```
Function MTFillWaveThreadAtATime(dest)
   WAVE dest

   Variable ncol= DimSize(dest,1)
   Variable col,nthreads= ThreadProcessorCount
   Variable threadGroupID= ThreadGroupCreate(nthreads)
   Variable dummy

   for(col=0; col<ncol; col+=1)
      // Get index of a free thread - Requires Igor Pro 6.23 or later
      Variable threadIndex = ThreadGroupWait(threadGroupID,-2) - 1
      if (threadIndex < 0)
         dummy = ThreadGroupWait(threadGroupID, 50)// Let threads run a while
         col -= 1                    // Try again for the same column
         continue                    // No free threads yet
      endif
      ThreadStart threadGroupID, threadIndex, MyWorkerFunc(dest,col)
   endfor

   // Wait for all threads to finish
   do
      Variable threadGroupStatus = ThreadGroupWait(threadGroupID,100)
   while(threadGroupStatus != 0)

   dummy = ThreadGroupRelease(threadGroupID)
End
```

The ThreadGroupWait statement suspends the main thread for a while so that the preemptive threads get more processor time. The parameter 50 is the number of milliseconds to wait. You should tune this for your application.

Input/Output Queues

In this example, data folders containing a data wave and a string variable that specifies the task to be performed are created and posted to the thread group's input queue. The thread worker function waits for an input data folder to become available. It then processes the input and posts an output data folder to the thread group's output queue from which it is retrieved by the main thread.

```
ThreadSafe Function MyWorkerFunc()
   do
      do
         DFREF dfr = ThreadGroupGetDFR(0,1000)// Get free data folder from input queue
         if (DataFolderRefStatus(dfr) == 0)
            if( GetRTError(2) ) // New in 6.20 to allow this distinction:
               Print "worker closing down due to group release"
            else
               Print "worker thread still waiting for input queue"
            endif
         else
            break
         endif
      while(1)

      SVAR todo = dfr:todo
      WAVE jack = dfr:jack

      NewDataFolder/S outDF

      Duplicate jack,outw // WARNING: outw must be cleared. See WAVEClear below
      String/G did= todo
      if( CmpStr(todo,"sin") )
         outw= sin(outw)
      else
         outw= cos(outw)
      endif
```

```
      // Clear outw so Duplicate above does not try to use it and to allow
      // ThreadGroupPutDF to succeed.
      WAVEClear outw

      ThreadGroupPutDF 0,:    // Put current data folder in output queue

      KillDataFolder dfr      // We are done with the input data folder
   while(1)

   return 0
End

Function DemoThreadQueue()
   Variable i,ntries= 5,nthreads= 2

   Variable threadGroupID = ThreadGroupCreate(nthreads)

   for(i=0;i<nthreads;i+=1)
      ThreadStart threadGroupID,i,MyWorkerFunc()
   endfor

   for(i=0;i<ntries;i+=1)
      NewDataFolder/S forThread
      String/G todo
      if( mod(i,3) == 0 )
         todo= "sin"
      else
         todo= "cos"
      endif
      Make/N= 5 jack= x + gnoise(0.1)

      WAVEClear jack

      ThreadGroupPutDF threadGroupID,: // Send current data folder to input queue
   endfor

   for(i=0;i<ntries;i+=1)
      do
         // Get results in free data folder
         DFREF dfr= ThreadGroupGetDFR(threadGroupID,1000)
         if ( DatafolderRefStatus(dfr) == 0 )
            Print "Main still waiting for worker thread results."
         else
            break
         endif
      while(1)

      SVAR did = dfr:did
      WAVE outw = dfr:outw

      Print "task= ",did,"results= ",outw

      // The next two statements are not really needed as the same action
      // will happen the next time through the loop or, for the last iteration,
      // when this function returns.
      WAVEClear outw      // Redundant because of the WAVE statement above
      KillDataFolder dfr  // Redundant because dfr refers to a free data folder
   endfor

   // This terminates the MyWorkerFunc by setting an abort flag
   Variable tstatus= ThreadGroupRelease(threadGroupID)
   if( tstatus == -2 )
      Print "Thread would not quit normally, had to force kill it. Restart Igor."
   endif
End
```

Typical output:

```
•DemoThreadQueue()
   task=   sin  results=
outw[0]= {0.994567,0.660904,-0.516692,-0.996884,-0.63106}
   task=   cos  results=
outw[0]= {0.0786631,0.709576,0.873524,0.0586175,-0.718122}
   task=   cos  results=
```