Lesson 6

Manipulating Arrays, To/From Files, and Statistical Parameters

This lesson will examine how to store data using arrays and To/From files. It consists of a pre-lab and four labs.

Lesson 6 Pre-lab Summary

The following are described in the Lesson 6 Pre-Lab. See Appendix E, page E-45.

- Menu Bar => Data => Collector
- Menu Bar => Data => Concatenator
- To/From Files
- Introduction to Understanding I/O Transactions
- Introduction to Arrays

As noted in all previous lessons, Appendix B includes a cross-reference to each of these items and to all objects and subprograms contained in the labs of this and later lessons.

Overview

Lab 6.1 – Creating Arrays and Manipulating Array Data
This lab will show you how to use a Collector object and math expressions to create and display an array of data and how to use a Concatenator object to combine scalars and array elements into a single array.

Lab 6.2 – Storing and Accessing Data Using To/From File Objects
This lab will show you how to move test data to and from files. You will learn to store and retrieve three common test result items: a test name, a time stamp, and a one-dimension array of real values. (The same general process will apply to all VEE Pro data types.)

Lab 6.3 – Calculating Statistical Parameters
This lab will show you how to set up a function generator, graph its waveform time-domain output, and calculate the waveform mean, median, mode, variance, standard deviation, and rms values.

Lab 6.4 – Logging Vehicle Radiator Statistical Data
This lab will show you how to simultaneously monitor Vehicle Radiator temperature and pressure data, displaying the results of calculating standard deviation and root-mean-square statistical analyses of these data points.

Lab 6.1 – Creating Arrays and Manipulating Array Data

This lab will show you how to use a Collector object and math expressions to create and display an array of data, and how to use a Concatenator object to combine scalars and array elements into a single array.

Open your VEE Pro program and
1. Clear your Work Area and maximize Main.

Using the collector
2. Select Menu Bar => Flow => Repeat => For Count; place it at the left-center of your workspace; highlight the default count of 10 and replace it with: **4**.
3. Select Menu Bar => Data => Collector; place it to the right of the For Count object.
   Note: See Appendix E for the Collector description.
4. Select Menu Bar => Display => AlphaNumeric; place it to the right of the Collector object.
5. Double-click the Collector object to get its open view.
6. Read through Help in the For Count and Collector object menus to understand these objects.
   Note 1: The for-count output value is a Real64 scalar. (Real64 is a constant 8-byte real number.) It starts at 0 and counts to n-1, where **n** is the specified count value. The number of counts depends upon the number of iterations specified in the input field.
   Example: If For Count is to generate outputs of 0, 1, 2, and 3, then change the default number to 4, as in step 2 above.
   Note 2: The Collector is an object that collects data in a loop and provides an output array shape. The one-dimensional array is created from input data (generally scalar). The n+1-dimensional array is collected from input arrays. If the input arrays have "n" dimensions, the collector can either prepare an array of n+1 dimensions, or it can prepare a one-dimensional array. When all data have been collected, activating the XEQ pin will cause the resulting output array to propagate via a "ping".
7. Click n+1 Dim in the Collector to toggle the selection to 1 Dim Array.
8. Connect For Count data-output pin to the data-input pin on the Collector.
9. Connect For Count sequence output (bottom) pin to XEQ input pin on the Collector.
   Note: The XEQ pin is a control pin that exists on several different objects. It tells VEE Pro when you want that object to execute. For the above example, you want the object to fire (execute) after all of the data points from the array have been collected. In this example, XEQ is pinged (executed) when the For Count object output reaches **3**. Until then, the Collector's output will not propagate.
10. Connect the Collector data-output pin to the AlphaNumeric data-input pin.
11. Enlarge AlphaNumeric to accommodate the four-element array.
12. Run your program. See Figure 6.1.

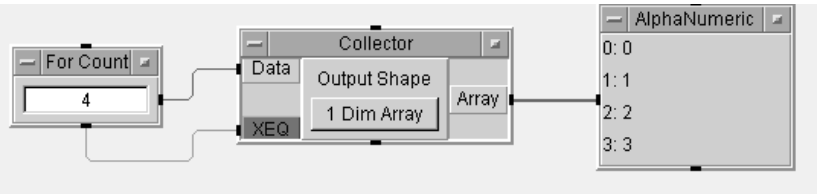

Figure 6.1. The Collector creating a displayed array

Note: The principles are the same regardless of the array size. The Collector will take any data type and create the array size, depending upon the number of elements sent.

Extracting values from an array

13. Delete the data line between the Collector and AlphaNumeric as follows:
    a. Place the mouse pointer over the line; click the mouse right button; select Delete Line; cut the line when the scissors appear. **OR:**
    b. Press Shift-Ctrl and click the mouse left button. **OR:**
    c. Use the right-hand scissors icon on the toolbar; mouse pointer changes to scissors; left-click on the line to be deleted.
14. Convert the Collector to an icon.
15. Move AlphaNumeric to the right so there is room for another UserObject.
16. Select Menu Bar => Device => Formula; clone it; place both Formula objects to the right of Collector.
17. Connect the Collector data-output pin to the data-input pins of the two Formula objects.
18. Enter A[2] in the upper Formula input field.
19. Enter A[3] in the lower Formula input field.
    Note: A[2] will extract the third element of the array as a Scalar because the indexes start at zero.
20. Clone AlphaNumeric.
21. Connect an AlphaNumeric display to each Formula object.
22. Run your program; the upper display should show a value of 2, while the lower display should show a value of 3.
23. Enter A[4] in the upper Formula input field.
24. Run this modified program. What happens? (Close the error pop-up box.)
25. Select Menu Bar => Debug => Stop.
    Note: This will release the program from the error mode.
26. Return the upper Formula input field to A[2].
27. Enter A[1:3] in the lower Formula input field.
28. Run this modified program; it should look like Figure 6.2.
29. Save this modified program as LAB6-1.

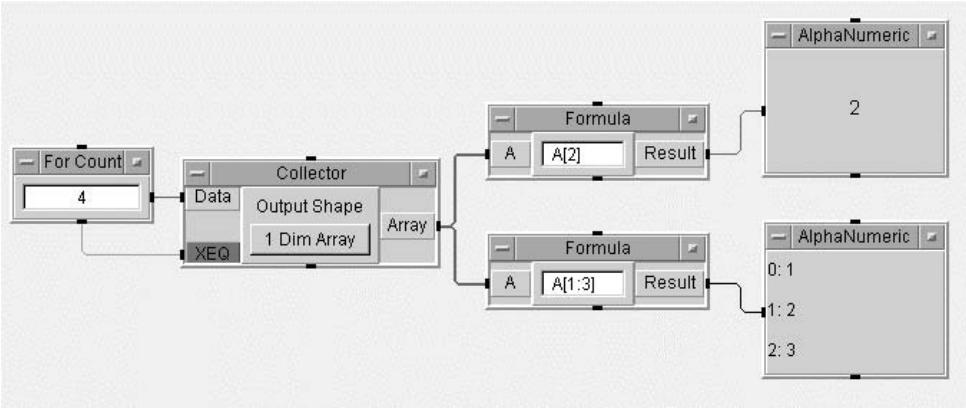

Figure 6.2. Extracting array elements with expressions

Using the Concatenator

30. Select Menu Bar => File => New.
31. Save As … LAB6-1a immediately.
32. Select Menu Bar => Data => Concatenator; place in the right-center of the screen.
33. Open the Concatenator icon; select Add Terminal; Data Input.
34. Select Menu Bar => Display => Alphanumeric; place it to the right of the Concatenator; expand it to allow for nine data lines; connect it to the Concatenator output pin.
35. Select Menu Bar => Data => Allocate Array => Real64; place it to the left of the Concatenator; connect its Array output to the Concatenator input-pin B.
36. Change Allocate Array Num Dims to 1; select Lin Ramp ─ range from 90 to 96; change its size to 7.
37. Select Menu Bar => Data => Constant => Text; place its object above Alloc Real64; connect its output pin to the Concatenator input-pin A; change its data value to Test#n.
38. Select Menu Bar => Constant => Real64; place its object below Alloc Real64; connect its output pin to Concatenator input-pin C; enter the number 82.06 into its data box.
39. Save this program again.
40. Run this program; it should look like Figure 6.3.

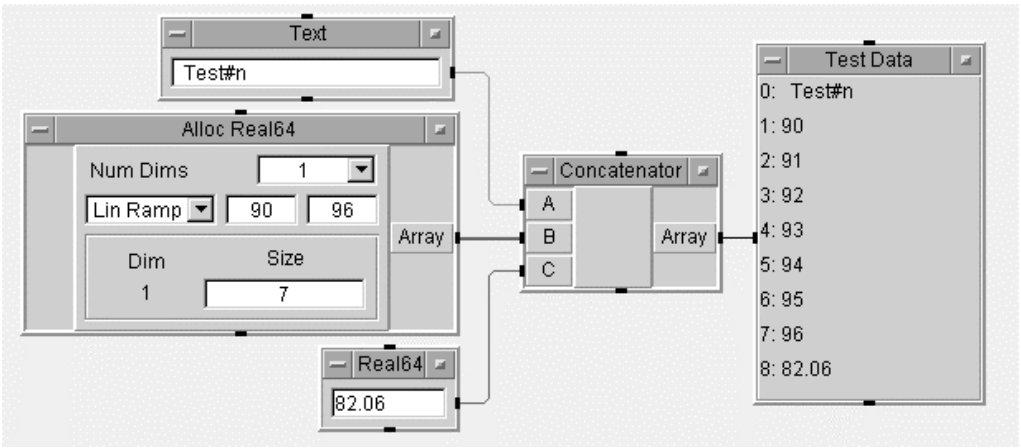

Figure 6.3. Creating arrays with the Concatenator (Lab 6-1a)

◊ 41. Experiment with several values in Alloc Real64.
Note: The Concatenator retains the data and displays it in the sequence in which it is received by its pin sequence: A, then B, then C.
42. Close this program; do not save this program again.

Lab 6.2 – Storing and Accessing Data Using To/From File Objects

This lab will show you how to move test data to and from files. You will learn to store and retrieve three common test result items: a test name, a time stamp, and a one-dimension array of real values. (The same general process will apply to all VEE Pro data types.) This lab should be performed without interruption.

Open your VEE Pro program and

1. Clear your Work Area and maximize Main.

Sending a text string to a file

2. Select Menu Bar => I/O => To => File.
Note 1: The To/File default file name is myFile; it appears in the ToFile object. This default can easily be changed by clicking on myFile. (The button to the right of the ToFile: input field label); a box with a list of files will be displayed from your home directory. See Figure 6.4.

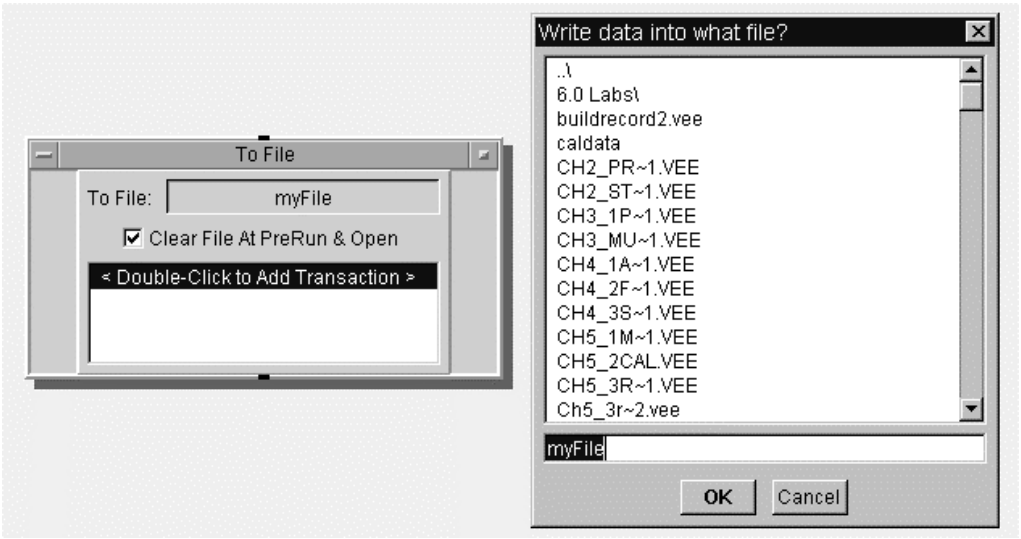

Figure 6.4 The ToFile object and its Home Directory box

Note 2: Click the box next to Clear File At PreRun & Open; by default, VEE Pro appends new data to the end of an existing file. Check the box; the file is now cleared before you write new data.
Note 3: Finally, WRITE TEXT a EOL is the default transaction. It means that you will write the data of variable a on pin A using TEXT encoding and a specified end-of-line sequence. VEE Pro is not case-sensitive with regard to pin names, so you may use lower-case or upper-case strings to for pin names. See Figure 6.5.

3. Double-click the transaction line shown in Figure 6.4 ("Double-Click here …")to open the I/O Transaction dialog box of Figure 6.5.
   Note 1: This is the default transaction. It means that you will write the data of variable **a** on pin **a** using TEXT encoding and a specified end-of-line sequence.
   Note 2: VEE Pro is not case-sensitive when labeling pin names; you may use lower-case or upper-case strings for pin names.

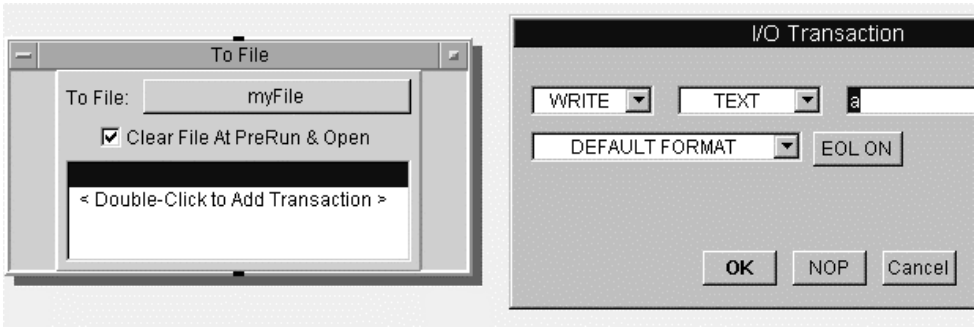

Figure 6.5. An I/O Transaction dialog box

4. In the I/O Transaction dialog box, the expression list field (the right-most field in the top row) is highlighted; type "Test1" in that field, including the quotation marks.
   Note 1: You need the quotation marks to indicate a Text string. If you typed Test1 without the quotation marks, VEE Pro would interpret the entry as a pin name or a global variable. You can leave the other defaults, since you want the action WRITE.
   Note 2: The data will be sent in the following manner:
   - The encoding TEXT will send the data using ASCII characters.
   - The DEFAULT FORMAT will choose an appropriate VEE Pro format such as STRING.
   - The default EOL sequence is the escape character for a new line, \n.
5. Click OK; the transaction line of the To File object should now display
   WRITE TEXT "Test1" EOL. This transaction will send the string Test1 to the specified file, myFile.
   Note: If you must shut down at this time, save your work as LAB6-2.

Sending a time stamp to a file

Note: This is a continuation of the prior ToFile transaction; the ToFile object we just developed must be on-screen. The function now( ) is obtained via the following menu selections: Menu Bar: Device => Function & Object Browser => Time & Date submenu provides the current time expressed as a Real Scalar. The value of the Real is the number of seconds elapsed since 00:00 hours on Jan. 1, 0001 AD. As of this date, now( ) returns a value of about 63 gigaseconds. VEE Pro provides this format because it is easier to manipulate mathematically. It also conserves storage space. However, if you want to store the time stamp in a more readable string format, use the TIME STAMP FORMAT in the ToFile object.

6. Open LAB6-2 if necessary; add an input pin to the To File object.
7. Double-click just below the first transaction line in the ToFile object; this will add a second transaction and open a new I/O Transaction box.
8. Double-click the expression list input field to highlight the a (if necessary) and type the expression: now( ).
   Note: The now() function will send the current time from the computer clock to a Real format, which we will next change to the Time Stamp Format.
9. Go to the I/O Transaction box:

10. Click the down-arrow next to the DEFAULT FORMAT field to open a menu, and select TIME STAMP FORMAT. VEE Pro now adds some additional buttons and fields to the I/O Transaction box.
11. Click the down-arrow next to Date & Time, select time.
12. Click HH:MM:SS (hour, minute, seconds format); it will toggle to HH:MM (the hour and minute format).
13. Click 24 HOUR; it will toggle to 12 HOUR (providing the am/pm format).
    Your I/O Transaction box should look like Figure 6.6. If it does, click OK.

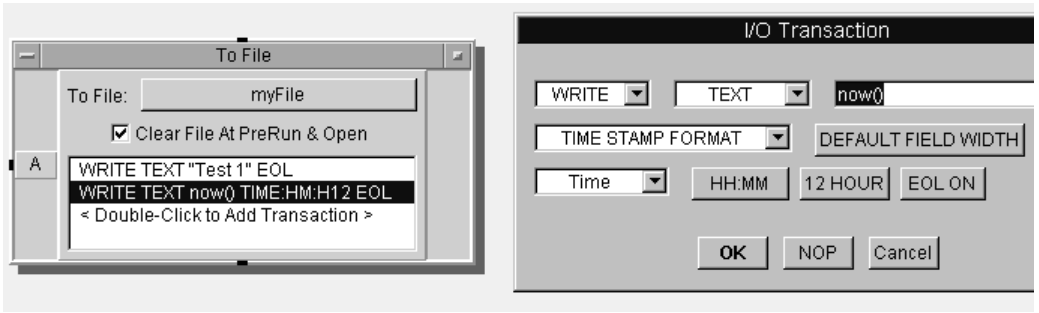

Figure 6.6. The TIME STAMP I/O Transaction box

Sending a real array to a file

We will now create a one-dimension array of four elements using the For Count and Collector objects. We will append this to myFile.

14. Maximize your workspace; drag the ToFile object to the right.
15. Select Menu Bar => Flow => Repeat => For Count; double-click the For Count input field and replace the default number with **4**; move the object to the left-center of your workspace.
16. Select Menu Bar => Data => Collector; place it between the two objects in the workspace.
17. Connect the data-output pin of the For Count object to the data-input pin of the Collector object.
18. Connect the For Count sequence output pin to the XEQ pin of the Collector.
    Note: The Collector will now create the array [0,1,2,3], which you can send to your ToFile data object containing myFile.
19. Connect the Collector data output pin (Array) to the A pin of the ToFile object.
20. Double-click below the second transaction line in the ToFile object to add another I/O Transaction box.
21. Go to the Transaction dialog box:
22. Open the DEFAULT FORMAT menu and select REAL32 FORMAT.
    Note: This offers a new choice of selections. You can leave the default choices, although you may want to investigate them for future reference.
23. Click OK to close the I/O Transaction box and complete the process. The third line of the To File object will now display WRITE TEXT a REAL32 STD EOL. Figure 6.7 shows the function, its ToFile list, and its I/O Transaction box.

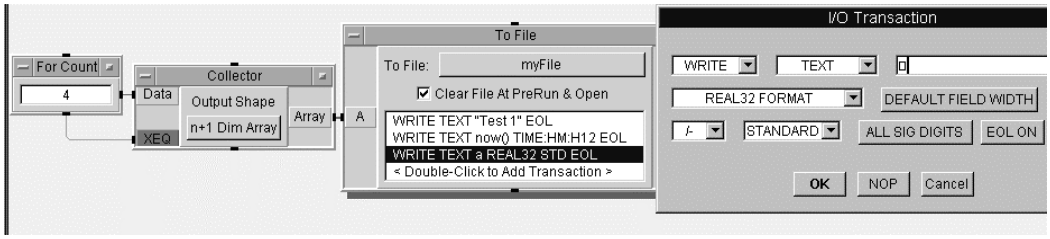

Figure 6.7. Storing data using the ToFile object

24. Save this program as LAB6-2 before proceeding.

Retrieving data using the FromFile object

Note: At this moment, you know the format and location of stored data. You will later learn to retrieve data without having to know its type. In the present example, you know that you stored the name of a test in a String Format, followed by a time stamp in Time Stamp format, then followed by an array of Real numbers. You will now create three transactions in FromFile to read that data back into VEE Pro.

25. Select Menu Bar => I/O => From => File and place it below your ToFile object.
26. Connect the sequence output of the ToFile object to the sequence input of the FromFile object.
    Note: This ensures that ToFile has finished sending data to myFile before FromFile begins to extract data (from myFile).
27. Leave the default file of the FromFile object as myFile.
28. Double-click the From File transaction line to add an I/O Transaction box, because you want tell FromFile how to read the data.
29. Click the format field down-arrow in the FromFile I/O Transaction box; change REAL FORMAT to STRING FORMAT; click OK to close the FromFile I/O Transaction box because all of the other defaults are correct.
    Note 1: The first transaction line of the FromFile object should read:
    READ TEXT x STR.
    This will read the first line of the file as format Text, into the "x" output terminal.
    Note 2: Two more transactions are required to read back the time stamp and the real array.
30. Add a data output to the FromFile object by clicking the mouse cursor over the output area and simultaneously depressing **Ctrl-a**. A dialog box will appear labeled "Select output to add" will appear.
31. Select Y and click OK. VEE Pro will add the data-output pin Y.
32. Add a third terminal by repeating the above procedure. VEE Pro will now offer a data-output pin labeled Z.
33. Select Z and click OK. You now have three outputs labeled X, Y, and Z.

Storing the time stamp

34. Double-click below the first transaction line of the FromFile object to add an I/O Transaction.
35. Go to the FromFile I/O Transaction dialog box.
36. Double-click the expression list input field to highlight **x** (if needed), and type **y**. This causes the second transaction to read data back to output pin **y**.

Note: If this pin remained **x**, the second transaction would overwrite the data placed into **x** by the first transaction.

37. Change REAL FORMAT to STRING FORMAT; then click OK.
    Note: If you want to read the time stamp back as a text string, use STRING FORMAT encoding. The TIME STAMP FORMAT converts the time stamp data back to a Real number. See Figure 6.8.

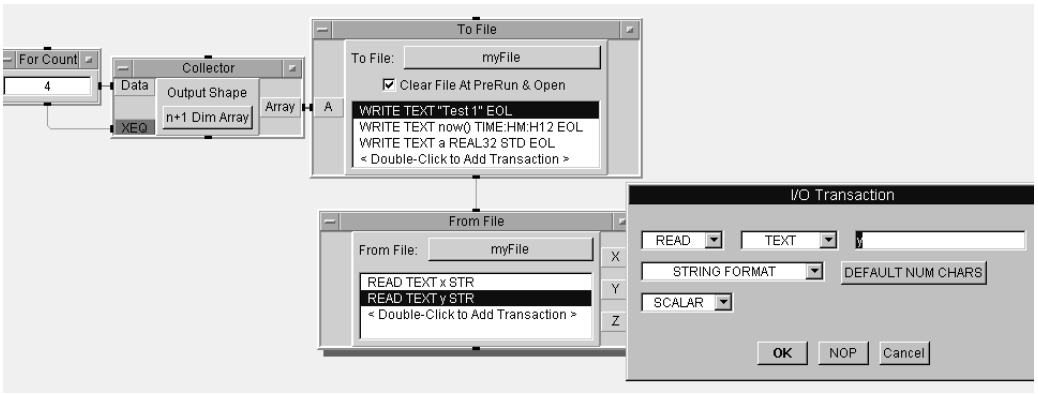

Figure 6.8. Preparing to read data using the FromFile object

Storing real numbers

38. Double-click below the third transaction line to open the I/O Transaction dialog box, go to the expression list input field; double-click to highlight x (if needed); type in z so that the Real array is read back to the Z output pin of the FromFile object.
39. Leave REAL64 FORMAT unchanged as it is correct for this case.
40. Change SCALAR to ARRAY 1D and SIZE to 4; click OK.
    Note 1: If you cannot recall the size of array, you may toggle SIZE to TO END. This will read data to the end of the file without VEE Pro knowing its exact size. You could use this feature to read the entire contents of a file as a string array to examine the file contents.
    Note 2: The FromFile object will now display READ TEXT y STR on the second transaction line, and READ TEXT z REAL64 ARRAY 4 on the third transaction line.
41. Select Menu Bar => Display => AlphaNumeric; clone it twice to get three displays.
42. Connect each output pin of the From File object to the data input pin of an AlphaNumeric display. Size the array connected to the FromFile Z output to hold four values.
    Note: Remember – you can size AlphaNumeric displays as you clone them by simply clicking and dragging the object outline when you first select it from the menu. This procedure works for any object.
43. Run the program. It should look like Figure 6.9.
44 Save again LAB6-2a.

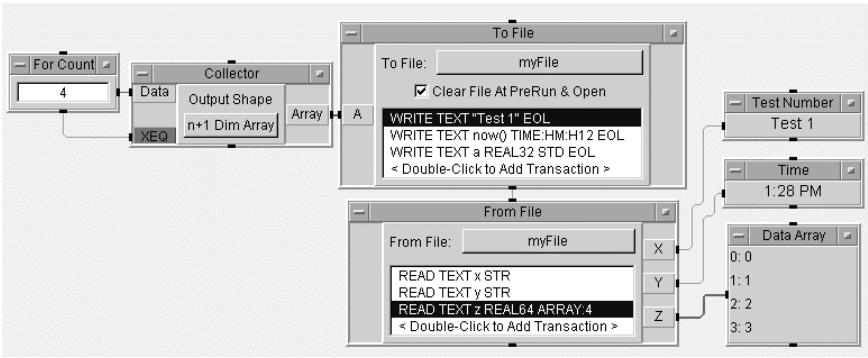

Figure 6.9. Retrieving and displaying data using the FromFile object

Lab 6.3 – Calculating Statistical Parameters

This lab will show you how to set up a function generator, graph its waveform time-domain output, and calculate the waveform mean, median, mode, variance, standard deviation, and rms values.

Open your VEE Pro program and

1. Clear your Work Area, maximize Main; toggle Program Explorer off.

Calculating various statistical parameters

2. Select Menu Bar => Device => Virtual Source => Function Generator.
3. Place Function Generator in the left-center of your Work Area; change Frequency to 100 and DcOffset to 1.
4. Select Menu Bar => Device => Function & Object Browser => Type: Built-in Functions; Category: Probability & Statistics; Member: mean; click Create Formula; mean(x) will appear. Place it to the top right of the Function Generator object.

   or

   Click f_x on the Tool Bar for Function & Object Browser => Type: Built-in Functions; Category: Probability & Statistics; Member: mean; click Create Formula; mean(x) will appear.

Note: Examine Help from the mean(x) object menu for more details.

5. Repeat step 4 for the following statistical parameters:

   median, mode, variance(vari), standard deviation(sdev), and root-mean-square(rms).
6. Place each Formula object vertically under the mean(x) object.
7. Select Menu Bar => Display => AlphaNumeric; place it to the right of mean(x).
8. Clone the AlphaNumeric object five times; place them beside each Formula Object.
9. Change the names of each AlphaNumeric Object to match the name of its Formula Object.
10. Connect each Formula Object input pin to the Function Generator output (Func) pin.
11. Connect each renamed AlphaNumeric Object input pin to its corresponding Formula Object output pin (Result).
12. Select Menu Bar => Display => Waveform (Time); place it to the right of all of the AlphaNumeric objects.
13. Connect the Waveform (Time) input pin (Trace 1) to the Function Generator output pin.

14. Run your program. It should look like Figure 6.10.

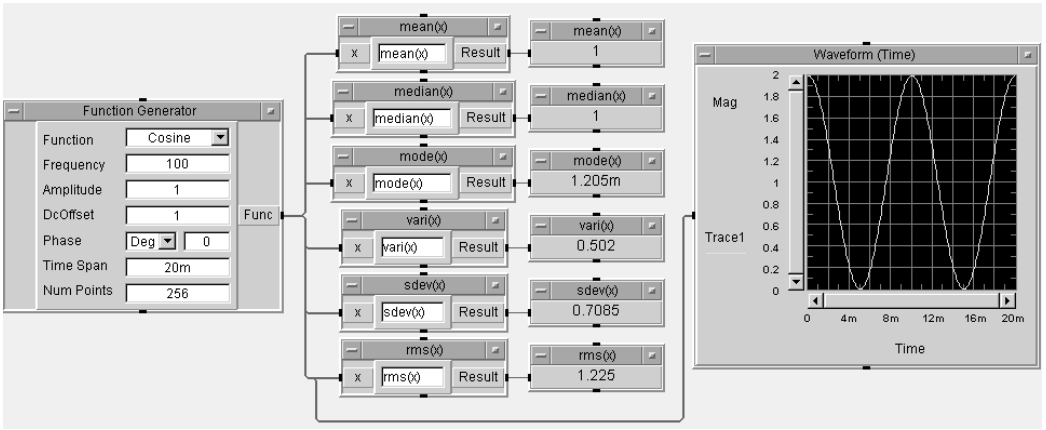

Figure 6.10. Calculating statistical parameters

15. Save your program as LAB6-3.

◊ 16. Vary the following Function Generator parameters: Frequency, DcOffset, Function, Phase, and Num Points (in exponents of 2: 2, 4, 8, 16, 32, etc).
Note: If you change the FncGen frequency, then you must also change the Time Span of FncGen and Waveform (Time) to be the reciprocal of the frequency value.

Your notes:

_____

_____

_____

_____

_____

17. Do *not* save the modifications to this program.

Lab 6.4 – Logging Vehicle Radiator Statistical Data

This lab will show you how to simultaneously monitor Vehicle Radiator temperature and pressure data, displaying the results of calculating standard deviation and root-mean-square statistical analyses of these data points.

Open your VEE Pro program and

1. Clear your Work Area, maximize Main; toggle Program Explorer off.

Displaying a waveform

2. Select Menu Bar => File => Open; open LAB3-4. It should look like Figure 6.11.

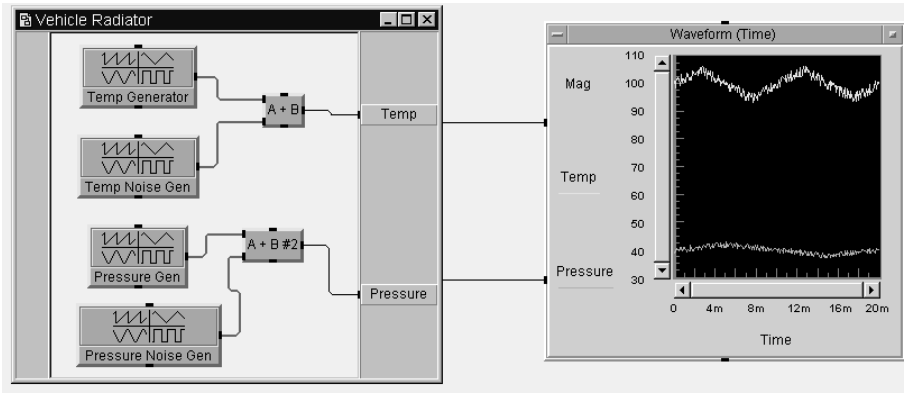

Figure 6.11. The original LAB3-4 after running

3. Reduce Vehicle Radiator to an icon; save As… this program As LAB6-4 immediately.
4. Move Waveform (Time) to the far right of your screen border.
5. Move Vehicle Radiator to the far left of your screen border.
6. Select Menu Bar => Device => Formula; place it between Vehicle Radiator and Waveform (Time), next to Vehicle Radiator.
7. Select Add Terminal, Data Output twice; use Delete Terminal, Data Output to remove "Result"; change the output-pin letters to B (top) and C (bottom).
8. Insert Formula data into the white (editing) space as B=sdev(A);C=rms(A).
9. Clone Formula; place the clone below the other Formula Object.
10. Change the upper Formula Object name to Temperature; change the lower Formula Object name to Pressure.
11. Connect the Temperature object input pin A to the Vehicle Radiator Temp output pin; connect the Pressure object input pin A to the Vehicle Radiator Pressure output pin.
12. Select Menu Bar => Display => Logging AlphaNumeric; clone it three times; place them in a square format (two over; two under) between the Formula and Waveform (Time) objects.
13. Change the name of the upper-left Logging AlphaNumeric to Temp sdev; change the name of the upper-right Logging AlphaNumeric to Temp rms.
14. Change the name of the lower-left Logging AlphaNumeric to Pressure sdev; change the name of the lower-right Logging AlphaNumeric to Pressure rms.
15. Connect the Temperature object data-output pin, pin B to the Temp sdev data-input pin (Data).
16. Connect the Temperature object data-output pin, pin C to the Temp rms data-input pin (Data).
17. Connect the Pressure object data-output pin, pin B to the Pressure sdev data-input pin (Data).
18. Connect the Pressure object data-output pin, pin C to the Pressure rms data-input pin (Data).
19. Double-click separately on each of the four logging-alphanumeric title bars; remove the checkmarks from Initialization: Clear at PreRun and Clear at Activate.
20. Save this modified program; leave its name: LAB6-4. See Figure 6.12.

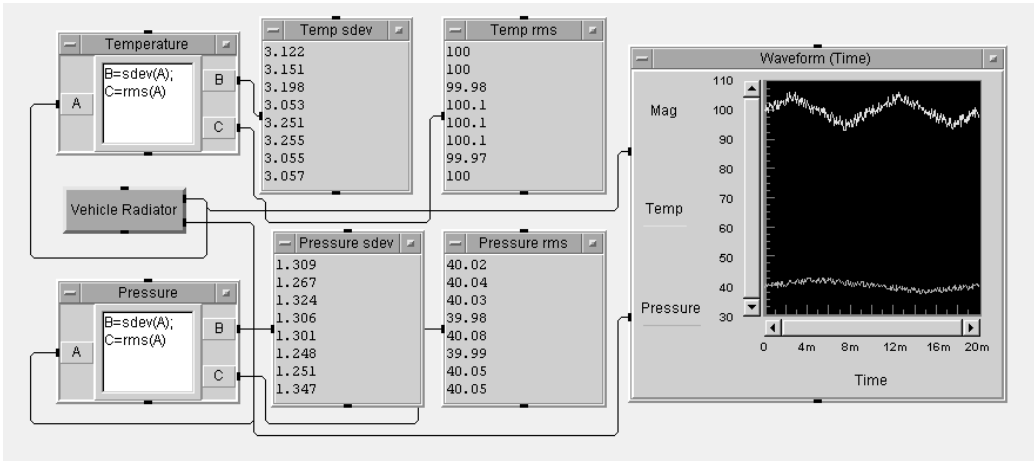

Figure 6.12. Log of vehicle radiator statistical data

◊ 21. Run this program several times as shown in Figure 6.12.
Note: You will later further modify LAB6-4 so it will send its data points to a spreadsheet.

Lesson 6 Summary

Lab 6.1 showed you how to use a Collector object and math expressions to create and display an array of data, and to use a Concatenator object to combine scalars and array elements into a single array.

Lab 6.2 showed you how to move test data to and from files. You learned to store and retrieve three common test result items: a test name, a time stamp, and a one-dimension array of real values.

Lab 6.3 showed you how to set up a function generator, graph its waveform time-domain output, and calculate the waveform mean, median, mode, variance, standard deviation, and rms values.

Lab 6.4 showed you how to simultaneously monitor Vehicle Radiator temperature and pressure data and display the results of calculating standard deviation and root-mean-square statistical analyses of these data points.

You are now ready to learn how to work with records.