Integer Waves

Igor provides support for integer waves primarily to aid in data acquisition projects. They allow people who are interfacing with hardware to write/read directly into integer waves. This allows for slightly quicker live display and also saves the programmer from having to convert integers to floating point.

Integer waves are also appropriate for storing images.

Aside from memory considerations there is no other reason to use integer waves. You might expect that wave assignment statements would evaluate more quickly when an integer wave is the destination. This is not the case, however, because Igor still uses floating point for the assignment and only converts to integer for storage.

Date/Time Waves

Dates are represented in Igor date format - as the number of seconds since midnight, January 1, 1904. Dates before that are represented by negative values. Igor supports dates from the year -32768 to the year 32767.

As of Igor7, Igor uses the Gregorian calendar convention in which there is no year 0. The year before year 1 is year -1.

A date can not be accurately stored in the data values of a single precision or integer wave. Make sure to use double precision to store dates and times.

You must set the data units of a wave containing date or date/time data to "dat". This tells Igor that the wave contains date/time data and affects the default display of axes in graphs and columns in tables.

The following example illustrates the use of date/time data. First we create some date/time data and view it in a table:

```
Make/D/N=10/O testDate = date2secs(2011,4,p+1)
SetScale d 0, 0, "dat", testDate // Tell Igor this wave stores date/time data
```

We used `SetScale d` to set the data units of the wave to "dat".

Next we view the wave in a table:

```
Edit testDate
ModifyTable width(testDate)=120  // Make column wider
```

The data is displayed as dates but it is stored as numbers - specifically the number of seconds since January 1, 1904. We can see this by changing the column format:

```
ModifyTable format=1             // Display as integer
```

Now we return to date format:

```
ModifyTable format(testDate)=6   // Display as date again
```

Next we create some time data. This wave will not store dates and therefore does not need to be double-precision:

```
Make/N=10/O testTime = 3600*p    // Data is stored in seconds
AppendToTable testTime
```

Now we create a date/time wave by adding the time data to the date data. Since this wave will store dates it must be double-precision and must have "dat" data units. We accomplish this by using the Duplicate operation to duplicate the original date wave:

```
Duplicate/O testDate, testDateTime
AppendToTable testDateTime
ModifyTable width(testDateTime)=120    // Make wider
```