If *colNum* is out of bounds it will be clipped to valid values and an error reported. If the row dimension does not match the structure size, as much data as possible will be copied to the structure.

By default, data are read in big-endian, high-byte order (Motorola). This allows data written on one platform to be read on the other.

See Also

The **StructPut** operation for writing structure data to waves or strings.

StructPut

`StructPut [/B=b] structVar, waveStruct[[colNum]]`

`StructPut /S [/B=b] structVar, strStruct`

The StructPut operation copies the binary numeric data in a structure variable to a specified column in a wave or to a string variable. The data in the wave or string can be read out into another structure using **StructGet**.

Parameters

structVar is the name of a structure from which data will be exported.

waveStruct is the name of an existing wave to which data will be exported. Use the optional *colNum* parameter to specify a column in *waveStruct* to contain the data. The first column of *waveStruct* will be filled if *colNum* is omitted.

strStruct is the name of an existing string variable to which data will be exported.

Flags

/B=*b* Sets the byte ordering for writing of structure data.

b=0: Writes in native byte order.

b=1: Writes bytes in reversed order.

b=2: Default; writes data in big-endian, high-byte-first order (Motorola).

b=3: Writes data in little-endian, low-byte-first order (Intel).

/S Writes binary data to a string variable.

Details

The structure fields to be exported must contain only numeric data in either integer, floating point, or double precision format. If the structure contains any objects such as String, NVAR, WAVE, etc., then only the numeric data at the end of the structure is copied. If there is no suitable data at the end, an error is generated at compile time. Prior to Igor Pro 8, the presence of any illegal field would result in an error.

If needed, StructPut will redimension *waveStruct* to unsigned byte format, will set the number of rows to equal the size of the structure, and set the column dimension large enough to accommodate the size specified by *colNum*. You can think of *waveStruct* as a one-dimensional array of structure contents indexed by *colNum* although the wave is actually two-dimensional with each column containing a copy of a separate structure.

By default, data are written in big-endian, high-byte order (Motorola). This allows data written on one platform to be read on the other.

After you have exported the structure data to *waveStruct* or *strStruct* they will contain binary data that you cannot inspect directly. To view the contents of *waveStruct* or *strStruct*, you must use the original structure or use StructGet to export them into another structure.

See Also

The **StructGet** operation for reading structure data from waves or strings.