is solid, you see interpolated plot which seems to be continuous.

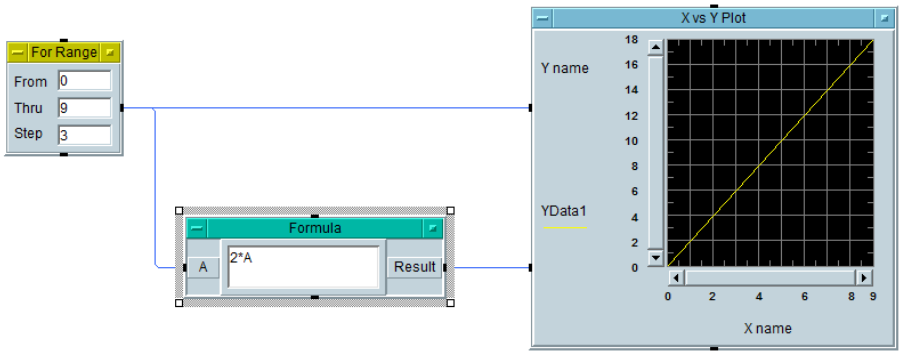

Fig. 36. For Range object

- Now, clear the work area and add the objects in in Fig. 37. Make all necessary connections. In this configuration, the output of For Range object, which is the frequency of the Function Generator, is swept from 100 to 1000 with step size 100. Successive waveforms with increasing frequency are genereated. We should add a control input to X vs Y Plot object in order to prevent it from deleting the previous plot. For this purpose, right click on X vs Y Plot object and select **Add Terminal → Control Input → Next Curve**. Connect the Func pin of the Function Generator to the Next Curve pin of X vs Y Plot object. This will ensure that when a new waveform is sent to X vs Y Plot from Function Generator, the previos plot is not deleted from the graph. Note that, since Next Curve is a control input, the line connected to it is a dashed line.

- When you run the program, you will observe a series of cosine waveforms with increasing frequency as shown in Fig. 37.

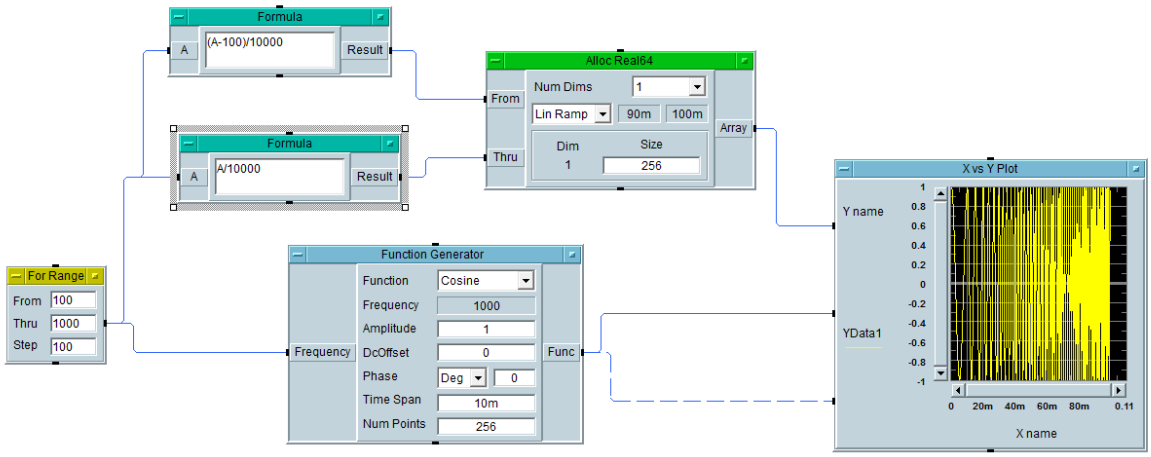

Fig. 37. Displaying successive waveforms