Wave Tracking

The **WaveTracking** operation is a tool for detecting and investigating wave leaks. WaveTracking was introduced in Igor Pro 9.00.

A fast alternative and lightweight alternative to wave tracking is **IgorInfo**(16). For details, see **Detecting Wave Leaks** above.

Igor waves fall into three categories and each category can be tracked separately: global, free, and local.

Global waves are waves stored in the main data folder hierarchy, i.e., in the root data folder or one of its descendants. These are the usual waves that you work with on a regular basis. They are global in the sense that they are persistent (you must explicitly kill them) and are accessible from any context (any function or from the command line) at any time.

Free waves are waves that are not contained in any data folder. You create them in user-defined functions via the **NewFreeWave** function, the **Make**/FREE operation, or other operations that support the /FREE flag. You use a wave reference variable to access a free wave (see **Wave References** on page IV-71). When the last wave reference for a given free wave has gone out of scope or is cleared by **WAVEClear**, Igor automatically kills the wave.

Local waves are waves that are stored in a free data folder or a descendant of a free data folder. They are local in the sense that they are automatically killed and are accessible only if you hold a data folder reference for the containing data folder or its ancestor.

Local waves become free waves when the containing data folder is destroyed if a wave reference still remains. See **Free Wave Lifetime** on page IV-92 and **Free Data Folder Lifetime** on page IV-97.

The topic **Free Wave Leaks** on page IV-94 illustrates how you can permanently leak a free wave. After they are leaked, there is no way to access them, so the only way to recover the memory is to restart Igor. You can use the WaveTracking operation to detect and investigate such leaks.

Sometimes you may have a backlog of global waves that were created for temporary use but not deleted. These waves are stored in memory and on disk when you save an experiment. You can use the WaveTracking operation to identify and clean up the backlog.

The **WaveTracking** operation allows you to keep track of the number of waves created but not killed over a given period of time and to get information on those waves. When Igor starts up, tracking is turned off. You must turn it on, as illustrated below, before you run code that you suspect is causing a problem.

WaveTracking provides two modes. Counter mode tells you the number waves of a given category created minus the number killed since counting started. Tracker mode keeps a list of every wave created but not yet been killed since tracking started. Tracker mode adds to the time it takes to create and kill waves, so counter mode is useful for detecting leaks while tracker mode is useful for investigating leaks once they are detected.

To illustrate, we will use the Leaks function defined here. It demonstrates how free wave leaks can occur and in the process leaks two free waves.

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

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

If you want to follow along with the example, copy these functions to your procedure window and then close it.

First we turn on the free wave counter: