Type	Bit Number	Decimal Value	Hexadecimal Value	
8-bit integer	3	8	8	
16-bit integer	4	16	10	
32-bit integer	5	32	20	
64-bit integer	7	128	80	Requires Igor Pro 7 or later
unsigned	6	64	40	

The unsigned bit is used only with the integer types while the complex bit can be used with any numeric type. Set only one of bits 1-5 or bit 7 as they are mutually exclusive. See **Setting Bit Parameters** on page IV-12 for details about bit settings.

The returned value for non-numeric waves (text waves, wave-reference waves and data folder-reference waves) is 0.

Examples

```
Variable waveIsComplex = WaveType(wave) & 0x01
Variable waveIs32BitFloat = WaveType(wave) & 0x02
Variable waveIs64BitFloat = WaveType(wave) & 0x04
Variable waveIs8BitInteger = WaveType(wave) & 0x08
Variable waveIs16BitInteger = WaveType(wave) & 0x10
Variable waveIs32BitInteger = WaveType(wave) & 0x20
Variable waveIs64BitInteger = WaveType(wave) & 0x80
Variable waveIsUnsigned = WaveType(wave) & 0x40
```

See Also

For concepts related to selector = 1 or 2, see **Free Waves** on page IV-91, **Wave Reference Waves** on page IV-77 and **Data Folder Reference Waves** on page IV-82.

WaveUnits

WaveUnits(*waveName*, *dimNumber*)

The WaveUnits function returns a string containing the units for the given dimension.

Use *dimNumber*=0 for rows, 1 for columns, 2 for layers, and 3 for chunks. Use -1 to get the data units. If the wave is just 1D, *dimNumber*=0 returns X units and 1 returns data units. This behavior is just like the WaveMetrics procedure WaveUnits found in the WaveMetrics Procedures folder in previous versions of Igor Pro.

See Also

DimDelta, **DimOffset**, **DimSize**, **SetScale**

wfprintf

wfprintf *refNumOrStr*, *formatStr* [*flags*] *waveName* [, *waveName*]…

The wfprintf operation is like the **printf** operation except that it prints the contents of the named waves to a file whose file reference number is in *refNum*.

The **Save** operation also outputs wave data to a text file. Use Save unless you need the added flexibility provided by wfprintf.

Parameters

refNumOrStr is a numeric expression, a string variable or an SVAR pointing to a global string variable.

If a numeric expression, then it is a file reference number returned by the **Open** operation or an expression that evaluates to 1.

If *refNumOrStr* is 1, Igor prints to the history area instead of to a file.

If *refNumOrStr* is the name of a string variable, the wave contents are "printed" to the named string variable. *refNumOrStr* can also be the name of an SVAR to print to a global string:

```
SVAR sv = root:globalString
wfprintf sv, "", wave0
```

refNumOrStr can not be an element of a text wave.

The value of each named wave is printed to the file according to the conversion specified in *formatStr*.

formatStr contains one numeric conversion specification per column. See **printf**. If *formatStr* is "", wfprintf uses a default format which gives tab-delimited columns.

Flags

Note: /R must follow the *formatStr* parameter directly without an intervening comma.

/R=(*startX*,*endX*) Specifies an X range in the wave(s) to print.

/R=[*startP*,*endP*] Specifies a point range in the wave(s) to print.

Details

As of Igor7, wfprintf supports 1D and 2D waves. Previously it supported 1D waves only.

The number of conversion characters in *formatStr* must exactly match the number of wave columns in all input waves. With real waves, the total number of columns is limited to 100. With complex waves, the real column and imaginary column each count as a column and the total number of columns is limited to 200.

The only conversion characters allowed are fFeEgdouxXcs (the floating point, integer and string conversion characters). You cannot use an asterisk to specify field width or precision. If any of these restrictions is intolerable, you can use fprintf in a loop.

With integer conversion characters d, o, u, x, and X, applied to floating point waves, wfprintf rounds the fractional part.

In Igor Pro 9.00 and later, there is no limit to the length of an element of a text wave parameter passed to the wfprintf operation. Previously text wave element contents were clipped to about 500 bytes. There is also no limit to the length of formatStr. Previously it was limited to 800 bytes.

Examples

```
Function Example1()
   Make/O/N=10 wave0=sin(p*pi/10)                  // test numeric wave
   Make/O/N=10/T textWave= "row "+num2istr(p)      // test text wave
   Variable refNum
   Open/P=home refNum as "output.txt"// open file for write
   wfprintf refNum, "%s = %g\r"/R=[0,5], textWave, wave0    // print 6 values each
   Close refNum
End
```

The resulting output.txt file contains:

```
row 0 = 0
row 1 = 0.309017
row 2 = 0.587785
row 3 = 0.809017
row 4 = 0.951057
row 5 = 1
```

```
Function/S NumericWaveToStringList(w)
   Wave w                          // numeric wave (if text, use /T here and %s below)
   String list
   wfprintf list, "%g;" w          // semicolon-separated list
   return list
End

Print NumericWaveToStringList(wave0)
  0;0.309017;0.587785;0.809017;0.951057;1;0.951057;0.809017;0.587785;0.309017;
```

See Also

The **printf** operation for complete format and parameter descriptions and **Creating Formatted Text** on page IV-259. The **Open** operation about *refNum* and for another way of writing wave files.

The **Save** operation.