Exercise: Timing, Structures, and Storing Data

Publish Date: Jul 02, 2008 | 8 Ratings | **4.00** out of 5

Overview

In this exercise the user learns the basics of using timing functions, shift registers, and case structures.

Table of Contents

Goal

Understand the basic behavior and operation of case structures, shift registers, and timing functions in the LabVIEW environment.

Description

This exercise consists of a series of tasks in which will walk you through how shift registers can be used to retain data between iterations of loops. The use of case structures to control the flow of a program will also be explored in detail. Timing functions will be explained and a simple implementation will be shown.

Timing VI

1. Launch LabVIEW and open a blank VI.

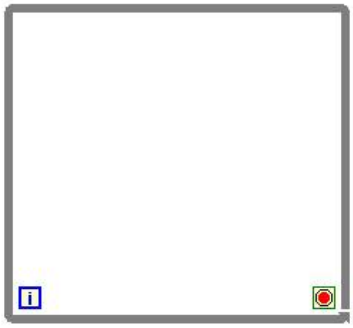
   Select File»New VI.

2. Open the block diagram of the VI.

   Select Window»Show Block Diagram.

3. Place a While Loop on the block diagram.

   Select **Programming»Structures»While Loop** from the functions palette.
   Place the **While Loop** on the block diagram by clicking and holding on the block diagram and then dragging the mouse to form a square shape.



4. Create a **Stop** control for the **While Loop**.

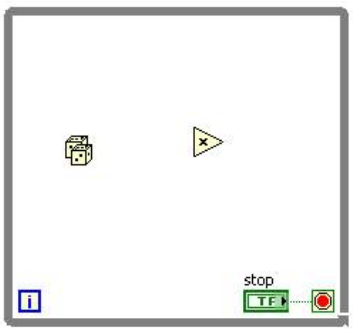
   Right click the input of the **Loop Condition** terminal of the While Loop and select **Create»Control**.

5. Place a **Random Number (0-1)** function on the block diagram.

   From the Functions palette select **Programming»Numeric»Random Number (0-1)** and place it inside the while loop.

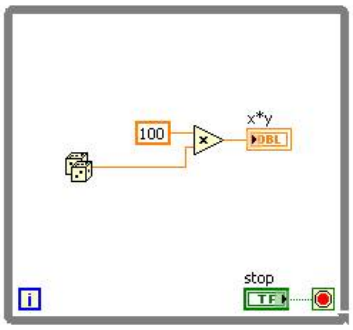
6. Place a **Multiply** function on the block diagram.

   From the Functions palette select **Programming»Numeric»Multiply** and place it inside the while loop.



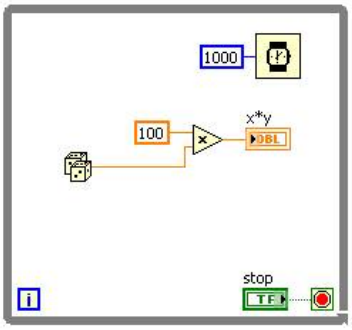
   Right click the first input of the **Multiply** function and select **Create»Constant**.
   Give this constant a value of **100**.
   Right-click the output of the multiply function and select **Create»Indicator**.

7. Wire the output of the **Random Number** function to the second input of the **Multiply** function.



8. Place a **Wait** function on the block diagram.

   Select **Programming»Timing»Wait** from the functions palette and place it on the block diagram inside of the while loop.
   Right click the **milliseconds to wait** input of the **Wait** function and select **Create»Constant**.
   Set the value of the constant to **1000**.



9. Switch to the front panel of the VI.

   Select Window»Show Front Panel.

10. Run the VI. Observe how a random number between zero and a hundred is generated every second.

11. Click the **STOP** button on the front panel.

12. Close the VI

Shift Registers

1. Launch LabVIEW and open a blank VI.

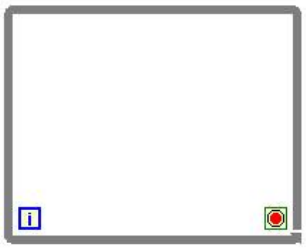
   Select File»New VI.

2. Open the block diagram of the VI.

   Select Window»Show Block Diagram.

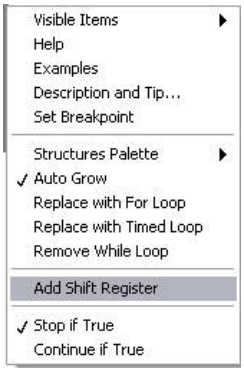
3. Place a While Loop on the block diagram.

   Select **Programming»Structures»While Loop** from the functions palette.
   Place the **While Loop** on the block diagram by clicking and holding on the block diagram and then dragging the mouse to form a square shape.



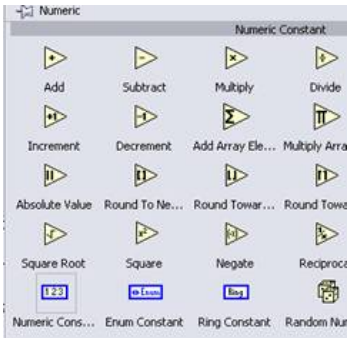
4. Create a shift register on the while loop.

   Right click on the left border of the While Loop and select **Add Shift Register**.



5. Place a numeric constant on the block diagram.

Select **Programming»Numeric»Numeric Constant** and place it on the block diagram inside the for loop.
Give the constant a value of **100.**
Connect constant to the input of the shift register.



6. Place an **Increment** function on the block diagram.

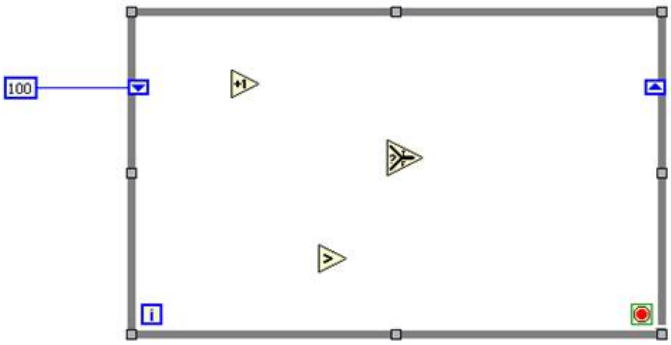
   From the Functions palette select **Programming»Numeric»Increment** and place it inside the while loop.

7. Place a **Greater?** function on the block diagram.

   From the Functions palette select **Programming»Comparison»Greater?** and place it inside the while loop.
   Right click the **y** input of the **Greater?** function and select **Create»Constant.**
   Give this constant a value of **9000**.

8. Place a **Select** function on the block diagram.

   From the Functions palette select **Programming»Comparison»Select** and place it inside the while loop.



   Right click the output of the **Select** function and select **Create»Indicator.**

9. Create a **String Constant**.

   Select **Programming»String»String Constant** from the Functions palette and place it inside the while loop.
   Give this constant a value of “Over 9000!”
   Wire this constant to the **t** input of the **Select** function.

10. Create a **String Constant**.

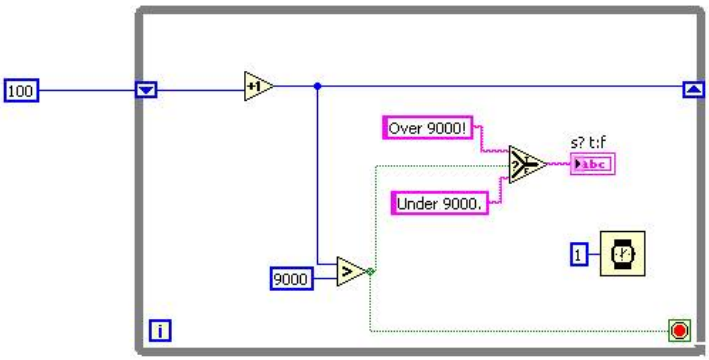
    Select **Programming»String»String Constant** from the Functions palette and place it inside the while loop.
    Give this constant a value of “Under 9000!”
    Wire this constant to the **f** input of the **Select** function.

11. Place a **Wait** function on the block diagram.

    Select **Programming»Timing»Wait** from the functions palette and place it on the block diagram inside of the while loop.
    Right click the **milliseconds to wait** input of the **Wait** function and select **Create»Constant**.
    Set the value of the constant to **1**.

12. Wire the block diagram as shown below.

Wire the output of the shift register on the left side of the while loop to the input of the **Increment** function
Wire the output of the **Increment** function to the input of the shift register on the right side of the loop.
Wire the output of the **Increment** function to the **x** input of the **Greater?** function.
Wire the output of the **Greater?** function to the **s** input of the **Select** function.
Wire the output of the **Greater?** function to the **Loop Condition** input of the while loop.



13. Switch to the front panel.
    Select Window»Show Front Panel.
14. Run the VI. Notice how the shift register always retains the value from the previous iteration of the loop. When the value reaches over 9000, a warning is displayed and the VI halts.
15. Close the VI.

Case Structures

Inputs and Outputs

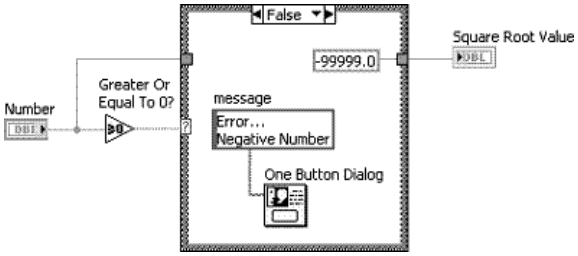
Type	Name	Properties
Input	Number	Double-precision, floating point; default value of 25
Output	Square Root Value	Double-precision, floating point

Implementation

1. Open a blank VI and build the front panel shown in the figure below.

1. Add a numeric control to the front panel window.
   Name the numeric control **Number**.
1. Add a numeric indicator to the front panel window.
   Rename the numeric indicator **Square Root Value**.

Build the block diagram shown in the figure below.



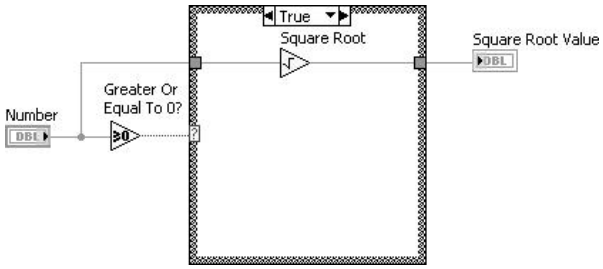
4. Determine whether **Number** is greater than or equal to zero, because you cannot calculate the square root of a negative number.
   Add the **Greater or Equal to 0?** function to the right of the Number control. This function returns True if Number is greater than or equal to 0.
   Wire Number to the input of the **Greater or Equal to 0?** function.
5. If Number is less than 0, display a dialog box that informs the user of the error.
   Add the **Case** structure to the block diagram.
   Click the decrement or increment button to select the **False** case.
   Add a numeric constant to the **False** case.
   Right-click the numeric constant and select **Representation»DBL**.
   Enter -**99999** in the numeric constant.
   Wire the numeric constant to the right edge of the **Case** structure.
   Wire the new tunnel to the **Square Root Value** indicator.
   Add the **One Button Dialog** function to the **False** case. This function displays a dialog box that contains a message you specify.
   Right-click the message input of the **One Button Dialog** function and select **Create»Constant** from the shortcut menu.
   Enter "Error...Negative Number" in the constant.
   Finish wiring the **False** case as shown in the above figure.

6. If Number is greater than or equal to 0, calculate the square root of the number.

Select the **True** case of the Case structure.

Place the **Square Root** function in the **True** case. This function returns the square root of Number.

Wire the function as shown in the figure below.



1. Save the VI as Square Root.vi.

Test

1. Display the front panel.

2. Enter a positive number in the Number control.

3. Run the VI.

4. Enter a negative number in the Number control.

Caution: Do not run this VI continuously. Under certain circumstances, continuously running this VI could result in an endless loop.

5. Run the VI.

If Number is positive, the VI executes the True case and returns the square root of Number. If Number is negative, the VI executes the False case, returns –99999, and displays a dialog box with the message Error...Negative Number.

6. Close the VI.

End of Exercise
