Instance Notation

There is a problem that occurs when you have multiple instances of the same wave in a graph or multiple instances of the same object in a layout. For example, assume you want to graph yWave versus xWave0, xWave1, and xWave2. To do this, you need to execute:

```
Display yWave vs xWave0
AppendToGraph yWave vs xWave1
AppendToGraph yWave vs xWave2
```

The result is a graph in which yWave occurs three times. Now, if you try to remove or modify yWave using:

```
RemoveFromGraph yWave
```

or

```
ModifyGraph lsize(yWave)=2
```

Igor will always remove or modify the first instance of yWave.

Instance notation provides a way for you to specify a particular instance of a particular wave. In our example, the command

```
RemoveFromGraph yWave#2
```

will remove instance number 2 of yWave and

```
ModifyGraph lsize(yWave#2)=2
```

will modify instance number 2 of yWave. Instance numbers start from zero so "yWave" is equivalent to "yWave#0". Instance number 2 is the instance of yWave plotted versus xWave2 in our example.

Where necessary to avoid ambiguity, Igor operation dialogs (e.g., Modify Trace Appearance) automatically use instance notation. Operations that accept trace names (e.g., ModifyGraph) or layout object names (e.g., ModifyObject) accept instance notation.

A graph can also display multiple waves with the same name if the waves reside in different datafolders. Instance notation applies to the this case also.

Instance Notation and $

The $ operator can be used with instance notation. The # symbol may be either inside the string operand or may be outside. For example `$"wave0#1"` or `$"wave0"#1`. However, because the # symbol may be inside the string, the string must be parsed by Igor. Consequently, unlike other uses of $, the wave name portion must be surrounded by single quotes if liberal names are used. For example, suppose you have a wave with the liberal name of `'ww#1'` plotted twice. The first instance would be `$"'ww#1'"` and the second `$"'ww#1'#1"` whereas `$"ww#1"` would reference the second instance of the wave ww.

Object Indexing

The ModifyGraph, ModifyTable and ModifyLayout operations, used to modify graphs, tables and page layouts, each support another method of identifying the object to modify. This method, object indexing, is used to generate style macros (see **Graph Style Macros** on page II-350). You may also find it handy in other situations.

Normally, you need to know the name of the object that you want to modify. For example, assume that we have a graph with three traces in it and we want to set the traces' markers from a procedure. We can write:

```
ModifyGraph marker(wave0)=1, marker(wave1)=2, marker(wave2)=3
```

Because it uses the names of particular traces, this command is specific to a particular graph. What do we do if we want to write a command that will set the markers of three traces in *any* graph, regardless of the names of the traces? This is where object indexing comes in.

Using object indexing, we can write:

```
ModifyGraph marker[0]=1, marker[1]=2, marker[2]=3
```

This command sets the markers for the first three traces in a graph, no matter what their names are.