creates a wave named W_MultiThreadingArraySizes and displays it in a table. This shows you the current threshold for each operation that supports automatic multithreading. The wave includes dimension labels so you can see which row represents which operation's threshold.

The meaning of a given threshold value depends on the operation. For most operations the threshold is in terms of the number of points in the input wave. For some operations the threshold depends on the complexity of the calculation. For example, the threshold for the CurveFit operation takes the complexity of the fitting function into account.

You can change the threshold for a given operation by setting the data for the appropriate row of W_MultiThreadingArraySizes and passing it back to the MultiThreadingControl operation using the setThresholds keyword. For example:

```
W_MultiThreadingArraySizes[%ICA]=5000      // Set the ICA threshold
MultiThreadingControl setThresholds=W_MultiThreadingArraySizes  // Apply
```

You should not change any aspect of the wave other than the threshold values.

Multithreading and Roundoff Error

Every floating point operation can potentially lead to roundoff error. Normally, if you run the exact same code on the the exact same data, you get the exact same roundoff error and the exact same result.

This is not necessarily the case with multithreaded calculations Because the partitioning of work among threads depends on circumstances, such as how many threads are free, different calculation runs on the same data may partition work differently. This can lead to differences in roundoff error and therefore differences in results.

You can obtain reproducibility at the expense of performance by turning off automatic multithreading:

```
MultiThreadingControl setMode=0
```

Although this provides reproducibility, it does not necessarily reduce the size of roundoff error.

Examples

```
MultiThreadingControl setMode=0         // Disable automatic multithreading
MultiThreadingControl setMode=8         // Always multithread regardless of wave size
```

See Also

Automatic Parallel Processing with TBB on page IV-323

Automatic Parallel Processing with MultiThread on page IV-323

ThreadSafe Functions on page IV-106, **ThreadSafe Functions and Multitasking** on page IV-329

NameOfWave

NameOfWave(*wave*)

The NameOfWave function returns a string containing the name of the specified wave.

In a user-defined function that has a parameter or local variable of type WAVE, NameOfWave returns the actual name of the wave identified by the WAVE reference. It can also be used with wave reference functions such as **WaveRefIndexedDFR**.

NameOfWave does not return the full data folder path to the wave. Use **GetWavesDataFolder** for this information.

A null wave reference returns a zero-length string. This might be encountered, for instance, when using WaveRefIndexedDFR in a loop to act on all waves in a data folder, and the loop has incremented beyond the highest valid index.

Examples

```
Function ZeroWave(w)
    Wave w
    w = 0
    Print "Zeroed the contents of", NameOfWave(w)
End
```

See Also

See **WAVE**; the **GetWavesDataFolder** and **WaveRefIndexed** functions; and **Wave Reference Functions** on page IV-197.