```
   endif

   STRUCT ThreadIOData ioData

   // Prepare input
   Make/O ioDataArray      // This wave will be redimensioned by StructPut
   Variable i, imax=100
   for(i=0; i<imax; i+=1)
      ioData.x = i                        // Set input data
      StructPut ioData, ioDataArray[i]    // Pack structure into wave column
   endfor

   // Generate output
   Make/O/N=(imax) threadOutput
   MultiThread threadOutput = Worker(ioDataArray, p)

   // Extract output
   Make/O/N=(imax) outputData
   for(i=0; i<imax; i+=1)
      StructGet ioData, ioDataArray[i]
      outputData[i] = ioData.out
   endfor

   KillWaves ioDataArray, threadOutput
End

ThreadSafe Function Worker(w, point)
   WAVE w
   Variable point

   STRUCT ThreadIOData ioData
   StructGet ioData, w[point]       // Extract structure from wave column

   ioData.out = sin(ioData.x)       // Calculate of output data

   StructPut ioData, w[point]       // Pack structure into wave column

   // The return value from the thread worker function is accessible
   // via ThreadReturnValue. It is not used in this example.
   return point
End
```

To run the demo, execute:

```
Demo()
```

ThreadSafe Functions and Multitasking

Igor supports two multitasking techniques that are easy to use:

- **Automatic Parallel Processing with TBB**
- **Automatic Parallel Processing with MultiThread**

This section discusses the third technique, **ThreadSafe Functions**, which expert programmers can use to create complex, preemptive multitasking background tasks.

Preemptive multitasking uses the following functions and operations:

ThreadProcessorCount	**ThreadGroupCreate**
ThreadStart	**ThreadGroupPutDF**