StringCRC

StringCRC(*inCRC*,*str*)

The StringCRC function returns a 32-bit cyclic redundancy check value of bytes in *str* starting with *inCRC*.

Pass 0 for *inCRC* the first time you call StringCRC for a particular stream of bytes as represented by the string data.

Pass the last-returned value from StringCRC for *inCRC* if you are creating a CRC value for a given stream of bytes through multiple calls to StringCRC.

Details

Polynomial used is:

x^32+x^26+x^23+x^22+x^16+x^12+x^11+x^10+x^8+x^7+x^5+x^4+x^2+x+1

See crc32.c in the public domain source code for zlib for more information.

See Also

Hash, **WaveHash**, **WaveCRC**

StringFromList

StringFromList(*index*, *listStr* [, *listSepStr*] [, *offset*])

The StringFromList function returns the *index*th substring extracted from *listStr* starting *offset* bytes into *listStr*. *listStr* should contain items separated by *listSepStr* which typically is ";".

Use StringFromList to extract an item from a string such as those returned by functions like **TraceNameList** and **AnnotationList**.

Parameters

index is the zero-based index of the list item that you want to get. If *index* < 0, or *index* ≥ the number of items in list, or if *listStr* or *listSepStr* is `""`, then a zero-length string is returned.

listStr contains a series of text items separated by *listSepStr*. The trailing separator is optional though recommended. For example, these are both valid lists:

```
"First;Second;"
"First;Second"
```

listSepStr is optional. If omitted it defaults to ";". Prior to Igor Pro 7.00, only the first byte of *listSepStr* was used. Now all bytes are used.

offset is optional and requires Igor Pro 7.00 or later. If omitted it defaults to 0. The search begins *offset* bytes into *listStr*. When iterating through lists containing large numbers of items, using the *offset* parameter provides dramatically faster execution.

Details

For optimal performance, especially with lists larger than 100 items, provide the *separatorStr* and *offset* parameters as shown in the DemoStringFromList example below. When using this technique, the *index* parameter must be 0 and the *offset* parameter controls which list item is returned.

Examples

```
Print StringFromList(0, "wave0;wave1;")              // Prints "wave0"
Print StringFromList(2, "wave0;wave1;")              // Prints ""
Print StringFromList(1, "wave0;;wave2")              // Prints ""

// Iterate quickly over a list using the offset parameter
Function DemoQuickStringFromList(list)
   String list      // A semicolon-separated string list

   String separator = ";"
   Variable separatorLen = strlen(separator)
   Variable numItems = ItemsInList(list, separator)

   Variable offset = 0
   Variable i
   for(i=0; i<numItems; i+=1)
      // When using offset, the index parameter is always 0
      String item = StringFromList(0, list, separator, offset)
```