```
Function/WAVE Subroutine2()
   DFREF dfSav= GetDataFolderDFR()

   // Create free data folder and set as current data folder
   SetDataFolder NewFreeDataFolder()

   // Create wave jack and an automatic wave reference to it
   Make jack={1,2,3}    // jack is not free - it is an object in a data folder

   SetDataFolder dfSav  // Change the current data folder
   // There are now no references to the free data folder so it is deleted
   // but wave jack remains because there is a reference to it.
   // jack is now a free wave.

   return jack          // Return reference to free wave to calling routine
End

Function MainRoutine2()
   WAVE w= Subroutine2()   // Wave reference w references free wave jack
   Print w
End   // Wave reference w ceases to exist so free wave jack is deleted
```

Not shown in this section is the case of a free wave that persists because a reference to it is stored in a wave reference wave. That situation is illustrated by the **Automatic Parallel Processing with MultiThread** on page IV-323 example.

Free Wave Leaks

A leak occurs when an object is created and is never released. Leaks waste memory. Igor uses wave reference counting to prevent leaks but in the case of free waves there are special considerations.

A free wave must be stored in a wave reference variable in order to be automatically released because Igor does the releasing when a wave reference variable goes out of scope.

To avoid the possibility of leaks, when you call a function that returns a free wave, always store the function result in a wave reference variable.

The following example results in two memory leaks:

```
Function/WAVE GenerateFree()
   return NewFreeWave(2,3)
End

Function Leaks()
   Duplicate/O GenerateFree(),dummy             // This leaks
   Variable maxVal = WaveMax(generateFree())    // So does this
End
```

Both lines leak because the free wave returned by GenerateFree is not stored in any wave reference variable. By contrast, this function does not leak:

```
Function NoLeaks()
   Wave w = GenerateFree()             // w references the free wave
   Duplicate/O w,dummy
   Variable maxVal = WaveMax(w)        // WaveMax is a built-in function
   // The free wave is released when w goes out of scope
End
```

In the Leaks function, there would be no leak if you replaced the call to WaveMax with a call to a user-defined function. This is because Igor automatically creates a wave reference variable when you pass a wave to a user-defined function. Because this distinction is subtle, it is best to always store a free wave in your own explicit wave reference variable.