You can also enter times in time-of-day format, for example 1:45 PM or 1:45:00 PM or 13:45 or 13:45:00. A space before the AM or PM is allowed but not required. AM and am have no effect on the resulting time. PM adds 12 hours to the time. PM has no effect if the hour is already 12 or greater.

Igor stores times the same way whether they are time-of-day times or elapsed times. The difference is in how you think of the value and how you choose to display it in a graph.

Here are some valid time-of-day times:

```
00:00:00 (midnight)           12:00:00 (noon)
06:30:00 (breakfast time)     18:30:00 (dinner time)
06:30 (breakfast time)        18:30 (dinner time)
06:30 PM (dinner time)        18:30 PM (dinner time)
```

Here are some valid elapsed times:

```
-00:00:10   (T minus 10 and counting)
72:00:00    (an elapsed time covering three days)
4:17:33.25  (an elapsed time with fractional seconds)
```

The format for a date/time wave in a table defaults to Date, Time, or Date/Time depending on the content of the wave.

For displaying date/time data in a graph, see **Date/Time Axes** on page II-315.

Octal Numeric Formats

You can format any numeric column in a table as octal. Igor ignores the digits setting for columns displayed as octal and always displays all digits required for the supported range of values.

For floating point waves (single-precision and double-precision), negative values are displayed using a minus sign as this is necessary to accurately present the wave value. For example, -1 is displayed as -00000001.

It is possible to store in floating point waves a value that can't be displayed in octal because the value is fractional or exceeds the supported range of values. In these cases, the table will display an error message such as "# Can't format fractional value as octal" or "# Value too large to display as octal".

If you are using a floating point wave for data you think of as integer data of a given size, consider redimensioning the wave to the appropriate integer type using the **Redimension** operation. For example, if you are storing 16-bit signed integer data in a floating point wave, consider redimensioning the wave as 16-bit signed integer to get a more appropriate octal display in a table.

Hexadecimal Numeric Formats

You can format any numeric column in a table as hexadecimal. Igor ignores the digits setting for columns displayed as hexadecimal and always displays all digits required for the supported range of values.

For floating point waves (single-precision and double-precision), negative values are displayed using a minus sign as this is necessary to accurately present the wave value. For example, -1 is displayed as -00000001.

It is possible to store in floating point waves a value that can't be displayed in hexadecimal because the value is fractional or exceeds the supported range of values. In these cases, the table will display an error message such as "# Can't display fractional value as hex" or "# Value too large to display as hex".

Although SP waves are displayed as 32 bits using 8 hex digits, they have only 24 bits of precision so not all values greater than 2^24 (16,777,216) can be represented by an SP wave.

If you are using a floating point wave for data you think of as integer data of a given size, consider redimensioning the wave to the appropriate integer type using the **Redimension** operation. For example, if you are storing 16-bit signed integer data in a floating point wave, consider redimensioning the wave as 16-bit signed integer to get a more appropriate hexadecimal display in a table.