Delimited Text Compatibility With Igor7

As of Igor8, LoadWave/J provides better handling of quoted strings in delimited text files, such as comma-separated values files. See **Quoted Strings in Delimited Text Files** on page II-135 for details.

As a side-effect, in rare cases, Igor8 and later may produce different results for a given file compared to Igor7. For example, Igor7 would automatically identify this data as containing two numeric columns:

```
1;,2;
```

Igor8 automatically identifies this as containing two text columns, because of the unexpected semicolons.

If this change interferes with your file loading, you can cause Igor8 and later to work like Igor7 by setting bit 4 of the loadFlags parameter of the LoadWave /V flag. For example:

```
LoadWave/J/V={",", " ", 0, 16}      // 16 means bit 4 is set
```

This disables Igor8's support for quoted strings which causes LoadWave to behave more like it did in Igor7.

Another workaround is to force LoadWave to use a given data type for a given column. You can do this using the Column Format popup menu in the Loading Delimited Text dialog (see **Editing Wave Names** on page II-133) or by using the LoadWave /B flag.

Delimited Text Tweaks

There are many variations on the basic form of a delimited text file. We've tried to provide tweaks that allow you to guide Igor when you need to load a file that uses one of the more common variations. To do this, use the Tweaks button in the Load Waves dialog.

The Tweaks dialog can specify the space character as a delimiter. Use the LoadWave operation to specify other delimiters as well.

The main reason for allowing space as a delimiter is so that we can load files that use spaces to align columns. This is a common format for files generated by FORTRAN programs. Normally, you should use the fixed field text loader to load these files, not the delimited text loader. If you do use the delimited text loader and if space is allowed as a delimiter then Igor treats any number of consecutive spaces as a single delimiter. This means that two consecutive spaces do not indicate a missing value as two consecutive tabs would.

When loading a delimited file, by default Igor expects the first line in the file to contain either column labels or the first row of data. There are several tweaks that you can use for a file that doesn't fit this expectation.

Lines and columns in the tweaks dialog are numbered starting from zero.

Using the "Line containing column labels" tweak, you can specify on what line column labels are to be found if not on line zero. Using this and the "First line containing data" tweak, you can instruct Igor to skip garbage, if any, at the beginning of the file.

The "First line containing data", "Number of lines containing data", "First column containing data", and "Number of columns containing data" tweaks are designed to allow you to load any block of data from anywhere within a file. This might come in handy if you have a file with hundreds of columns but you are only interested in a few of them.

If "Number of lines containing data" is set to "auto" or 0, Igor will load all lines until it hits the end of the file. If "Number of columns containing data" is set to "auto" or 0, Igor will load all columns until it hits the last column in the file.

The proper setting for the "Ignore blanks at the end of a column" tweak depends on the kind of 1D data stored in the file. If a file contains some number of similar columns, for example four channels of data from a digital oscilloscope, you probably want all of the columns in the file to be loaded into waves of the same length. Thus, if a particular column has one or more missing values at the end, the corresponding points in the wave should contain NaNs to represent the missing value. On the other hand, if the file contains a number of dissimilar columns, then you might want to ignore any blank points at the end of a column so that the resulting waves