If you make a text wave or a complex wave, you need to tell the Igor compiler about that by using Wave/T or Wave/C. The compiler needs to know the type of the wave in order to properly compile the assignment statement.

Inline WAVE Reference Statements

You can create a wave reference variable using /WAVE=<name> in the command that creates the output wave. For example:

```
Function Example3(nameForOutputWave)
   String nameForOutputWave

   Make $nameForOutputWave/WAVE=w   // Make a wave and a wave reference
   w = x^2
End
```

Here /WAVE=w is an inline wave reference statement. It does the same thing as the standalone wave reference in the preceding section.

Here are some more examples of inline wave declarations:

```
Function Example4()
   String name = "wave1"
   Duplicate/O wave0, $name/WAVE=wave1
   Differentiate wave0 /D=$name/WAVE=wave1
End
```

When using an inline wave reference statement, you do not need to, and in fact can not, specify the type of the wave using WAVE/T or WAVE/C. Just use WAVE by itself regardless of the type of the output wave. The Igor compiler automatically creates the right kind of wave reference. For example:

```
Function Example5()
   Make real1, $"real2"/WAVE=r2     // real1, real2 and r2 are real
   Make/C cplx1, $"cplx2"/WAVE=c2   // cplx1, cplx2 and c2 are complex
   Make/T text1, $"text2"/WAVE=t2   // text1, text2 and t2 are text
End
```

Inline wave reference statements are accepted by those operations which automatically create a wave reference for a simple object name.

Inline wave references are not allowed after a simple object name.

Inline wave references are allowed on the command line but do nothing.

WAVE Reference Types

When wave references are created at compile time, they are created with a specific numeric type or are defined as text. The compiler then uses this type when compiling expressions based on the WAVE reference or when trying to match two instances of the same name. For example:

```
Make rWave           // Creates single-precision real wave reference

Make/C cWave         // Creates single-precision complex wave reference

Make/L int64Wave     // Creates signed 64-bit integer wave reference

Make/L/U int64Wave   // Creates unsigned 64-bit integer wave reference

Make/T tWave         // Creates text wave reference
```

These types then define what kind of right-hand side expression Igor compiles:

```
rWave = expression      // Compiles real expression as double precision

cWave = expression      // Compiles complex expression as double precision
```