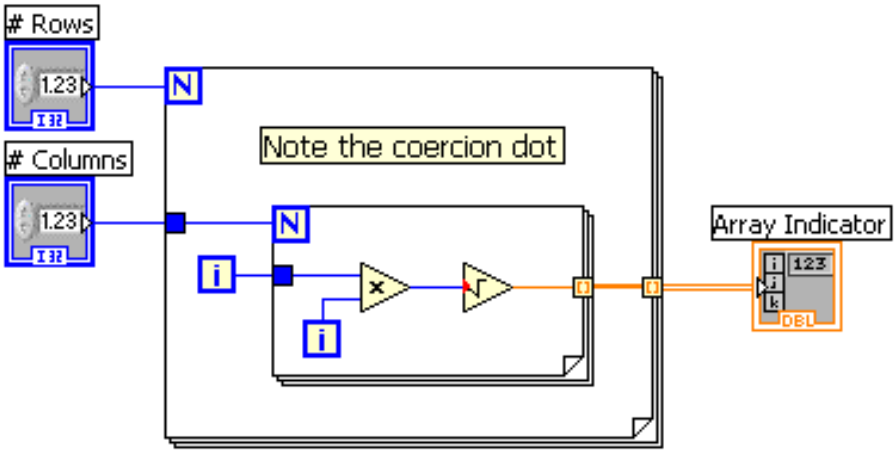
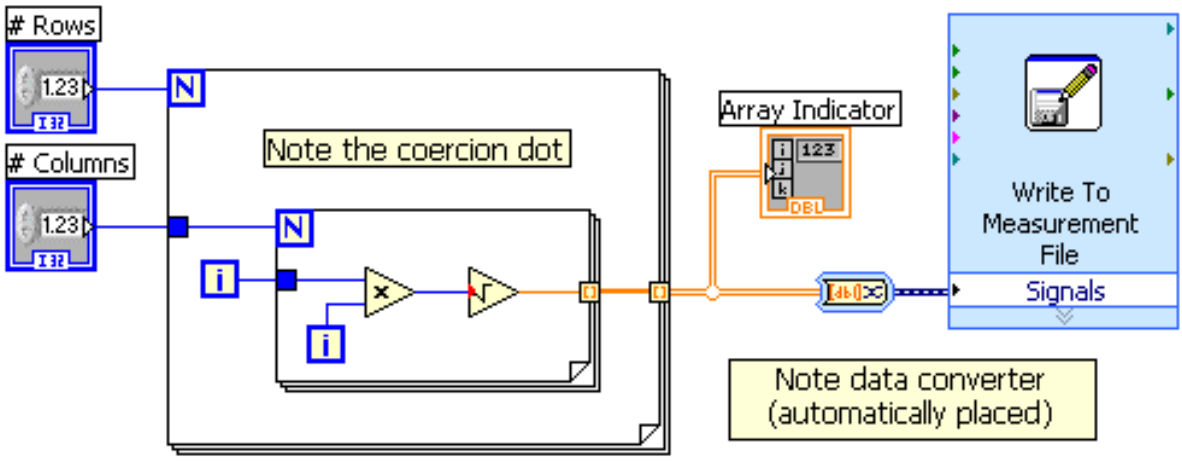
Worked Example: Fictitious Data Logger

The code shown below generates a 2-D array of double precision, floating-point numbers, of specified number of rows and number of columns:



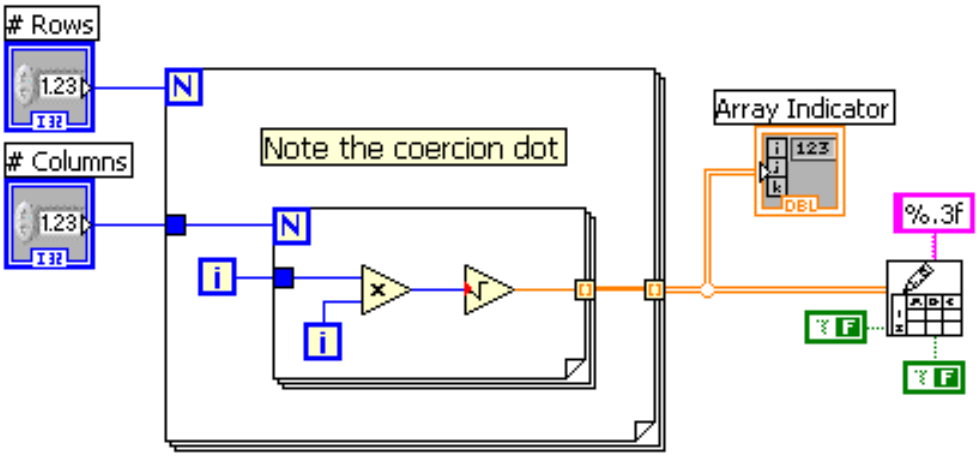
Consider different ways of saving the generated data to a simple text file that can be opened in Excel. The easiest method uses the *Write to LabVIEW Measurement File* Express VI from the *File I/O* palette. Note that Express VI's tend to work with the *Dynamic Data Type* (also called *Signals*), which often require specialized converters to interact with the more traditional functions and subVI's. In this example, the *Convert to Dynamic Data* Express VI is automatically placed (note that it still needs to be configured in order to work properly):



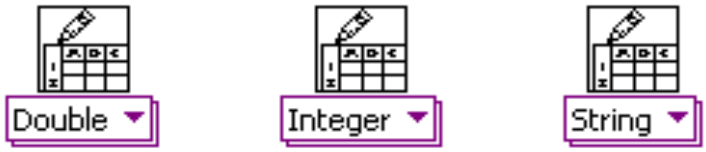
The *Write to LabVIEW Measurement File* Express VI has a good deal of advanced formatting and file management capabilities that this simple example does not

need. These additional capabilities explain the added complexity of the underlying programming, which can be seen if you right-click on the icon, select *Open Front Panel*, and proceed with the subsequent steps.

An equally simple approach is to use the *Write to Spreadsheet File* subVI from the same palette. Note that this subVI is **polymorphic**, meaning that it can automatically adapt to a variety of input data types. As shown, the subVI accepts a 2-D array of double precision numbers to format into a single text string, and write to a file:

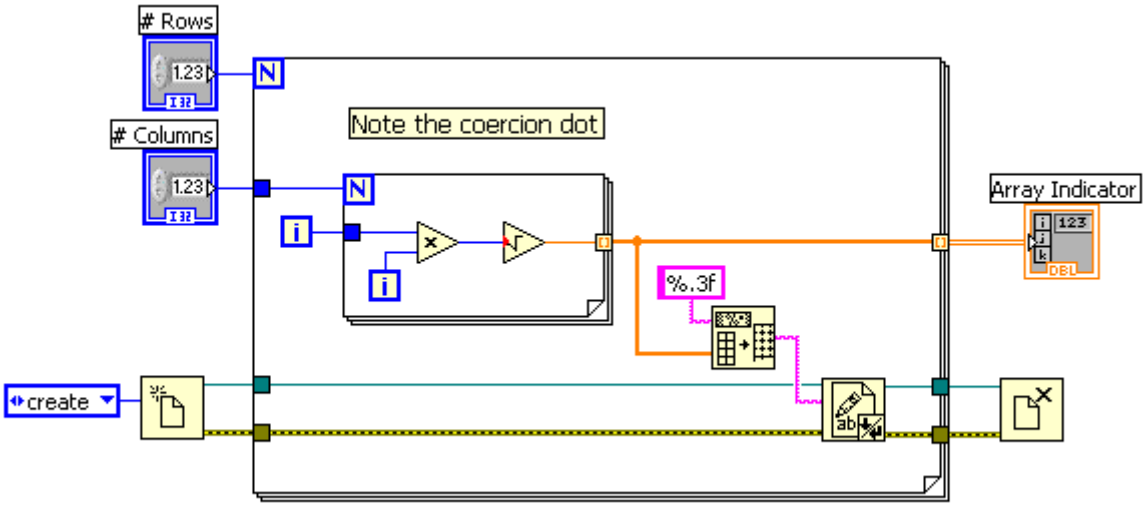


It is possible to force a particular input data type by right-clicking on the subVI icon and selecting *Visible Items>>Polymorphic Selector*, and then choosing the desired instance:



The code behind the *Write to Spreadsheet File* subVI is also available, simply by double-clicking on the icon. It has fewer levels and significantly less complexity than the Express VI, because it does not include as many capabilities.

Perhaps the most efficient, but most involved means is to use the low-level functions from the *File I/O* palette:



This approach is different in that it sequentially writes the individual rows of the 2-D array to the file, one at a time as the outer loop iterates. The operations that individually create, and eventually close the file reference are separated from the function that actually writes the data. In addition, we need to "manually" convert the numeric data to a "spreadsheet string" representation (a tab-delimited string with line feeds) in order for the write function to proceed. This approach is clearly more difficult to program than the other two methods, but allows more detailed control over the process.