A local wave becomes a free wave if the parent data folder is killed while other references to the wave exist, as shown in thus example:

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

   // Create a free data folder. It persists because it is the current
   // data folder (i.e., there is a reference to it in Igor itself).
   SetDataFolder NewFreeDataFolder()

   Make jack={1,2,3}    // jack is not a free wave at this point.
   // This also creates an automatic wave reference named jack.

   // The free data folder created above is killed because there
   // are no more references to it.
   SetDataFolder dfSav

   // The free data folder is now gone but jack persists
   // because of the wave reference to it and is now a free wave.

   return jack
End
```

In this example, jack was created as a local wave, not as a free wave, and consequently has the name jack, not _free_. When a local wave's data folder is killed, but the wave lives on due to a remaining wave reference, the wave retains the name it had when it was a local wave. See **Free Wave Names** (see page IV-95) for details.

Free Wave Created For User Function Input Parameter

If a user function takes a wave reference as in input parameter, you can create and pass a short free wave using a list of values as illustrated here:

```
Function Test(w)
   WAVE/D w
   Print w
End
```

You can invoke this function like this:

```
Test({1,2,3})
```

Igor automatically creates a free wave and passes it to Test. The free wave is automatically deleted when Test returns.

The data type of the free wave is determined by the type of the function's wave parameter. In this case the free wave is created as double-precision floating point because the wave parameter is defined using the /D flag. If /D were omitted, the free wave would be single-precision floating point. Wave/C/D would give a double-precision complex free wave. Wave/T would give a text free wave.

This list of values syntax is allowed only for user-defined functions because only they have code to test for and delete free waves upon exit.

Free Wave Lifetime

A free wave is automatically deleted when the last reference to it disappears.

Wave references can be stored in:

1. Wave reference variables in user-defined functions
2. Wave reference fields in structures
3. Elements of a wave reference wave (created by Make/WAVE)

The first case is the most common.

A wave reference disappears when:

1. The wave reference variable containing it is explicitly cleared using WaveClear.
2. The wave reference variable containing it is reassigned to refer to another wave.
3. The wave reference variable containing it goes out-of-scope and ceases to exist when the function in which it was created returns.
4. The wave reference wave element containing it is deleted or the wave reference wave is killed.

When there are no more references to a free wave, Igor automatically deletes it. This example illustrates the first three of these scenarios:

```
Function TestFreeWaveDeletion1()
   Wave w = NewFreeWave(2,3)  // Create a free wave with 3 points
   WaveClear w                // w no longer refers to the free wave
   // There are no more references to the free wave so it is deleted

   Wave w = NewFreeWave(2,3)  // Create a free wave with 3 points
   Wave w = root:wave0        // w no longer refers to the free wave
   // There are no more references to the free wave so it is deleted

   Wave w = NewFreeWave(2,3)  // Create a free wave with 3 points
End   // Wave reference w ceases to exist so the free wave is deleted
```

In the preceding example we used NewFreeWave which creates a free wave named '_free_' that is not part of any data folder. Next we will use Make/FREE instead of NewFreeWave. When reading this example, keep in mind that "Make jack" creates an automatic wave reference variable named jack:

```
Function TestFreeWaveDeletion2()
   Make /D /N=3 /FREE jack    // Create a free DP wave with 3 points
   // Make created an automatic wave reference named jack

   Make /D /N=5 /FREE jack    // Create a free DP wave with 5 points
   // Make created an automatic wave reference named jack
   // which refers to the 5-point jack.
   // There are now no references to 3-point jack so it is automatically deleted.

End   // Wave reference jack ceases to exist so free wave jack is deleted
```

In the next example, a subroutine returns a reference to a free wave to the calling routine:

```
Function/WAVE Subroutine1()
   Make /D /N=3 /FREE jack=p  // Create a free DP wave with 3 points
   return jack                // Return reference to calling routine
End

Function MainRoutine1()
   WAVE w= Subroutine1()   // Wave reference w references the free wave jack
   Print w
End   // Wave reference w ceases to exist so free wave jack is deleted
```

In the next example, the wave jack starts as an object in a free data folder (see **Free Data Folders** on page IV-96). It is not free because it is part of a data folder hierarchy even though the data folder is free. We call such a wave a local wave. When the free data folder is deleted, jack becomes a free wave.

When reading this example, keep in mind that the free data folder is automatically deleted when there are no more references to it. Originally, it survives because it is the current data folder and therefore is referenced by Igor internally. When it is no longer the current data folder, there are no more references to it and it is automatically deleted: