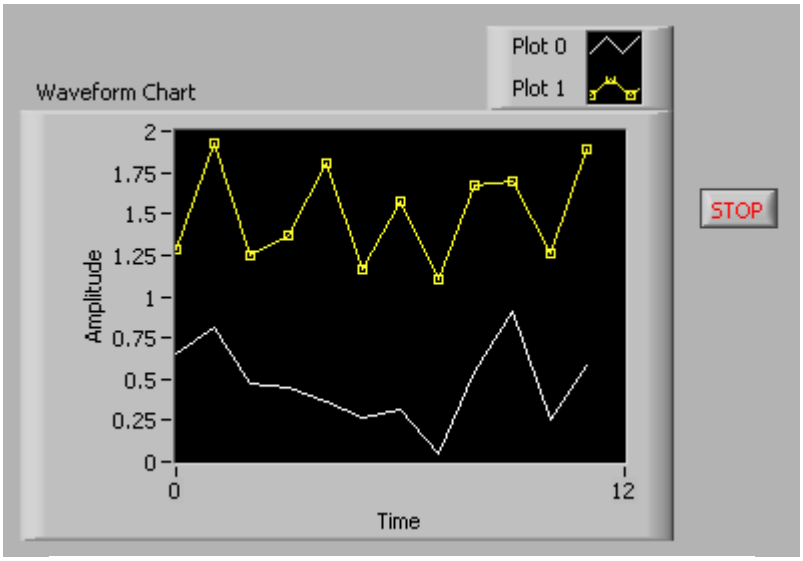
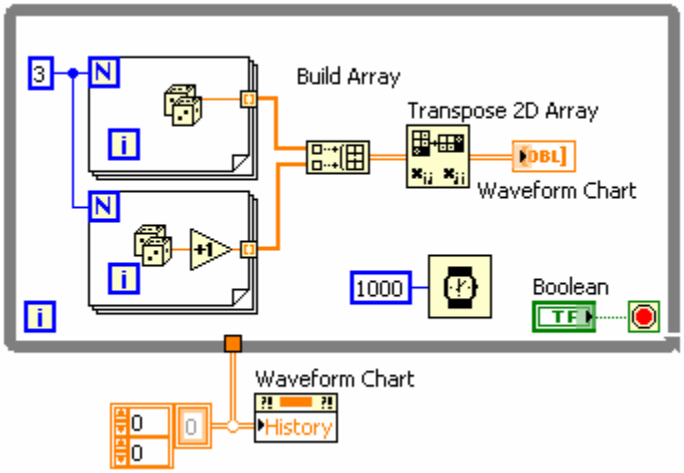
Supplement to **Figure 5.11** on page **78** of *LabVIEW Programming, Data Acquisition and Analysis*

The following figures show how to display multiple channels of *array* data using a Waveform Chart:





CHAPTER 5

Data Display

Data display is an essential aspect of almost any application. In fact, visualization capability is so important that the whole purpose of programming may be meaningless without it. Since LabVIEW is a virtual instrumentation tool that concentrates heavily on data acquisition, displaying data becomes an important part of G-Programming.

In LabVIEW, there are two methods for displaying data: either with charts or with graphs. And, within each method, there are different types. As for charts, there are two types: **Waveform Chart**, and **Intensity Chart**. As for graphs, there are three types: **Waveform Graph**, **XY Graph**, and **Intensity Graph**. Since the majority of applications utilize **Waveform Charts** and **Waveform Graphs**, they will be discussed first, in depth. Discussion about the remainder can be found in the latter part of this chapter. Let us begin with the **Waveform Chart**.

5.1 Waveform Chart

All of the charts and graphs can be obtained from the front panel. The **Waveform Chart** can be found under **Controls** >> **Graphs.** Before proceeding any further, you need to consider two important questions about displaying data in LabVIEW: 1) Do you want to display the data online or offline? and 2) Do you want to display a single line or multiple lines of data (multiple channels) on a graph or a chart?

> Generally, charts are used for online data display, and graphs are for offline display.

If you want to display real-time data acquisition, use a chart. If you want to first acquire data, save it, and then display it after acquisition is complete, use graphs. Consider the following example using the **Waveform Chart** in Figure 5.1 and Figure 5.2. Build the example following the diagram shown in Figure 5.2. You have a **While Loop** which is controlled by a Boolean switch **Quit**, and the **Waveform Chart** receives a random number from the random number generator at every iteration. In the front panel in Figure 5.1, change the maximum value of the *y*-axis from the default value of 10 to 1.

Since the default value of a Boolean switch is False, you will have to set the Boolean switch to the True position before you run the VI. Here, you will be introduced to the different settings of Boolean switches. Go to the front panel as shown in Figure 5.1, right click on the Boolean switch **Quit**, and select **Mechanical Action** from its pop-up menu. Choose **Latch When Released** and then set the switch **Quit** to the True position. Next, go to the **Operate** pull-down menu and select **Make Current Values Default**. This will cause the mechanical action **Latch When Released** to latch the switch back to the True position after generating False when it is pressed. At the same time, **Make Current Values Default** will set True (On) as the default state of the Boolean switch **Quit**. Then, the next time you open this VI, it will be ready to run since the switch is already at the True position. This simple combination of options may help you run a **While Loop** with the fewest preparation steps.

If you are using a **While Loop** with a Boolean switch to control it, first set the Boolean control to True. Select **Mechanical Action** >> **Latch When Released** from its pop-up menu, and choose **Operate** >> **Make Current Values Default**. Then, save the VI to preserve the settings.

More details about the rest of the mechanical action options are introduced in the next section. Assuming you have finished the VI in Figure 5.1 and Figure 5.2, run it to view the single channel data. The **Wait (ms)** is in the **While Loop** in Figure 5.2 to slow down the display of single channel data.

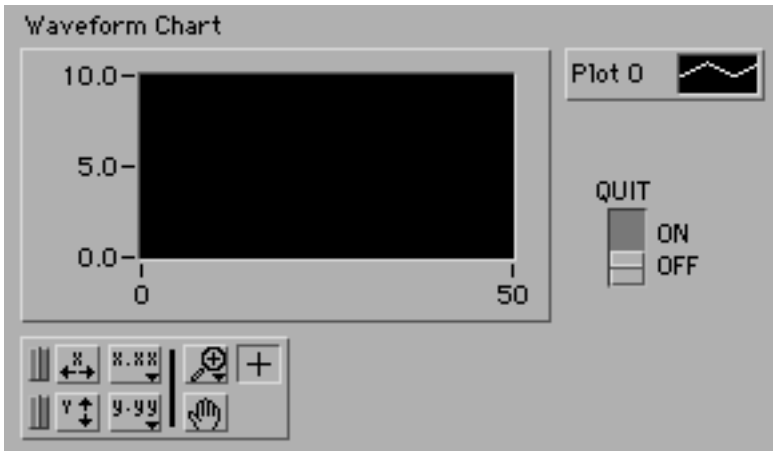


Figure 5.1 Front panel of an example VI with the **Waveform Chart**

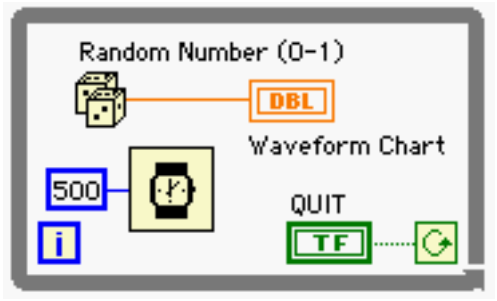


Figure 5.2 Diagram window of the VI in Figure 5.1

If you repeatedly run and stop the VI, you will realize that the chart does not refresh the screen. This means that the data from the previous run remains on the chart display. You can clear the display by selecting **Data Operations** >> **Clear Chart**. However, this can be done programmatically with **Attribute Node**. (See section 5.1.2 for the detailed discussion about the **Attribute Node**.) Before starting a discussion on multiple channel data display, the different mechanical actions of Boolean switches will first be discussed.

5.1.1 Mechanical actions of Boolean switches

Open an example from the following path: LabVIEW folder >> **examples** >> **general** >> **control** >> **booleans.llb** >> **Mechanical Action of Booleans.vi**. The front panel of the example is shown in Figure 5.3. This example has an excellent description of the six different mechanical actions you can set on Boolean switches. One of the options, **Latch When Released**, is useful with the use of **While Loop**, as discussed in the previous section, in combination with the option **Make Current Values Default** under the pull-down menu **Operate**. Execute the VI to see the differences between actions.

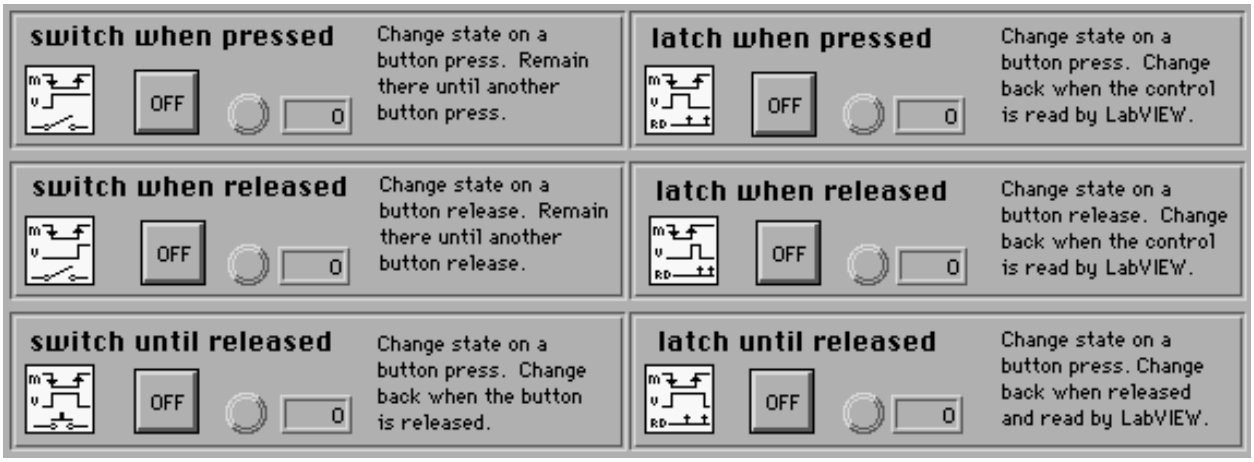


Figure 5.3 Front panel of **Mechanical Action of Booleans.vi**

5.1.2 Attribute Node

Users may encounter situations where they want to customize and control objects in the front

panel. For example, you may want to have a Boolean switch blink when a data reading such as temperature exceeds a preset threshold. You may want to make certain option buttons disappear if you do not want the users to be able to see them depending on different situations. Or, as mentioned in the previous section, you may want to clear the screen of the **Waveform Chart** every time you run the VI.

In LabVIEW, almost every object in the front panel has an Attribute Node with which you can change the object properties. As for the actions you can take with Attribute Nodes, there are two options available: 1) you can *read* the current setting of an object from its Attribute Node, or 2) you can *set* an object to desired settings. Suppose, for example, you want to programmatically position a Boolean switch in the front panel. To do so, you must first read the switch position to determine if repositioning is necessary. If it is, you would then change its coordinates to the desired position.

Some of the commonly used Attribute Nodes will be listed here, but you are encouraged to explore the others, too. Due to the wide variety of Attribute Nodes, it is not practical to discuss all of them. Therefore, only the most commonly used ones will be discussed.

Consider the **Waveform Chart** example shown in Figure 5.2 again. Right click to pop up the menu on the orange **Waveform Chart** with DBL indicating that it is displaying double precision type data. Select **Create** >> **Attribute Node** to create an Attribute Node as shown in Figure 5.4. Using the Hand Tool, click and hold on the icon to review all of the options available. A partial list of the options is as shown in Figure 5.5. As seen in Figure 5.5, there are quite a number of options available just for the **Waveform Chart**. Since some of the options are for general purpose, they will be explained in Figure 5.5 in the order they appear.

Figure 5.4 Attribute Node of the **Waveform Chart**

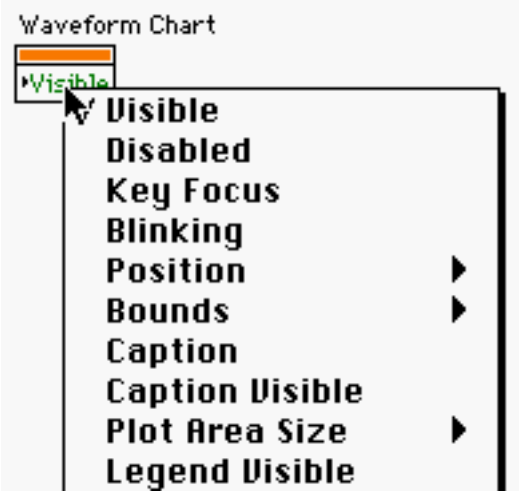


Figure 5.5 A partial list of options of the Attribute Node of the **Waveform Chart**

The first option, **Visible,** allows users to programmatically make the chart visible or invisible. Note that displays in the front panel in LabVIEW impose overheads onto the performance of VIs. Excessive use of overheads can significantly slow down the VI execution speed. Therefore, it is recommended to reduce the number of on-screen displays and display updates of graphs, charts, and numeric indicators. If you must have multiple **Waveform Chart**s in the front panel, you may optimize your VI by making charts visible only when you need to view them using Attribute Nodes. Therefore, using a Boolean switch in the front panel to control the visibility of charts can increase the execution speed of your VI.

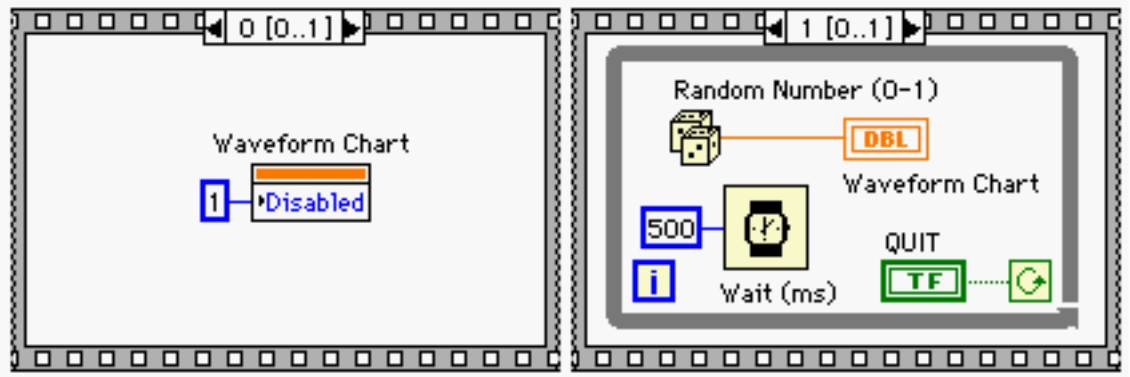


Figure 5.6 Attribute Node **Disabled** of the **Waveform Chart**

An example of the next option, **Disabled,** is shown in Figure 5.6. A few aspects in this example need to be pointed out. First, note that you are using the **Sequence** structure to make sure that the **Disabled** setting becomes effective *before* the main task (displaying data) starts. Since LabVIEW is a G-Programming language, the mere presence of objects in the diagram window will not specify the execution order. The order of execution is defined when objects are wired or a **Sequence** structure is used. (If two objects are connected through a wire, the one expecting data will *wait* until the data arrives.) To see the significance of the order of execution, consider a case where the Attribute Node refreshes the screen of the **Waveform Chart**. This option is the last one, **History Data** in Figure 5.5, and will be discussed again shortly. Then, consider the following question: What would happen if the **Waveform Chart** displayed the data followed by calling the Attribute Node that will refresh the chart display? All of the acquired data would disappear because the execution order is not correct! Such a situation can be avoided by using a **Sequence** structure to clarify the execution order.

The next aspect that needs attention is the arrow on the left side of the Attribute Node where numeric constant 1 is wired. If the arrow is on the left side, it implies that the Attribute Node is expecting an input setting. In other words, you can *set* a property to the object of the Attribute Node. If the arrow is on the right side, it means that you can *read* the current setting of the object that the Attribute Node belongs to (in this case, the **Waveform Chart**). You can change the direction by right clicking on the Attribute Node and choosing **Change To Read** or **Change To Write**.

The third aspect you must consider pertains to the creation of a correct input or output to the Attribute Node. Due to the wide variety of options of Attribute Nodes, it is impossible to remember all of the correct input or output data types for the Attribute Nodes. However, creating the correct inputs and outputs can be achieved easily by right clicking on the terminal and choosing either **Create Constant**, **Create Control**, or **Create Indicator**. This method was already introduced in Chapter 4 in the discussion about coercion dots. (See section 4.1.1.) Without using this method, it would be difficult to find out that the **Disabled** option in Figure 5.6 expects an unsigned 8-bit integer data type.

And finally, note that you can control more than one property with a single Attribute Node. To see this, go to frame 0 of the **Sequence** structure in Figure 5.6 and change the cursor to

the Arrow Tool. Point the Arrow Tool to the right bottom corner of the Attribute Node box until it turns to a reversed double L shape as shown in Figure 5.7. Now, click and hold the mouse button and drag it down to increase the number of elements, then release the button to complete the process. In Figure 5.7, there are four options selected, but there is no limit to the number of options you can have. The order of execution of these options is from top to bottom and the same item can appear in the list more than once. Also, each item can have different input or output directions so that the combination of reading and setting can be done in a sequential manner.

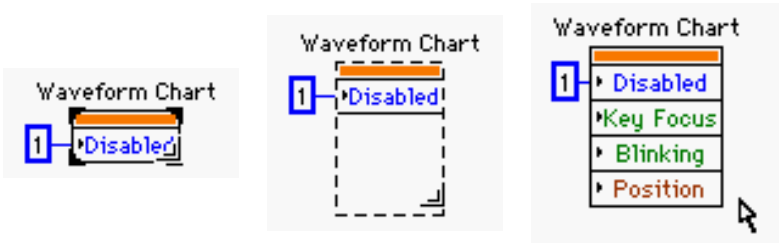


Figure 5.7 Multiple options with a single Attribute Node. A different direction can be assigned to each individual option.

The next option in Figure 5.5 is **Key Focus.** This option selects the object of the Attribute Node as default, but it does not apply to the **Waveform Chart** or the **Waveform Graph** since you do not enter inputs to them using keyboard keys. (You *select* an object such as a numeric control to enter inputs.) A good example of the use of **Key Focus** would be Boolean switches. If you select **Key Focus** for the option of the Attribute Node of a Boolean switch and wire a Boolean constant True to its input, the switch will automatically be selected when the Attribute Node is executed. Therefore, you would not need to move your mouse to select the Boolean switch. Instead, you can just hit the Return key since the switch would already be selected. This is a useful option if you have multiple switches in the front panel and want to have one of them selected as a default.

The next option, **Blinking**, will control the blinking of an object. For example, it can be used with a **Case** structure for displaying a warning message. If a certain condition happens, set an object to blink; otherwise, do nothing. The option **Position** allows users to programmatically change the position of an object in the front panel. **Plot Area Size** will control the plot size so that you can organize the screen more efficiently if you have a number of objects. **X Scale Info** and **Y Scale Info** are options that allow users to set the minimum, maximum, and increment of *x*

and *y*-axes of **Waveform Chart**s or **Waveform Graph**s. The last item that is commonly used is **History Data**. Figure 5.8 is an example of this option.

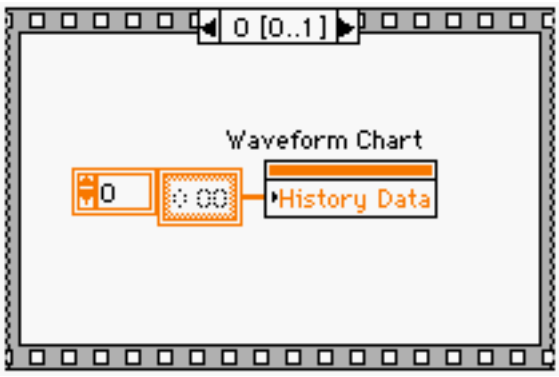


Figure 5.8 Attribute Node **History Data** of the **Waveform Chart**

To begin, modify frame 0 of the **Sequence** structure in the diagram window shown in Figure 5.6 to resemble Figure 5.8. In Figure 5.8, the input is a 1-dimensional (1-D) array constant that is created by selecting **Create Constant** from its pop-up menu. In this example, LabVIEW created a 1-D array of orange floating-point numbers because you have one channel of data to be displayed. If, however, you had multiple channels of data wired to the **Waveform Chart**, LabVIEW would have automatically adjusted the dimension of the input array to 2-D. For a detailed discussion about arrays, see Chapter 6.

Figure 5.8 shows that the input to the Attribute Node **History Data** is an empty array. Setting the history of **Waveform Chart** to an empty array results in deleting all of the traces of the previous data in the chart. Run the VI to see the effect. The display should be refreshed before graphing new data every time the VI executes.

The alternative method to using the **Sequence** structure to specify the execution order is illustrated in Figure 5.9. Even though the **While Loop** does not expect any data, the wire from the Attribute Node completes its end on the boundary of the **While Loop** without resulting in the Broken Run Arrow. The simple act of attaching the wire to the **While Loop** from the Attribute Node specifies that the **While Loop** will not run until the Attribute Node executes. (This trick also works with the **For Loop**.) Therefore, Figure 5.9 has the same effect as Figure 5.6: executing the Attribute Node first. To verify this, use the **Highlight Execution** to run the VI,

and you will see the array constant reaches both the Attribute Node and the *boundary* of the **While Loop** simultaneously, but the *content in* the **While Loop** does not start until the constant gets written to the Attribute Node.

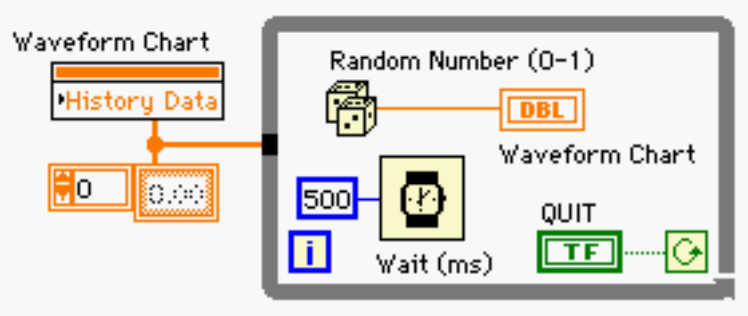


Figure 5.9 Alternative way to specify the execution order

As for the rest of the options in Figure 5.5, you are encouraged to explore them for yourself. When you do, you will likely find that some of them are self-explanatory and some are not. In order to find out what other options do, create a simple VI to test the effect of them before implementing them.

5.1.3 Update mode of the Waveform Chart

There are three different update modes for the **Waveform Chart**: **Strip Chart**, **Scope Chart**, and **Sweep Chart**. The **Strip Chart** is the default setting, and the screen scrolls as the display reaches the right end of the chart. The **Scope Chart** refreshes the display once the chart screen is full, but the **Sweep Chart** does not refresh the screen. Therefore, **Sweep Chart** mode will display new data over the old data display. In summary, the **Scope Chart** and the **Sweep Chart** both start from the beginning of the chart screen when the screen is full, but the **Scope Chart** does not keep track of the previous data (refresh its screen). Since the mode **Scope Chart** does not trace the previous data (does not have a memory), it has the fastest and most efficient memory usage. The **Sweep Chart** mode will be the next fastest, and the **Strip Chart** mode will be the slowest since it has to remember previous values to display. Changing the display mode is very easy. Simply click on the **Waveform Chart** and go to **Data Operations** >> **Update Mode** to choose one of the three modes. This can also be done programmatically with the Attribute

Node of the **Waveform Chart**.

In LabVIEW, the number of displays or display updates is closely related to the amount of overheads; therefore, updating many of the **Waveform Chart**s can consume quite a bit of memory and require significant execution time. To optimize your VI in terms of execution speed and overheads, you should minimize the number of image updates in the front panel. For example, suppose you have 1000 data samples to display. In this case, displaying them all at once will be much faster than displaying 100 points 10 times. A rule of thumb for optimizing overhead is that you should simplify the front panel of the VI and its operation as much as possible. This can be done by simplifying the number of updates in the front panel and hiding objects that need not be shown all of the time.

5.1.4 Multiple channels of data on the Waveform Chart

To understand how to plot more than one channel of data, a brief discussion of a new data type *bundle* will follow, and will later be presented in detail in Chapter 6. Consider Figure 5.10 and Figure 5.11. The new data type bundle is introduced in Figure 5.11. In this example, a wire specifies the execution order between the **While Loop** and the Attribute Node of the **Waveform Chart**. As you can see from the constant wired to the input of the Attribute Node, its type is different from the empty array you saw in Figure 5.8 and Figure 5.9. Three random number generator VIs are generating three data points simulating data from three different channels. In this example, you can see how **Bundle** combines them and sends them to the **Waveform Chart**.

The front panel in Figure 5.10 shows the screen shot of the VI while it is running. Note that the **Waveform Chart**s display a 2-D array of data (multiple channel data) column-wise. Consider the following 2-D array (matrix) **X** of size *M* by *N*, which is the collection of multiple channels of data:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1 & \cdots & \mathbf{x}_N \end{bmatrix} \tag{5.1}$$

LabVIEW returns data from multiple input channels in a 2-D array format as in (5.1). (This is discussed again in the chapters about data acquisition.) **X** is a matrix where each column $\mathbf{x}_i$ is the data vector of size *M* from the *i*th channel. Then, the *i*th line in the **Waveform Chart** will correspond to the *i*th data vector $\mathbf{x}_i$. Therefore, Plot 0 displays the vector $\mathbf{x}_1$, Plot 1 does $\mathbf{x}_2$,

and Plot 2 corresponds to $\mathbf{x}_3$ in Figure 5.10 if **X** in (5.1) is wired to the **Waveform Chart**. This completes the discussion about the **Waveform Chart**.

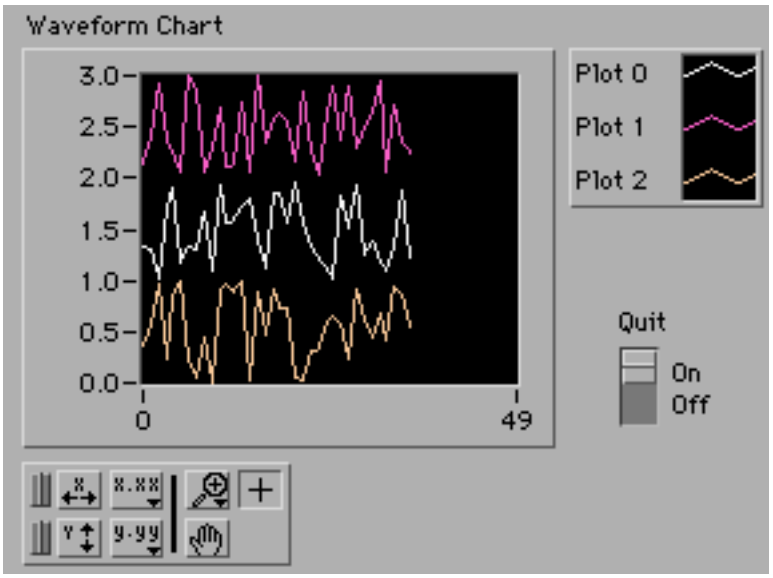


Figure 5.10 Displaying multiple channels of data on the **Waveform Chart** (front panel)

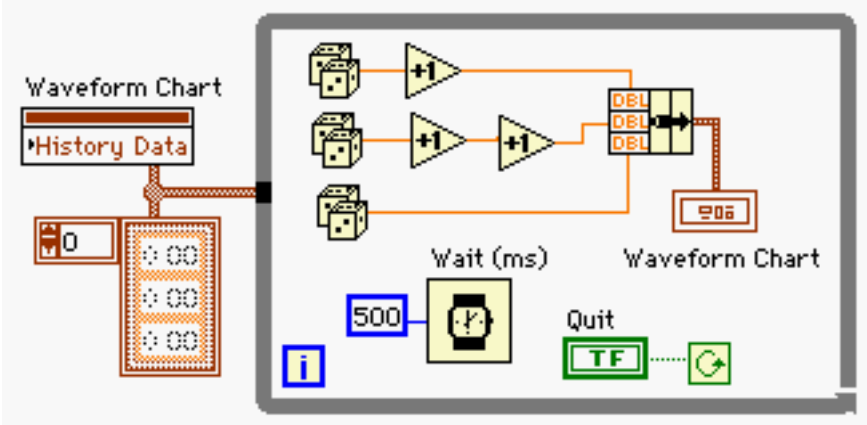


Figure 5.11 Displaying multiple channels of data on the **Waveform Chart** (diagram window)

5.2 Waveform Graph

Waveform Graphs are different from **Waveform Chart**s in many ways. First, **Waveform Graph**s are for offline display. In other words, **Waveform Graph**s display

data that is already acquired and saved, whereas **Waveform Chart**s display data in real time. Second, you can specify the initial point and interval of the *x*-axis with **Waveform Graph**s using **Bundle**, whereas **Waveform Chart**s do not allow it without using the Attribute Node. This is useful if you need to customize the *x*-axis scales when, for example, you are displaying the result of Fourier Transform. (This is discussed in detail in Chapter 14.) Third, you use **Build Array** to combine multiple channels of data to display them together on a single **Waveform Graph**, whereas **Bundle** is used for multiple channel data display on a single **Waveform Chart**. Fourth, **Waveform Graph**s display data row-wise, not column-wise. In other words, if you have a matrix **X** that is the collection of multiple channels of data as in equation (5.1), you must *transpose* it before you feed the matrix to the **Waveform Graph.** This is because each graph will correspond to each row vector instead of column vector in **X**.

Often, users forget to transpose the data matrix before displaying it on the **Waveform Graph**, so extra attention must be paid to the correct format of data. Transposing a matrix can be done easily by using **Transpose 2D Array** found under the **Functions** >> **Array** sub-palette. An alternative method to the use of **Transpose 2D Array** would be to select the option **Transpose Array** from the pop-up menu of the **Waveform Graph**. To do so, right click on the **Waveform Graph** and select **Transpose Array**. Specifying the initial value and interval of the *x*-axis is illustrated in Figure 5.12. The numeric constant zero is the initial value of *x*-axis, 0.5 is the interval of each data sample, and the array constant is a 1-D data to be displayed.

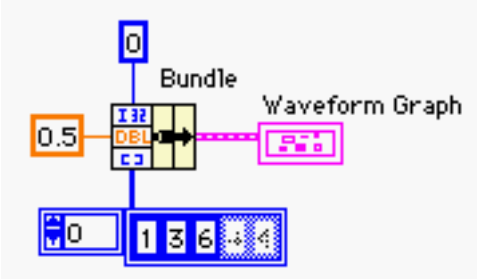


Figure 5.12 Specifying the initial value and the interval of *x*-axis with the **Waveform Graph**

5.2.1 Multiple channels of data on the Waveform Graph (I)

You can display multiple channels of data where the initial value and the interval can be individually specified. An example is shown in Figure 5.13.

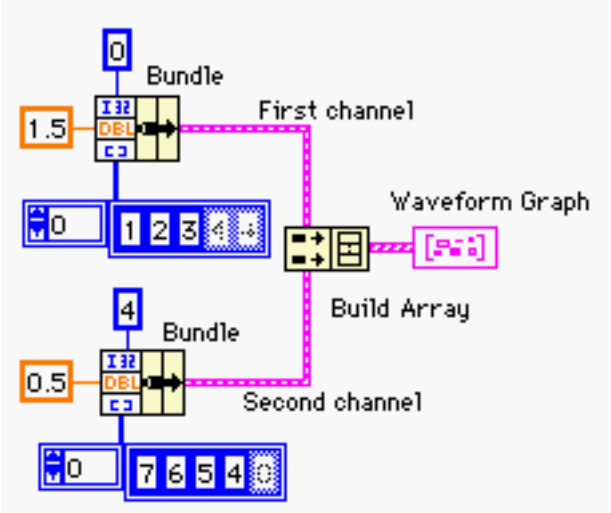


Figure 5.13 Multiple channel display with individual initial and interval values for *x*-axis. The lengths of the two data arrays from each channel can be different.

The first channel has 0 and 1.5 for its initial value and interval of *x*-axis, and the second one, 4 and 0.5. The bottom input of each **Bundle** is a 1-D array data. Also, notice that the thickness of the output wire of **Build Array** increases as you combine more input arrays. Thicker wire indicates the increment of dimension. If you have multiple channels of data with the same initial value and interval, you may realize it using both **Build Array** and **Bundle** as shown in Figure 5.14. **Build Array** and **Bundle** can be found in the **Array** and the **Cluster** sub-palettes in the **Functions** pop-up menu. More details about arrays and clusters are introduced in Chapter 6.

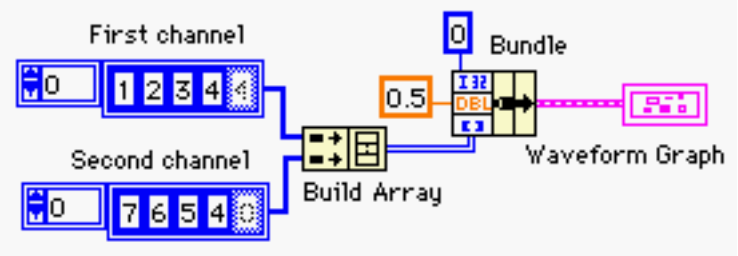


Figure 5.14 Multiple channel display with the same initial and interval value for *x*-axis. The lengths of the two data arrays must be the same. Otherwise, the smaller one will use 0 (zero) to match its length to the bigger size data array.

5.2.2 Multiple channels of data on the Waveform Graph (II)

The previous section discussed how to display multiple channels of data on the **Waveform Graph** while specifying the initial value and the interval of *x*-axis. This section will discuss how to perform the same task with default *x*-axis instead. The default *x*-axis starts at zero with interval one, and Figure 5.15 shows the diagram window to display two channels of data without specifying the initial value and the interval of *x*-axis. Note that this will provide a plot similar to that in Figure 5.10, but the **Waveform Chart**s as in Figure 5.10 are for real-time display.

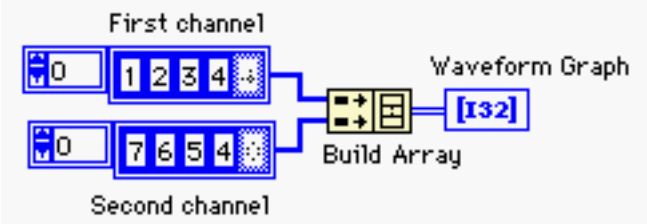


Figure 5.15 Multiple channel display with default *x*-axis. The lengths of the two data arrays must be the same. Otherwise, the smaller one will use 0 (zero) to match its length to the bigger size data array.

5.3 XY Graph

If you need to plot data for given *x*-axis values, the **XY Graph** can be used. For example, you want to plot sin(*x*) versus cos(*x*), as shown in the diagram window of Figure 5.16. Note that **Bundle** is used for a single channel display using the **XY Graph**. In case of multiple channels of data, **Build Array** needs to be used after **Bundle** as shown in Figure 5.17. Figure 5.17 shows an example of plotting sin(*x*) versus cos(*x*) and $\sin^{-1}(x)$. The *x* axis whose values are sin(*x*) is shared for both plots.

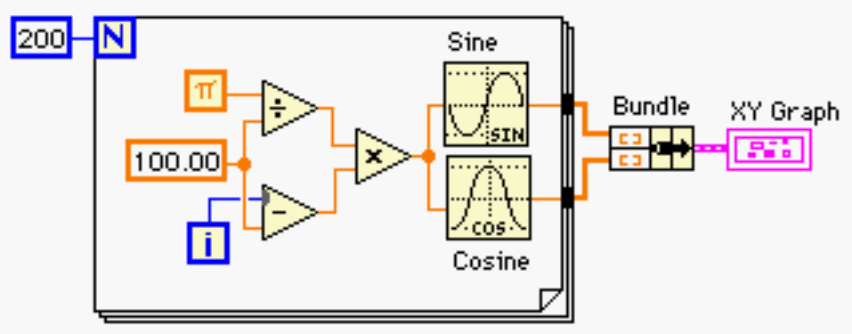


Figure 5.16 Single channel display with the **XY Graph**

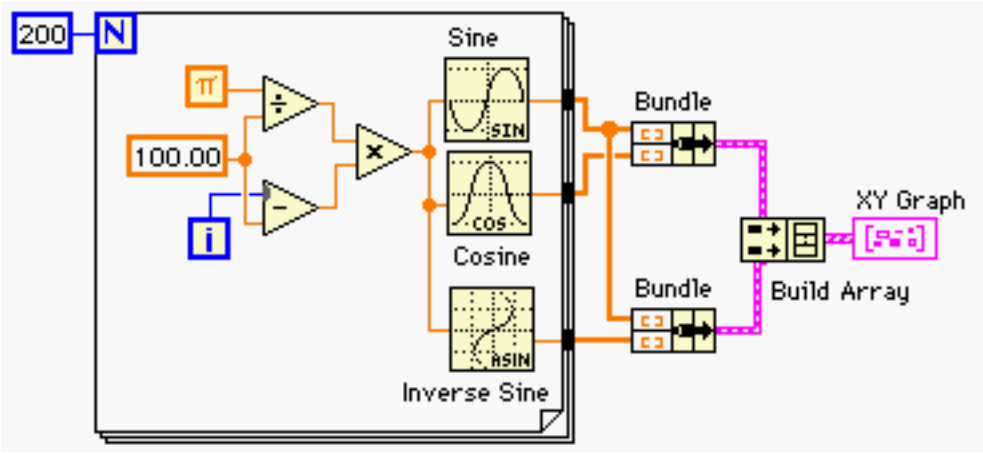
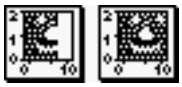


Figure 5.17 Multiple channel display with the **XY Graph**

5.4 Intensity Chart and Intensity Graph



Since the relationship between these two is very similar to that between the **Waveform Chart** and the **Waveform Graph**, such as the existence of memory, the major difference will be discussed here. The **Intensity Chart** and the **Intensity Graph** are especially useful if you have a 2-D array data (a matrix) where each element represents the density of a pixel of an image. The mapping between a data matrix **X** and the pixel index of the **Intensity Graph** is shown in Figure 5.18.

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & x_{13} & x_{14} \\ x_{21} & x_{22} & x_{23} & x_{24} \\ x_{31} & x_{32} & x_{33} & x_{34} \\ x_{41} & x_{42} & x_{43} & x_{44} \end{bmatrix}$$

x_{14}	x_{24}	x_{34}	x_{44}
x_{13}	x_{23}	x_{33}	x_{43}
x_{12}	x_{22}	x_{32}	x_{42}
x_{11}	x_{21}	x_{31}	x_{41}

Figure 5.18 Index mapping between the **Intensity Graph** and a 2-D array **X**

Therefore, the **Intensity Chart** and the **Intensity Graph** expect 2-D arrays as their input. You can also change the color for the value, and a couple of excellent examples of this are shown in the accompanying hands-on exercise manual. Interested readers may refer to it for the examples of **Intensity Chart** and **Intensity Graph**.

PROBLEMS

5.1 Create a VI **P05_01.vi** which performs the following:
(1) A chart **RN Display** displays a random number between 0 and 1 every 50 msec.
(2) **P05_01.vi** stops when you press a Boolean button **QUIT**.
(3) The default state of **QUIT** is TRUE.
(4) When **QUIT** is pressed, it depresses by itself when you release the mouse button.
(5) *y*-axis scale of the chart is between 0 and 1.

5.2 Modify **P05_01.vi** so that the display of **RN Display** refreshes every time you execute the VI using the Attribute Node **History Data**. Note that the option **History Data** is only available for **Waveform Chart**s, not for **Waveform Graph**s. Save the VI as **P05_02.vi**.

5.3 Create a VI **P05_03.vi** that has two **Waveform Chart**s, **RN Display 1** and **RN Display 2**, and each one performs the five tasks listed in Problem 5.1. Using the Attribute Node option **Visible** for both charts, create two Boolean buttons **Show 1** and **Show 2** to make each chart visible or invisible while displaying a random number continuously. Also, make sure that both charts refresh their screen whenever you restart the VI.

5.4 Using a **Waveform Chart**, display two channels of random numbers every 50 msec. One channel displays a random number between 0 and 1, and the other, between 1 and 2, in different color. Label the chart **RN 2Ch Display**, and adjust the *y*-axis scale to 0 and 2. Be sure to use the Attribute Node to refresh the screen whenever you execute the VI, and change the mechanical action and the default state of the Boolean switch **QUIT**. (Refer to Figure P5.4.) Also, label each channel **CH 0** and **CH 1** in the legend. Save the VI as **P05_04.vi**.

5.5 Using a **Waveform Graph**, display two channels of data, where each channel contains 10 random numbers. (Use Auto-indexing with a **For Loop**.) One channel has random numbers between 0 and 1, and the other, between 1 and 2. Use 0, 1, …, 9 for *x*-indices for both channel data. Label the graph **RN 2Ch Display**, and display the data from each channel using a different color. Also, label each channel **CH 0** and **CH 1** in the legend. The complete front panel is shown in Figure P5.5. Save the VI as **P05_05.vi**.

5.6 Modify **P05_05.vi** such that *x*-axis indices for both channel data are now 1, 3, 5, 7, …, 19, and all of the other information is identical. The complete front panel is shown in Figure P5.6. Save the VI as **P05_06.vi**.

5.7 Modify **P05_05.vi** such that *x*-axis indices for each channel are different. The indices are 1, 3, 5, 7, …, 19 for **CH 0**, and 3, 6, 9, 12, …, 30 for **CH 1**. All of the other information is identical, and the complete front panel is shown in Figure P5.7. Save the VI as **P05_07.vi**.

5.8 Realize **P05_07.vi** using a **XY Graph** in lieu of **Waveform Graph**, and all of the other information is identical. Therefore, the complete front panel should be the same as Figure P5.7. Save the VI as **P05_08.vi**.

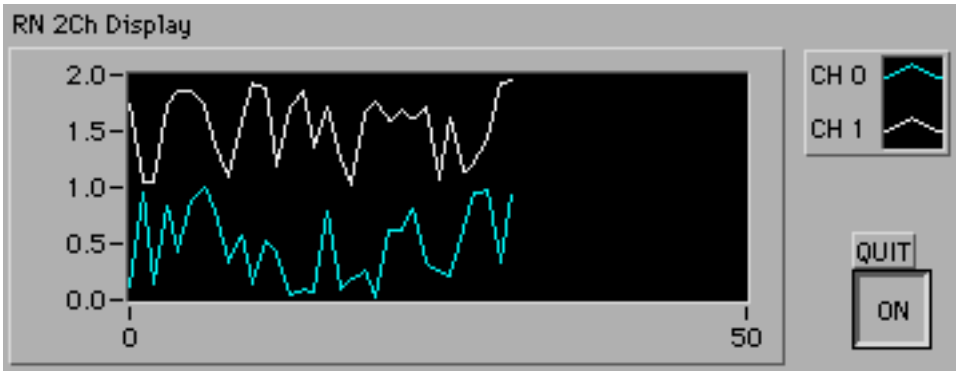


Figure P5.4

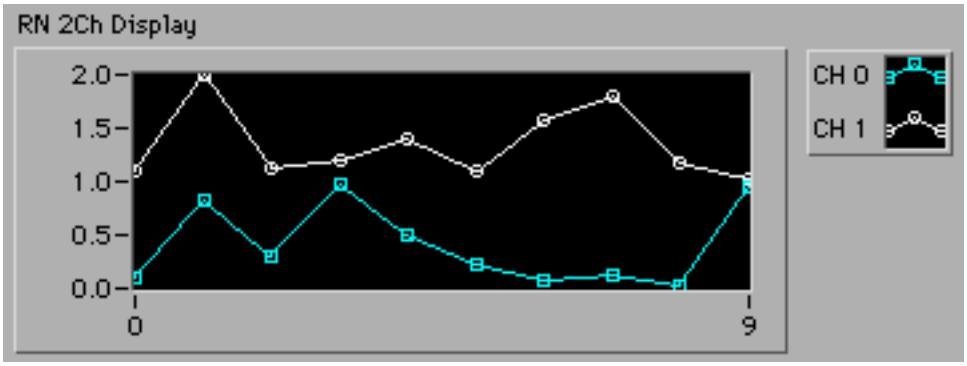


Figure P5.5

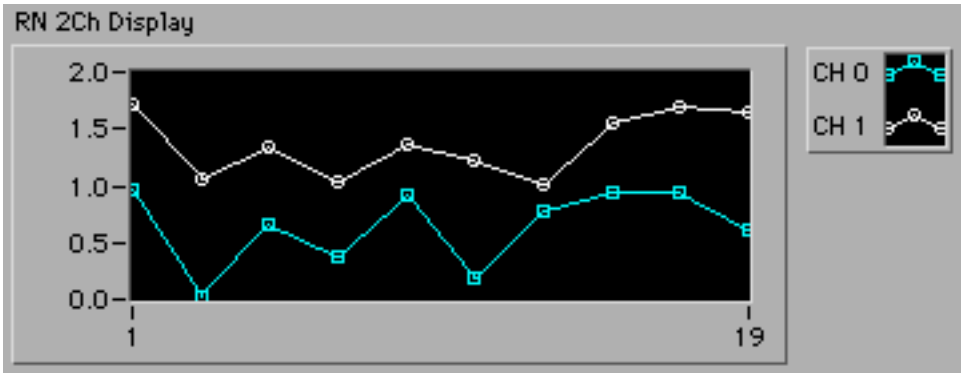


Figure P5.6

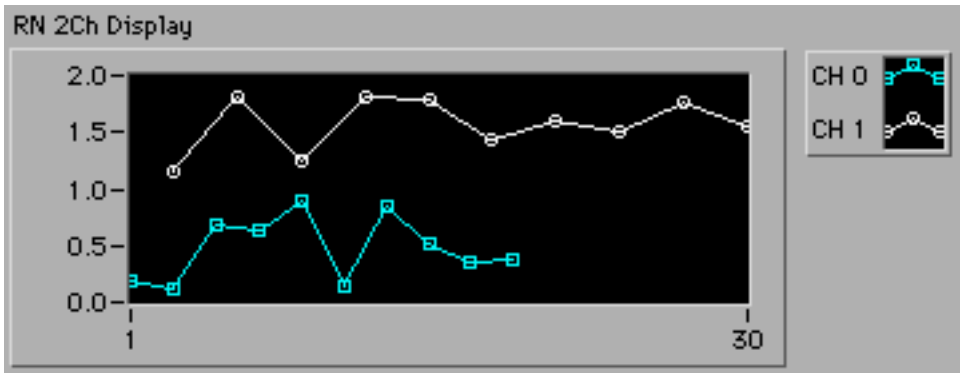


Figure P5.7