LabVIEW for Engineers

Ronald W. Larsen

E SOURCE

About ESource

Your Introductory Engineering Course—Your Way

Welcome to ESource, Prentice Hall's Introductory Engineering series. Over 25 modules in this series cover topics frequently taught in introductory engineering courses. Topics include an introduction to the various fields of engineering, design and problem solving skills, communication and teamwork, computer applications such as MATLAB and Mathcad, an introduction to engineering graphics and visualization, and more. All the books in the ESource series are written by educators specifically for freshman/first-year students. A complete list of all of our ESource authors and their respective backgrounds is available at **www.prenhall.com/esource**.

Customize

Every book in this series is available separately or packaged together at a discount to students—or, using our electronic customization program—instructors can create their own customized ESource textbook, selecting any combination and sequence of chapters from any of the books in the series. Plus, instructors can add their own material to the book as well (syllabi, course notes, etc.) For more information, visit **www.prenhall.com/esource**.

ESource Access

Instructors who choose to bundle two or more texts from the ESource series or use a customized ESource textbook can provide their students with an on-line library of selected ESource content—ESource Access. Student access codes are valid for six months after initial registration. Contact your local Prentice Hall sales representative for more information.

Classroom and Instructor Resources

A wealth of resources are available to adopting instructors, including PowerPoints and Instructors Manuals. Visit **www.prenhall.com/esource** for more information.

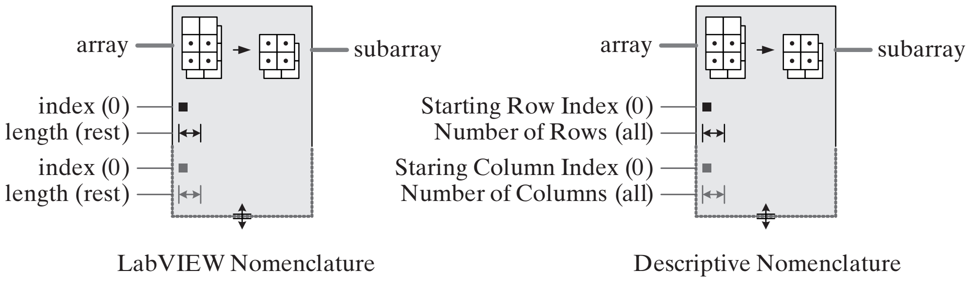


Figure 4.5 Array Subset function description.

The four terminals on the lower left side of the icon (see Figure 4.5) are used to inform LabVIEW which portion of the original array to use for the subarray.

- **Starting Row Index** (0)—the index of the first row in the original array that should be included in the subarray. The default value is (0), which is the top row in the original matrix.
- **Number of Rows** (all)—the number of rows from the original array to include in the subarray. The default is (all) rows (i.e., all rows after the starting row).
- **Starting Column Index** (0)—the index of the first column in the original array that should be included in the subarray. The default value is (0), which is the left column in the original matrix.
- **Number of Columns** (all)—the number of columns from the original array to include in the subarray. The default is (all) columns (i.e., all columns after the starting column).

Notice that array indexing starts at 0 (not 1) in LabVIEW. The top element of array X is called X[0].

Figure 4.6 shows the block diagram of a VI designed to allow any subset of the data set shown in Table 4.1 to be selected.

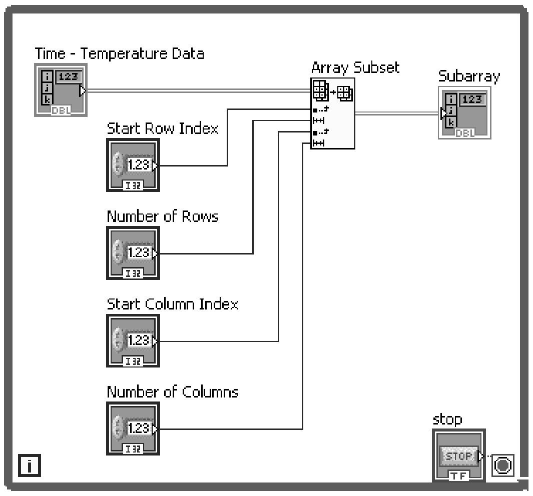


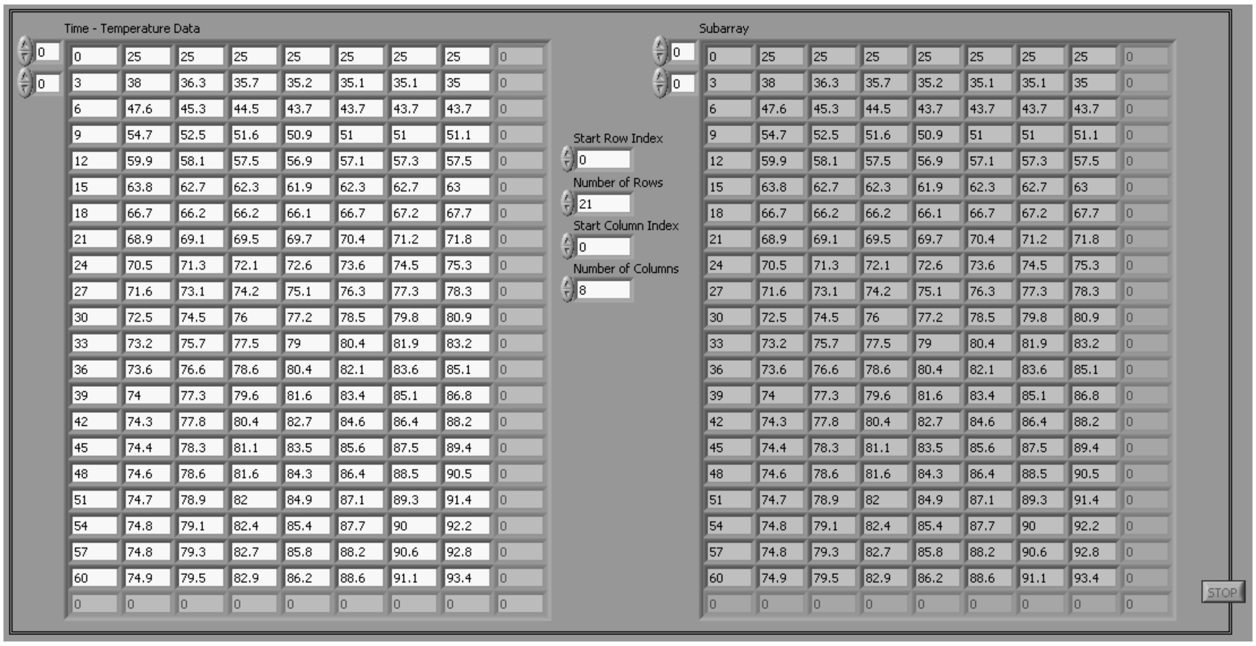
Figure 4.6 Block diagram of a VI allowing a subarray to be selected at run time.

Functions Palette / Programming Group / Array Group / Array Subset

Note: When only a single column or row is needed, the Index Array function should be used. The Index Array function returns a 1D array.

Functions Palette / Programming Group / Array Group / Index Array

The front panel is shown in Figure 4.7. Initially the entire original array has been selected.



Time - Temperature Data

0
0

0	25	25	25	25	25	25	25	0
3	38	36.3	35.7	35.2	35.1	35.1	35	0
6	47.6	45.3	44.5	43.7	43.7	43.7	43.7	0
9	54.7	52.5	51.6	50.9	51	51	51.1	0
12	59.9	58.1	57.5	56.9	57.1	57.3	57.5	0
15	63.8	62.7	62.3	61.9	62.3	62.7	63	0
18	66.7	66.2	66.2	66.1	66.7	67.2	67.7	0
21	68.9	69.1	69.5	69.7	70.4	71.2	71.8	0
24	70.5	71.3	72.1	72.6	73.6	74.5	75.3	0
27	71.6	73.1	74.2	75.1	76.3	77.3	78.3	0
30	72.5	74.5	76	77.2	78.5	79.8	80.9	0
33	73.2	75.7	77.5	79	80.4	81.9	83.2	0
36	73.6	76.6	78.6	80.4	82.1	83.6	85.1	0
39	74	77.3	79.6	81.6	83.4	85.1	86.8	0
42	74.3	77.8	80.4	82.7	84.6	86.4	88.2	0
45	74.4	78.3	81.1	83.5	85.6	87.5	89.4	0
48	74.6	78.6	81.6	84.3	86.4	88.5	90.5	0
51	74.7	78.9	82	84.9	87.1	89.3	91.4	0
54	74.8	79.1	82.4	85.4	87.7	90	92.2	0
57	74.8	79.3	82.7	85.8	88.2	90.6	92.8	0
60	74.9	79.5	82.9	86.2	88.6	91.1	93.4	0
0	0	0	0	0	0	0	0	0

Start Row Index: 0
Number of Rows: 21
Start Column Index: 0
Number of Columns: 8

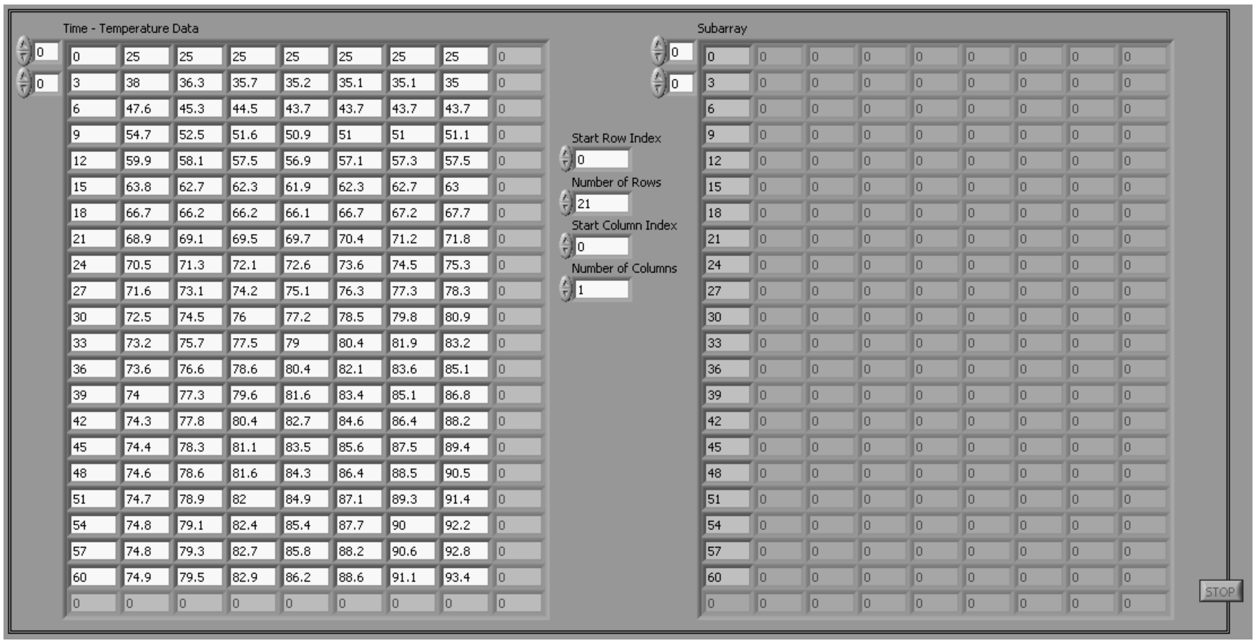
Subarray

0
0

0	25	25	25	25	25	25	25	0
3	38	36.3	35.7	35.2	35.1	35.1	35	0
6	47.6	45.3	44.5	43.7	43.7	43.7	43.7	0
9	54.7	52.5	51.6	50.9	51	51	51.1	0
12	59.9	58.1	57.5	56.9	57.1	57.3	57.5	0
15	63.8	62.7	62.3	61.9	62.3	62.7	63	0
18	66.7	66.2	66.2	66.1	66.7	67.2	67.7	0
21	68.9	69.1	69.5	69.7	70.4	71.2	71.8	0
24	70.5	71.3	72.1	72.6	73.6	74.5	75.3	0
27	71.6	73.1	74.2	75.1	76.3	77.3	78.3	0
30	72.5	74.5	76	77.2	78.5	79.8	80.9	0
33	73.2	75.7	77.5	79	80.4	81.9	83.2	0
36	73.6	76.6	78.6	80.4	82.1	83.6	85.1	0
39	74	77.3	79.6	81.6	83.4	85.1	86.8	0
42	74.3	77.8	80.4	82.7	84.6	86.4	88.2	0
45	74.4	78.3	81.1	83.5	85.6	87.5	89.4	0
48	74.6	78.6	81.6	84.3	86.4	88.5	90.5	0
51	74.7	78.9	82	84.9	87.1	89.3	91.4	0
54	74.8	79.1	82.4	85.4	87.7	90	92.2	0
57	74.8	79.3	82.7	85.8	88.2	90.6	92.8	0
60	74.9	79.5	82.9	86.2	88.6	91.1	93.4	0
0	0	0	0	0	0	0	0	0

Figure 4.7
The entire original array has been selected as the subarray.

By changing the **Number of Columns** from 8 to 1, we can select only the time values for the subarray, as shown in Figure 4.8.



Time - Temperature Data

0
0

0	25	25	25	25	25	25	25	0
3	38	36.3	35.7	35.2	35.1	35.1	35	0
6	47.6	45.3	44.5	43.7	43.7	43.7	43.7	0
9	54.7	52.5	51.6	50.9	51	51	51.1	0
12	59.9	58.1	57.5	56.9	57.1	57.3	57.5	0
15	63.8	62.7	62.3	61.9	62.3	62.7	63	0
18	66.7	66.2	66.2	66.1	66.7	67.2	67.7	0
21	68.9	69.1	69.5	69.7	70.4	71.2	71.8	0
24	70.5	71.3	72.1	72.6	73.6	74.5	75.3	0
27	71.6	73.1	74.2	75.1	76.3	77.3	78.3	0
30	72.5	74.5	76	77.2	78.5	79.8	80.9	0
33	73.2	75.7	77.5	79	80.4	81.9	83.2	0
36	73.6	76.6	78.6	80.4	82.1	83.6	85.1	0
39	74	77.3	79.6	81.6	83.4	85.1	86.8	0
42	74.3	77.8	80.4	82.7	84.6	86.4	88.2	0
45	74.4	78.3	81.1	83.5	85.6	87.5	89.4	0
48	74.6	78.6	81.6	84.3	86.4	88.5	90.5	0
51	74.7	78.9	82	84.9	87.1	89.3	91.4	0
54	74.8	79.1	82.4	85.4	87.7	90	92.2	0
57	74.8	79.3	82.7	85.8	88.2	90.6	92.8	0
60	74.9	79.5	82.9	86.2	88.6	91.1	93.4	0
0	0	0	0	0	0	0	0	0

Start Row Index: 0
Number of Rows: 21
Start Column Index: 0
Number of Columns: 1

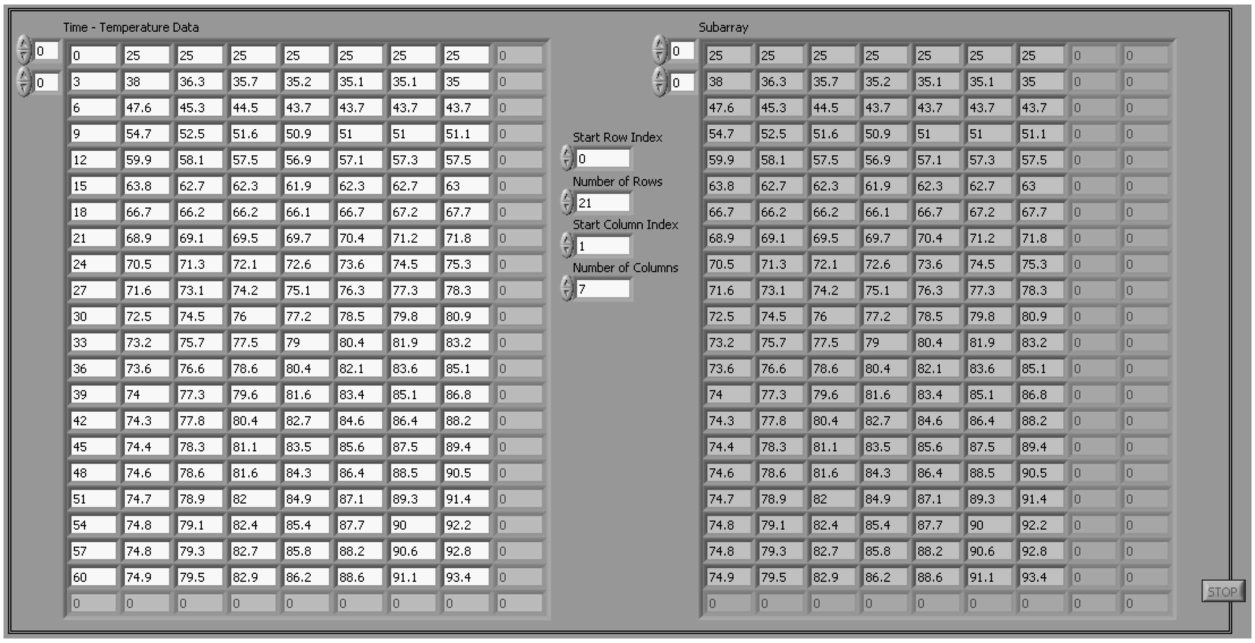
Subarray

0
0

0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0
27	0	0	0	0	0	0	0	0
30	0	0	0	0	0	0	0	0
33	0	0	0	0	0	0	0	0
36	0	0	0	0	0	0	0	0
39	0	0	0	0	0	0	0	0
42	0	0	0	0	0	0	0	0
45	0	0	0	0	0	0	0	0
48	0	0	0	0	0	0	0	0
51	0	0	0	0	0	0	0	0
54	0	0	0	0	0	0	0	0
57	0	0	0	0	0	0	0	0
60	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

Figure 4.8
Only the time values (left column) have been selected for the subarray.

By setting the **Start Column Index** to 1, and the **Number of Columns** to 7, we can select all of the temperature values, as shown in Figure 4.9.



Time - Temperature Data

Index: 0, 0

0	25	25	25	25	25	25	25	0
3	38	36.3	35.7	35.2	35.1	35.1	35	0
6	47.6	45.3	44.5	43.7	43.7	43.7	43.7	0
9	54.7	52.5	51.6	50.9	51	51	51.1	0
12	59.9	58.1	57.5	56.9	57.1	57.3	57.5	0
15	63.8	62.7	62.3	61.9	62.3	62.7	63	0
18	66.7	66.2	66.2	66.1	66.7	67.2	67.7	0
21	68.9	69.1	69.5	69.7	70.4	71.2	71.8	0
24	70.5	71.3	72.1	72.6	73.6	74.5	75.3	0
27	71.6	73.1	74.2	75.1	76.3	77.3	78.3	0
30	72.5	74.5	76	77.2	78.5	79.8	80.9	0
33	73.2	75.7	77.5	79	80.4	81.9	83.2	0
36	73.6	76.6	78.6	80.4	82.1	83.6	85.1	0
39	74	77.3	79.6	81.6	83.4	85.1	86.8	0
42	74.3	77.8	80.4	82.7	84.6	86.4	88.2	0
45	74.4	78.3	81.1	83.5	85.6	87.5	89.4	0
48	74.6	78.6	81.6	84.3	86.4	88.5	90.5	0
51	74.7	78.9	82	84.9	87.1	89.3	91.4	0
54	74.8	79.1	82.4	85.4	87.7	90	92.2	0
57	74.8	79.3	82.7	85.8	88.2	90.6	92.8	0
60	74.9	79.5	82.9	86.2	88.6	91.1	93.4	0
0	0	0	0	0	0	0	0	0

Start Row Index: 0
Number of Rows: 21
Start Column Index: 1
Number of Columns: 7

Subarray

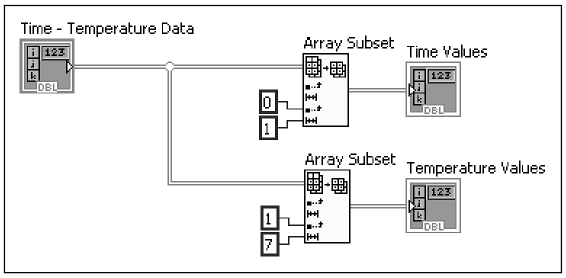
Index: 0, 0

25	25	25	25	25	25	25	0	0
38	36.3	35.7	35.2	35.1	35.1	35	0	0
47.6	45.3	44.5	43.7	43.7	43.7	43.7	0	0
54.7	52.5	51.6	50.9	51	51	51.1	0	0
59.9	58.1	57.5	56.9	57.1	57.3	57.5	0	0
63.8	62.7	62.3	61.9	62.3	62.7	63	0	0
66.7	66.2	66.2	66.1	66.7	67.2	67.7	0	0
68.9	69.1	69.5	69.7	70.4	71.2	71.8	0	0
70.5	71.3	72.1	72.6	73.6	74.5	75.3	0	0
71.6	73.1	74.2	75.1	76.3	77.3	78.3	0	0
72.5	74.5	76	77.2	78.5	79.8	80.9	0	0
73.2	75.7	77.5	79	80.4	81.9	83.2	0	0
73.6	76.6	78.6	80.4	82.1	83.6	85.1	0	0
74	77.3	79.6	81.6	83.4	85.1	86.8	0	0
74.3	77.8	80.4	82.7	84.6	86.4	88.2	0	0
74.4	78.3	81.1	83.5	85.6	87.5	89.4	0	0
74.6	78.6	81.6	84.3	86.4	88.5	90.5	0	0
74.7	78.9	82	84.9	87.1	89.3	91.4	0	0
74.8	79.1	82.4	85.4	87.7	90	92.2	0	0
74.8	79.3	82.7	85.8	88.2	90.6	92.8	0	0
74.9	79.5	82.9	86.2	88.6	91.1	93.4	0	0
0	0	0	0	0	0	0	0	0

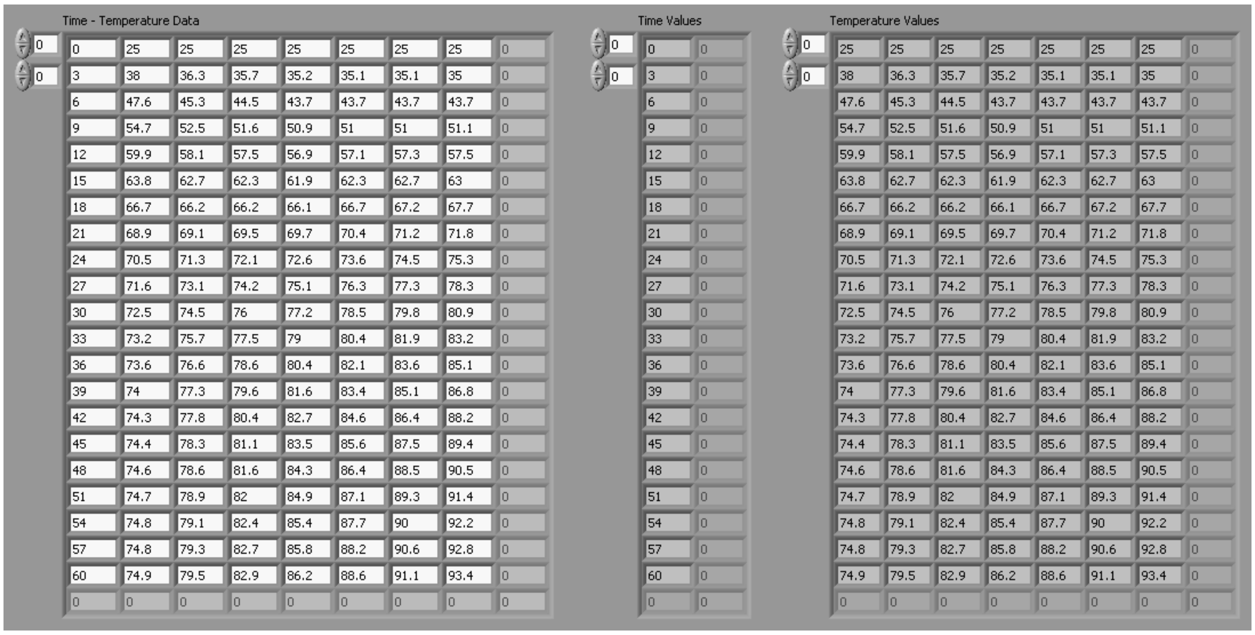
Figure 4.9
Selecting only the temperature values for the subarray.

In practice you would rarely use controls to select the subarray. The next VI uses constants on the block diagram to create two subarrays: one containing the time values and one containing the temperature values. The block diagram is shown in Figure 4.10 and the front panel in Figure 4.11.

Figure 4.10
Block diagram for selecting Time and Temperature subarrays from original array.



Notice in Figure 4.10 that constants (0, 1 and 1, 7) were used to indicate the desired columns for each subarray, but no constants were sent into the Array Subset functions for the row terminals. Because we wanted all rows, we accepted the default values for Starting Row Index and Number of Rows by leaving them unwired.



Time - Temperature Data

0
0

0	25	25	25	25	25	25	25	0
3	38	36.3	35.7	35.2	35.1	35.1	35	0
6	47.6	45.3	44.5	43.7	43.7	43.7	43.7	0
9	54.7	52.5	51.6	50.9	51	51	51.1	0
12	59.9	58.1	57.5	56.9	57.1	57.3	57.5	0
15	63.8	62.7	62.3	61.9	62.3	62.7	63	0
18	66.7	66.2	66.2	66.1	66.7	67.2	67.7	0
21	68.9	69.1	69.5	69.7	70.4	71.2	71.8	0
24	70.5	71.3	72.1	72.6	73.6	74.5	75.3	0
27	71.6	73.1	74.2	75.1	76.3	77.3	78.3	0
30	72.5	74.5	76	77.2	78.5	79.8	80.9	0
33	73.2	75.7	77.5	79	80.4	81.9	83.2	0
36	73.6	76.6	78.6	80.4	82.1	83.6	85.1	0
39	74	77.3	79.6	81.6	83.4	85.1	86.8	0
42	74.3	77.8	80.4	82.7	84.6	86.4	88.2	0
45	74.4	78.3	81.1	83.5	85.6	87.5	89.4	0
48	74.6	78.6	81.6	84.3	86.4	88.5	90.5	0
51	74.7	78.9	82	84.9	87.1	89.3	91.4	0
54	74.8	79.1	82.4	85.4	87.7	90	92.2	0
57	74.8	79.3	82.7	85.8	88.2	90.6	92.8	0
60	74.9	79.5	82.9	86.2	88.6	91.1	93.4	0
0	0	0	0	0	0	0	0	0

Time Values

0
0

0	0
3	0
6	0
9	0
12	0
15	0
18	0
21	0
24	0
27	0
30	0
33	0
36	0
39	0
42	0
45	0
48	0
51	0
54	0
57	0
60	0
0	0

Temperature Values

0
0

25	25	25	25	25	25	25	0
38	36.3	35.7	35.2	35.1	35.1	35	0
47.6	45.3	44.5	43.7	43.7	43.7	43.7	0
54.7	52.5	51.6	50.9	51	51	51.1	0
59.9	58.1	57.5	56.9	57.1	57.3	57.5	0
63.8	62.7	62.3	61.9	62.3	62.7	63	0
66.7	66.2	66.2	66.1	66.7	67.2	67.7	0
68.9	69.1	69.5	69.7	70.4	71.2	71.8	0
70.5	71.3	72.1	72.6	73.6	74.5	75.3	0
71.6	73.1	74.2	75.1	76.3	77.3	78.3	0
72.5	74.5	76	77.2	78.5	79.8	80.9	0
73.2	75.7	77.5	79	80.4	81.9	83.2	0
73.6	76.6	78.6	80.4	82.1	83.6	85.1	0
74	77.3	79.6	81.6	83.4	85.1	86.8	0
74.3	77.8	80.4	82.7	84.6	86.4	88.2	0
74.4	78.3	81.1	83.5	85.6	87.5	89.4	0
74.6	78.6	81.6	84.3	86.4	88.5	90.5	0
74.7	78.9	82	84.9	87.1	89.3	91.4	0
74.8	79.1	82.4	85.4	87.7	90	92.2	0
74.8	79.3	82.7	85.8	88.2	90.6	92.8	0
74.9	79.5	82.9	86.2	88.6	91.1	93.4	0
0	0	0	0	0	0	0	0

Figure 4.11
Result of selecting Time and Temperature subarrays.

4.3 ADDING ARRAYS

One of the most fundamental matrix operations is adding two arrays, such as the [A] and [B] arrays shown here:

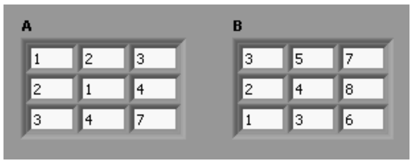
$$A = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 1 & 4 \\ 3 & 4 & 7 \end{bmatrix} \qquad B = \begin{bmatrix} 3 & 5 & 7 \\ 2 & 4 & 8 \\ 1 & 3 & 6 \end{bmatrix}$$

In order to add two arrays, they must be the same size (same number of rows and columns). The process used to add two matrices is to add corresponding elements from each matrix. For example, when [A] and [B] are added together, the top-left element of the resulting array will be $1 + 3 = 4$. All of the corresponding elements are similarly added together. LabVIEW's Add function is used to add arrays and matrices.

Functions Palette / Mathematics Group / Numeric Group / Add function

The following steps are used to add two arrays in LabVIEW:

On the front panel (see Figure 4.12) . . .



A

1	2	3
2	1	4
3	4	7

B

3	5	7
2	4	8
1	3	6

Figure 4.12
The *A* and *B* arrays on the front panel.

1. Create two array controls on the front panel. The steps required to create an array are as follows:
   - Place the Array container on the front panel.

     Controls Palette / Modern Group / Array, Matrix & Cluster Group / Array

   - Place one numeric control (from either location) inside the Array container.

     Controls Palette / Modern Group / Numeric Group / Numeric Control

     Controls Palette / Express Group / Numeric Controls Group / Num Ctrl

   - Add a dimension to create a 2D array. Right-click on the Index Display and select **Add Dimension** from the pop-up menu.
   - Expand the size of the array. Drag the handles at the sides of the array to change the number of displayed array elements.
2. Enter values into the arrays. Once the right number of rows and columns are displayed, double-click in each array element to enter the value for the element.

   Note: The *Index Display* for each matrix has been hidden in Figure 4.12. The Index Display is used to scroll through a large matrix. When the entire matrix can be seen in the control, the Index Display is not needed. Hiding the Index Display simplifies the front panel display.

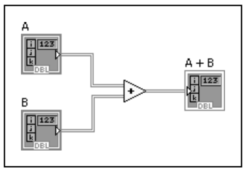


Figure 4.13
Block diagram for adding arrays *A* and *B*.

On the block diagram (see Figure 4.13) . . .

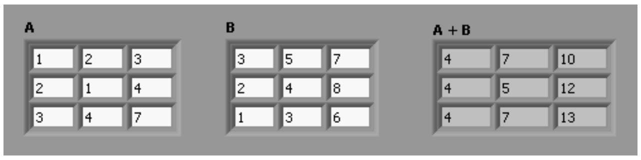
3. Place an Add function on the block diagram.

   Functions Palette / Mathematics Group / Numeric Group / Add function

4. Wire the matrix output terminals to the Add function input terminals.
5. Right-click on the Add function output terminal and select **Create / Indicator** from the pop-up menu. (The created array indicator will be 2D, but it will need to be resized on the front panel to show three rows and three columns.)

Run the VI to add the arrays. The solution is shown in Figure 4.14.

Figure 4.14
The added arrays, *A* + *B*.



A

1	2	3
2	1	4
3	4	7

B

3	5	7
2	4	8
1	3	6

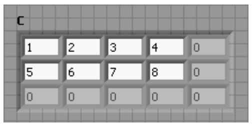
A + B

4	7	10
4	5	12
4	7	13

4.4 TRANSPOSE ARRAY

When an array is *transposed*, the rows and columns are interchanged. Any matrix or array can be transposed. The effect of transposing an array is most apparent when the array has significantly more rows than columns, or vice versa, so we will use the following array as an example:

$$C = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \end{bmatrix}$$



C

1	2	3	4	0
5	6	7	8	0
0	0	0	0	0

Figure 4.15
The C array defined on the front panel.

The first step is to create the array on the front panel, as shown in Figure 4.15.

In Figure 4.15 the visible portion of the array has been expanded to illustrate how LabVIEW handles variable array sizes. An array can be as large as needed, but unused elements are indicated in gray. As values are entered into the array, the active elements are shown with a white background. With this approach, it is easy to create arrays of whatever size is needed.

LabVIEW provides two functions for transposing an array:

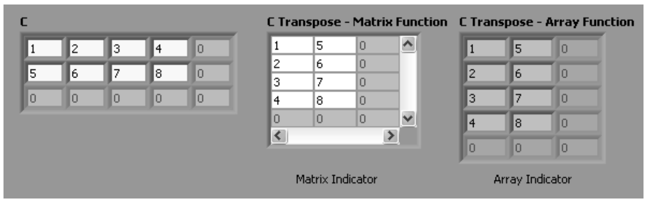
- Matrix function: **Transpose Matrix**

  Function Palette / Mathematics Group / Linear Algebra Group / Transpose Matrix

- Array function: **Transpose 2D Array**

  Function Palette / Programming Group / Array Group / Transpose 2D Array

To demonstrate that both functions generate the same result, we used both functions in the Matrix Transpose VI shown in Figure 4.16 (front panel) and Figure 4.17 (block diagram).



1	2	3	4	0
5	6	7	8	0
0	0	0	0	0

1	5	0
2	6	0
3	7	0
4	8	0
0	0	0

1	5	0
2	6	0
3	7	0
4	8	0
0	0	0

Figure 4.16 The results of transposing using the Transpose Matrix function and the Transpose 2D Array function.

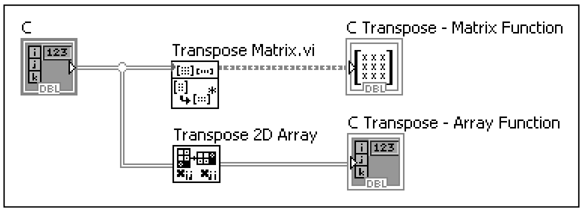


Figure 4.17 Transposing the C array using two methods.

Notice that the Transpose Matrix function created a matrix result, whereas the Transpose 2D Array created an array result. The results are numerically equivalent, but the values are stored in variables of differing data types.

4.5 MULTIPLYING AN ARRAY BY A SCALAR

The process of multiplying an array or matrix by a *scalar* (a single value) is termed *scalar multiplication*. Each element of the array is multiplied by the scalar value. For example, if the C array is multiplied by the scalar 10, each element of the C array is multiplied by 10.

$$C = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \end{bmatrix}$$

Functions Palette / Mathematics Group / Numeric Group / Multiply Function

A VI for multiplying an array by a scalar is illustrated in Figure 4.18 (front panel) and Figure 4.19 (block diagram).