

LabVIEW Lesson 4 – Arrays


Lesson Overview



- What are arrays?
 - How to create a one-dimensional array.
 - How to create a multi-dimensional array.
 - How to use data/values from previous loops.
-

I. What is an array?

- a. An array can either resemble a vector or a matrix. As does a vector and a matrix, an array groups similar pieces of data.
- b. An array consists of two different components, the elements (pieces of data) and the dimension (the size of the array).
- c. Arrays may contain numeric, Boolean, path, string, waveform, and cluster data types. They may be used as an indicator (output) or a control (input).
- d. Arrays are advantageous to use when you are dealing with similar pieces of data and/or repetitive computations, which means that they are ideal for storing waveform data and data from loop iterations with each array element being the loop iteration value.
- e. The elements in the array are ordered or assigned a certain index. Arrays begin indexing the elements at zero, so the first element will have an index of zero instead of one. ***(This is very important to remember when accessing data from an array.)***

II. Creating a One-Dimension Array.


- a. Depending on the type of array you want to create, different methods in creating the array must be taken.
 - i. For an input or output array, proceed as follows in the front panel:
 1. **Controls Palette** → **All-Controls** → **Array & Cluster** → **Array** 
 2. The array you create will have a control on the top-left side that will allow you to navigate through the array. The number in the control box will indicate the index of the element shown in the leftmost cell.
 3. Notice when you first put the array on the front panel that it is empty. You can determine your array type by inserting either a control or indicator inside the array. For example, for a numerical indicator array:
 - **Controls Palette** → **Num Inds** → **Num Ind** → **Place inside Array** (*Inside array box will have blinking border*).

4. You can control the viewing size of your array by dragging one of the corners where there is a little blue box in either the vertical or horizontal direction. Thus, you can either make your one-dimensional array represent a column vector or a row vector, which will be important when performing mathematical operations as arrays follow the same mathematical rules as do vectors and matrices. You may also notice that the size of the array increases or decreases in increments of the current cell size.
- ii. For a constant array, proceed as follows in the block diagram:
 1. **Functions Palette** → **All-Functions** → **Array** → **Array Constant** 
 2. This array will look similar to the previous one as it will be empty and it will have an array control on the left side.
 3. To make the array a numerical constant array,
 - **Right-Click** on the box → **Arith/Compare** → **Numeric** → **Num Const**  → **Drag** into the empty array.




III. Creating a **Multi-Dimensional Array**.

- a. Creating a multi-dimensional array (*in mathematical terms, a matrix*) follows the same procedures as for creating a one-dimensional array.
 - i. To add another dimension to the one-dimensional array, **Right-Click** on the **Array Control** (*upper left corner*) → **Add Dimension** (*another array control will appear below the original and you may now drag the viewing window in both directions – vertically and horizontally*)
 - ii. The top array control will control the row index and the bottom array control will control the column index.

IV. Storing values in an **Array** from a **For Loop**.

- a. An array can be used to store a for loop's iterative outputs as different entries in the array.
- b. **Example 4.1 – Part A:** Storing For Loop Values in an Array.
 - i. Starting with the **Front Panel**.
 1. Create an Numerical Indicator Array
 - **Controls Palette** → **All-Controls** → **Array & Cluster** → **Array** 
 2. Insert a Numerical Indicator inside the array
 - **Controls Palette** → **Num Inds** → **Num Ind** → **Place inside Array** (*Inside array box will have blinking border*).

ii. Switch to the **Block Diagram**.

1. Create a **For Loop** 
 - **Functions Palette** → **All Functions** → **Structures** → **For Loop** → **Left-Click** and **Hold** on block diagram → **Drag** to create a box.
2. Insert a **Random Number Generator**  inside **For Loop**
 - **Functions Palette** → **All Functions** → **Arith/Compare** → **Numeric** → **Random Num**
3. Wire **Random Number Generator** to the **Indicator Array**
4. Create a **Numerical Constant** of **10** and wire to the **Iteration Terminal** .
5. Your **Block Diagram** should look similar to Figure 4.1.

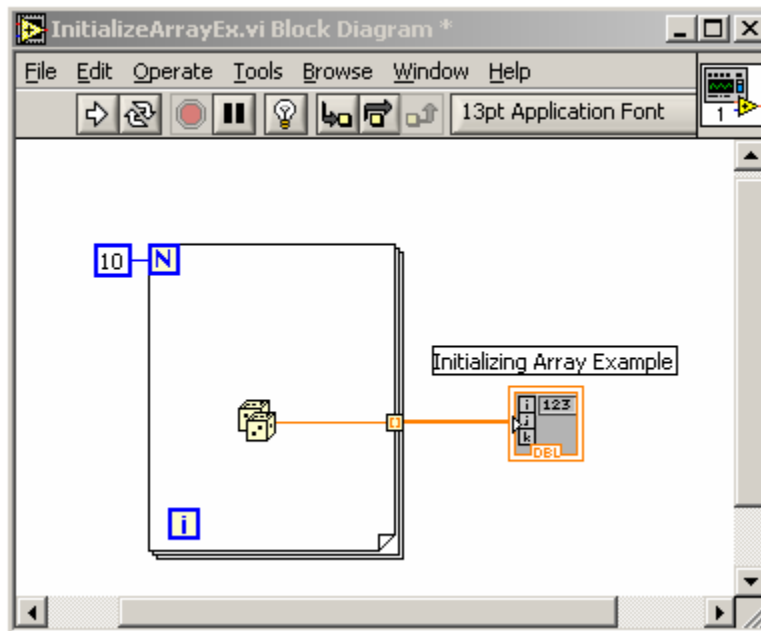


Figure 4.1: Block Diagram for Storing For Loop Values in an Array

iii. Switch to the **FRONT PANEL**

1. Run the VI. What is the highest index that has a value in it? Why?
Important: Remember the counting system.
2. Your **Front Panel** with results should look similar to Figures 4.2a and b.

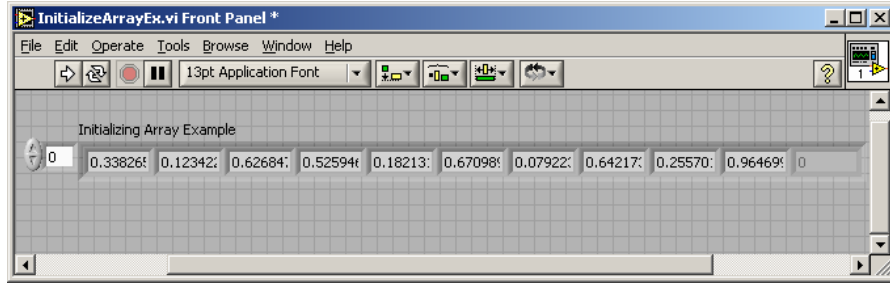


Figure 4.2a: Front Panel for Storing For Loop Values in an Array

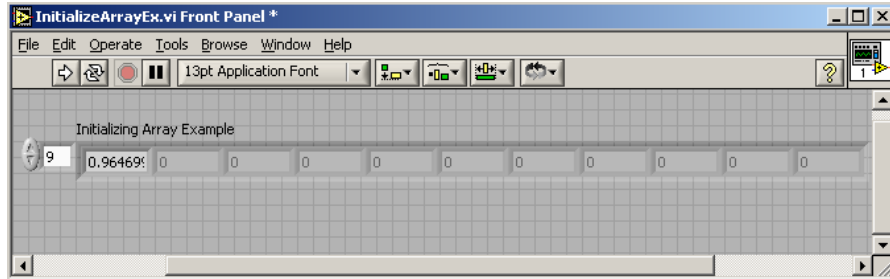




Figure 4.2b: Front Panel – Cell Index 9 Corresponds to 10th Value

c. **Example 4.1 – Part B: Regulating the Number of Loop Iterations Performed**

i. Switch to the **Block Diagram** you created in **Part A**.

1. Create a **Numerical Constant Array** below the **For Loop**
 - **Functions Palette** → **All-Functions** → **Array** → **Array Constant** 
 - **Right-Click** on the box → **Arith/Compare** → **Numeric** → **Num Const**  → **Drag** into the empty array
2. Lengthen the array by dragging the right border until there are 10 cells visible. Then, initialize the first ten values from 1 to 10 by clicking on each cell and typing in the appropriate value. (*You will notice that when the cell is initialized it turns from gray to white.*)
3. Wire the **Array Constant** to the **Left Border** of the **For Loop**.
4. Change the **Loop Iteration Constant** from **10** to **5**.
5. Your **Block Diagram** should now look similar to Figure 4.3.

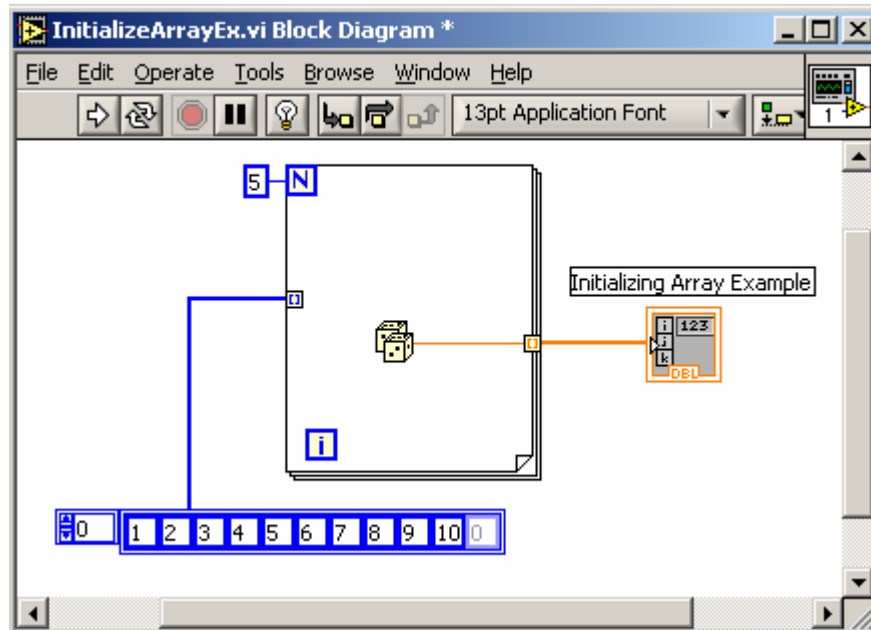


Figure 4.3: Block Diagram for Part B

- ii. Switch to the **Front Panel** and **Run** the VI.
 1. How many values are in the output array? Why aren't there 10 values?
 2. Now change the **Loop Iteration Constant** to **15** and **Run** the VI again. How many values are in the output array now? What determines the size of the output array?
 3. **Note:** To reset your array to its default values:
 - **Left-Click** on the border of the array so that everything is selected → **Right-Click** anywhere on the array → **Data Operations** → **Empty Array**
 4. **Note:** To reset a particular value/cell in your array to its default value:
 - **Right-Click** on the cell → **Data Operations** → **Reinitialize to Default Value**
- d. **Example 4.2:** Creating a 2-D Array (Matrix) from For Loops
 - i. Switch back to the **Block Diagram** of the VI created in **Example 4.1 – Part B**.
 1. Delete the **Array Constant** and the **Wire** connecting the **Random Number Generator** to the **Output Array**.
 2. Create another **For Loop** around the **Current For Loop**, but excluding the **Output Array**.

3. Create a **Numerical Constant** wired to the **Outside For Loop's Iteration Terminal** of **5** and change the **Inside For Loop's Constant** back to **10**.
4. Wire the **Random Number Generator** to the **Output Array**. (The wire will be broken until will add another dimension to our array on the Front Panel.)
5. Your **Block Diagram** should look similar to Figure 4.4.

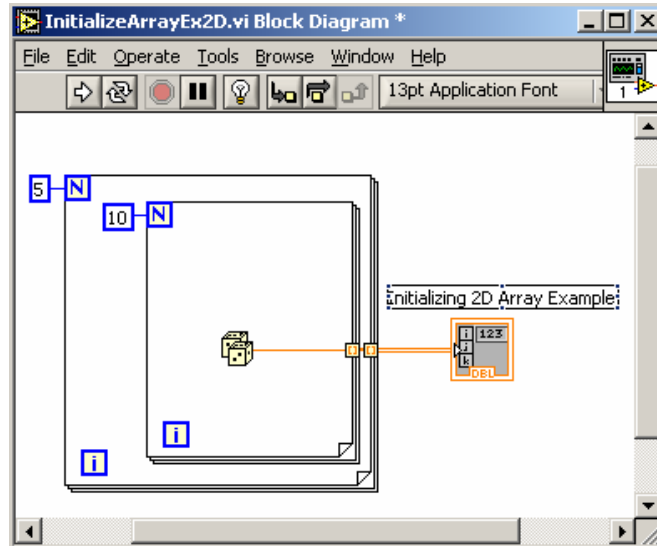


Figure 4.4: Block Diagram for Creating a 2-D Array from For Loops

- ii. Switch to the **Front Panel**
 1. Add another dimension to the **Output Array**
 - **Right-Click** on the **Array Control** → **Add Dimension**
 2. Run your VI and your **Front Panel** should look similar to Figure 4.5. Which loop controls the rows and which controls the columns?

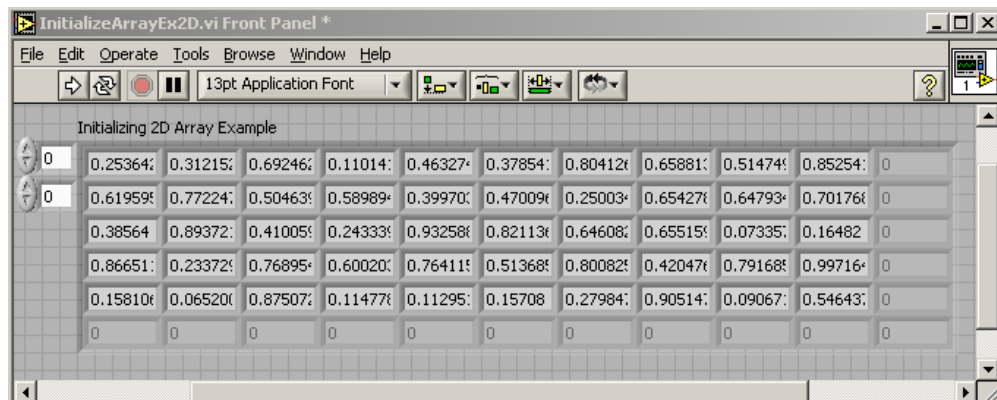


Figure 4.5: Front Panel for Creating a 2-D Array with 10 Rows and 5 Columns

e. **Example 4.3: Exploring Available Array Functions**

- i. This VI will incorporate many of the available array functions all in one program. Many others exist that can be explored using the help tool, which will explain what each function does and what parameters are required.
- ii. Beginning with the **Front Panel** of a **New Blank VI**.
 1. Create **5 Arrays**
 - a. The **First Array** will be a **Numerical Control Array**, the **Second, Third** and **Fifth** will be **Numerical Indicator Arrays**, and the **Forth** will be a **2-D Numerical Indicator Array**.
 - b. Name the **Arrays** as follows: “**Array**”, “**Subarray**”, “**Initialized Array**”, “**Appended Array**”, and “**Concatenated Array**”
 2. Create **2 Numerical Indicators**
 - a. Name the **Numerical Indicators** as follows: “**Array Size**” and “**Indexed Element**”
 - b. Arrange your **Front Panel** similar to the one shown in Figure 4.6.

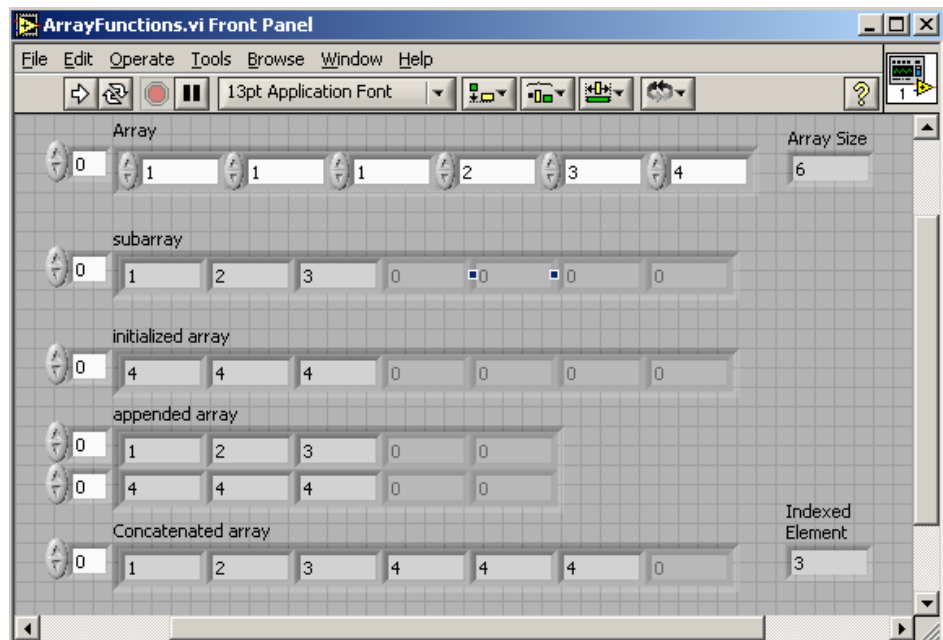





Figure 4.6: Front Panel for VI with Several Array Functions

- iii. Switch to the **Block Diagram**
 1. Using the **Array Size Function** 
 - **Functions Palette** → **All-Functions** → **Arrays and Clusters** → **Array Size**

- Wire **“Array” Terminal** → **Array Size Function** → **“Array Size” Numerical Indicator**
2. Using the **Array Subset Function** 
- **Functions Palette** → **All-Functions** → **Arrays and Clusters** → **Array Subset**
 - Wire **“Array” Terminal** → **Array Subset Function** → **“Subarray” Array Indicator**
3. Using the **Initialize Array Function** 
- **Functions Palette** → **All-Functions** → **Arrays and Clusters** → **Initialize Array**
 - Wire **Numerical Constant of 4** → **Element Terminal of Initialize Array Function**
 - Wire **Numerical Constant of 3** → **Length Terminal of Initialize Array Function**
 - Wire **Initialize Array Function** → **“Initialized Array” Array Indicator**
4. Using the **Build Array Function** 
- a. The default for this function is to **Append** to the original array.
 - **Functions Palette** → **All-Functions** → **Arrays and Clusters** → **Build Array**
 - Wire **“Array” Terminal** → **Top Terminal of Append/Build Array Function**
 - Branch a wire from the **Output** of the **Initialize Array Function** → **Bottom Terminal of Append/Build Array Function**
 - Wire **Output of Append/Build Array Function** → **“Appended Array” Array Indicator**
 - b. The **Build Array Function** can also be used to **Concatenate** to the original array.
 - **Functions Palette** → **All-Functions** → **Arrays and Clusters** → **Build Array** → **Right-Click on Build Array Function** → **Concatenate**
 - Wire **“Array” Terminal** → **Top Terminal of Concatenate/Build Array Function**

- Branch a wire from the **Output** of the **Initialize Array Function** → **Bottom Terminal** of **Concatenate/Build Array Function**
- Wire **Output** of **Concatenate/Build Array Function** → **“Concatenated Array” Array Indicator**

5. Using the **Index Array Function** 
- **Functions Palette** → **All-Functions** → **Arrays and Clusters** → **Index Array**

- **Branch** a wire from the **Output** of the **Concatenated/Build Array Function** → **Top Terminal** of **Index Array Function**
- Wire **Numerical Constant** of **2** → **Bottom Terminal** of **Index Array Function**
- Wire **Output** of **Index Function** to **“Indexed Element” Numerical Indicator Terminal**

6. Your **Block Diagram** should look similar to the one shown in Figure 4.7.

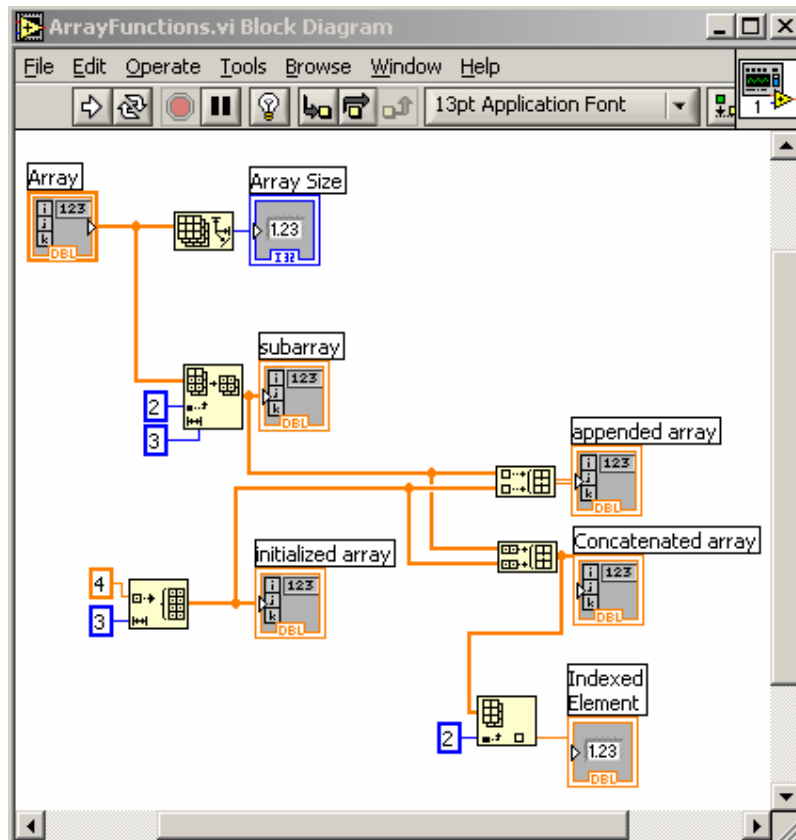


Figure 4.7: Block Diagram for VI with Several Array Functions

- iv. Switch to the **Front Panel**
 1. Save as "**ArrayFunctions.VI**"
 2. Initialize the values in the "**Array**" **Control Array** to **1, 1, 1, 2, 3,** and **4** and then, **Run**.
 3. Observe the outputs and investigate how the functions work.