How to Pass Data Into and Out of a Loop Tutorial (Tunneling and Auto Indexing)

Properties of Loop Structures

When passing data values into or out of a loop structure, we must create "Tunnels" where the data values enter and exit the structure. Tunnels are indicated on structure borders by a solid square or square with brackets (depending on Auto Indexing) as indicated in Figure 1.

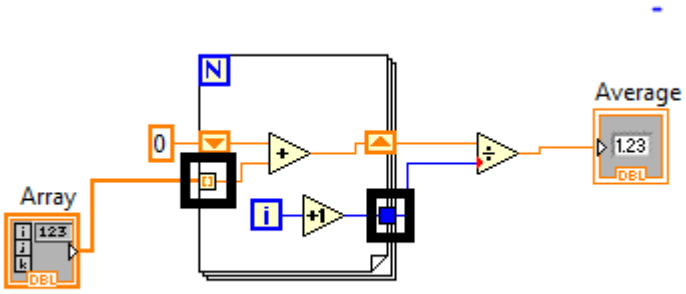

Figure 1

When data like arrays enter tunnels, the values can either be passed with Auto Indexing or without Auto Indexing (note that Auto Indexing is a property only of loop structures). In Figure 1, the left tunnel has square bracket indicating it has Auto Indexing Enabled while the right tunnel is solid indicating Auto Indexing is disabled.

Passing Data Into a Loop

If Auto Indexing is enabled, values from array structures will be passed one at a time. If Auto Indexing is disabled, then the entire data structure or value will be passed at once.

So for example, for the array structure, if Auto Indexing is enabled, on each new iteration the tunnel would pass the next value IN the array (notice this is may NOT be an array data type). But if Auto Indexing was disabled, the tunnel would pass the entire array structure.

For example, say we wanted to pass the array [1 2 3 4 5] into a loop structure. The general flow of data would be something like this:

Original Input to Structure	Iteration	Value passed into Structure WITH Auto Indexing	Value passed into Structure WITHOUT Auto Indexing
[1 2 3 4 5]	1	1	[1 2 3 4 5]
[1 2 3 4 5]	2	2	[1 2 3 4 5]
[1 2 3 4 5]	3	3	[1 2 3 4 5]
[1 2 3 4 5]	4	4	[1 2 3 4 5]
[1 2 3 4 5]	5	5	[1 2 3 4 5]

Passing Data Out of a Loop

To pass data out of a loop structure, the behavior of the tunnel and effect of the Auto Indexing is the same as when passing data into the loop. If we enable Auto Indexing, the values that the loop generates will accumulate into an array, which will then be passed from the structure as an array of values.

If we simply want to use each individual value in each iteration as they are produced, we need to disable Auto Indexing.

Figure 2 gives an example of how Auto Indexing can be used to produce an array from 0 to 5.

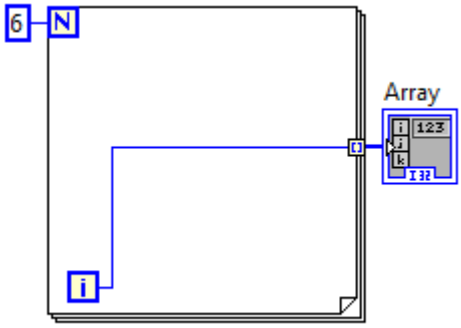

Figure 2

In this example, we notice that the loop terminates after i = 6, so the loop will only execute from 0 to 5. In addition, we notice that since Auto Indexing is enabled on the tunnel, the values generated by the count i are accumulated and passed in a matrix.

Tunnels and Auto Indexing with Higher Dimensional Arrays

For the two dimensional case, on each iteration, the Auto Index Enabled tunnel will provide a one dimensional sub-array as shown in the table below.

Original Array	Iteration	Value Passed WITH Auto Indexing	Value Passed WITHOUT Auto Indexing
$\begin{bmatrix}1 & 2 & 3\\4 & 5 & 6\\7 & 8 & 9\end{bmatrix}$	1	$[1 \quad 2 \quad 3]$	$\begin{bmatrix}1 & 2 & 3\\4 & 5 & 6\\7 & 8 & 9\end{bmatrix}$
$\begin{bmatrix}1 & 2 & 3\\4 & 5 & 6\\7 & 8 & 9\end{bmatrix}$	2	$[4 \quad 5 \quad 6]$	$\begin{bmatrix}1 & 2 & 3\\4 & 5 & 6\\7 & 8 & 9\end{bmatrix}$
$\begin{bmatrix}1 & 2 & 3\\4 & 5 & 6\\7 & 8 & 9\end{bmatrix}$	3	$[7 \quad 8 \quad 9]$	$\begin{bmatrix}1 & 2 & 3\\4 & 5 & 6\\7 & 8 & 9\end{bmatrix}$

For example, consider the following example in Figure 3 that performs an element-wise increment of a two dimensional array. We will notice that the first Auto Indexing tunnel passes the matrix as a

row vector or one dimensional array and the second Auto Indexing tunnel on the nested For Loop passes the individual elements.

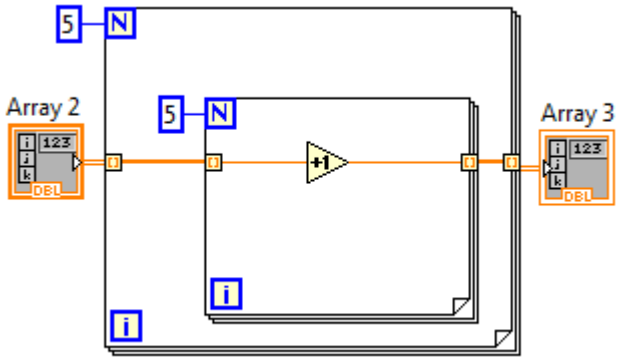

Figure 3

Note that the same procedure occurs when rebuilding the array. The individual values for each row vector are accumulated at the end of the nested For Loop and these row vectors are accumulated at the end of the main For Loop and passed as a 2 dimensional array.

Generally with higher dimensional arrays, the value or array passed on the ith iteration through the tunnel corresponds to the array containing all entries under the ith index of the first array dimension.

Enabling and Disabling Auto Indexing

To enable or disable Auto Indexing, right-click on the tunnel that you want to change and select "Enable Indexing" or "Disable Indexing" as shown in Figures 4 and 5.

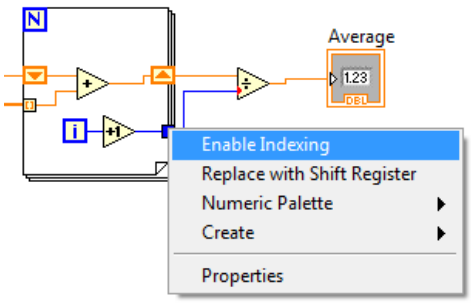

Figure 4

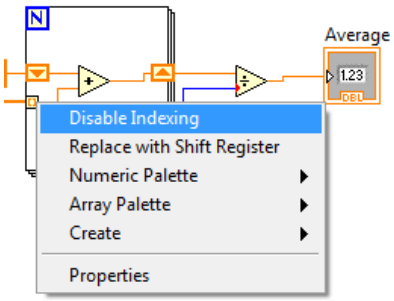

Figure 5

You will also notice that if you choose Auto Indexing when you should not be using Auto Indexing, LabView will indicate a data type mismatch at the divide block shown in Figure 6, since Auto Indexing when tunneling out of the loop structure will generate an array instead of a numeric double. Also, you will notice that wires that carry a set of data like an array will appear thicker than wires that carry individual pieces of data.

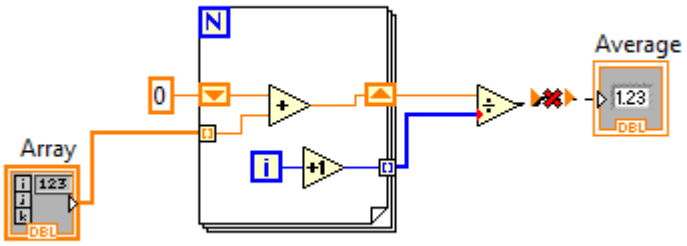

Figure 6

The error in Figure 5 is that the divide block is taking in a single numeric Double and an array of numeric I32s. Even before coercion, this results in an incompatible data structure for the Numeric Indicator because you cannot divide a single numeric double by an array and feed the resulting data type to the input of a numeric indicator. To solve this problem we simply need to turn off the Auto Indexing for the tunnel on the right side of the structure.