Because memory is not deallocated until all references to a given wave are gone, memory may not be freed when you think. Consider the function:

```
Function myfunc()
   Make/N=10E6 bigwave
   // do stuff
   FunctionThatKillsBigwave()
   // do more stuff
End
```

The memory allocated for bigwave is not deallocated until the function returns because Make creates an automatic wave reference variable which, by default, exists until the function returns. To free memory before the function returns, use the **WAVEClear**:

```
Function myfunc()
   Make/N=10E6 bigwave
   // do stuff
   FunctionThatKillsBigwave()
   WAVEClear bigwave
   // do more stuff
End
```

The WAVEClear command takes a list of WAVE reference variables and stores NULL into them. It is the same as:

```
WAVE/Z wavref= $""
```

Detecting Wave Leaks

Igor Pro 9.00 introduced two ways to detect wave leaks: **IgorInfo**(16) and **Wave Tracking**. These are tools for advanced Igor programmers. Wave tracking is discussed in the next section.

IgorInfo(16) is a quick and lightweight way to detect possible wave leaks. Unlike wave tracking, IgorInfo(16) can be used independently by different procedure packages without interference among them.

IgorInfo(16) returns the total number of waves of all kinds in existence at any given time. You can use it like this:

```
Function DetectWaveLeaks()
   int originalNumberOfWaves = str2num(IgorInfo(16))

   <Do a lot of processing that you suspect might leak waves>

   int newNumberOfWaves = str2num(IgorInfo(16))
   int change = newNumberOfWaves - originalNumberOfWaves
   if (change > 0)
      Beep
      Printf "Possible leak: Number of waves changed by %d\r", change
   endif
End
```

IgorInfo(16) may sometimes create false positives. For example, the Igor debugger and the Igor Data Browser sometimes create hidden waves for their internal use. An example is the debugger if you choose Show Waves in Graph from the Local Waves, Strings, and SVARs popup menu. This can cause a discrepancy in the number of waves at the start and end of a function but it is not a real leak as Igor will eventually reuse or kill the wave.

Because of the possibility of false positives, you should consider a discrepancy such as detected by the DetectWaveLeaks function above evidence of a possible leak only. To confirm the leak, rerun the function without the use of the debugger or any other user-interface interactions. If it still appears to be a leak, use Wave Tracking to investigate the cause.