number into local variable v1, skip a colon, and read another number into local variable v2, skip a colon, and read another number into local variable v3.

```
Variable v1, v2, v3
sscanf "12:30:45", "%d%*[:]%d%*[:]%d", v1, v2, v3
```

Here "%*[:]" means "read a colon character but don't store it anywhere". The "*" character must appear immediately after the percent. Note that there is nothing in the parameter list corresponding to the suppressed strings.

If the text in the scan string is not consistent with the text in the format string, sscanf may not read all of the values that you expected. You can check for this using the V_flag variable, which is set to the number of values read. This kind of inconsistency does not cause sscanf to return an error to Igor, which would cause procedure execution to abort. It is a situation that you can deal with in your procedure code.

The sscanf operation returns the following kinds of errors:

- Out-of-memory.
- The number of parameters implied by *formatStr* does not match the number of parameters in the *var* list.
- *formatStr* calls for a numeric variable but the parameter list expects a string variable.
- *formatStr* calls for a string variable but the parameter list expects a numeric variable.
- *formatStr* includes an unsupported, unknown or incorrectly constructed conversion specification.
- The *var* list references a global variable that does not exist.

Examples

Here is a simple example to give you the general idea:

```
Function SimpleExample()
   Variable v1, valuesRead
   sscanf "Value=1.234", "Value=%g", v1
   valuesRead = V_flag
   if (valuesRead != 1)
      Printf "Error: Expected 1 value, got %d values\r", valuesRead
   else
      Printf "Value read = %g\r", v1
   endif
End
```

For an example that uses sscanf to load data from a text file, choose File→Example Experiments→Programming→Load File Demo.

See Also

str2num, **strsearch**, **StringMatch**, **SplitString**

Stack

Stack [***flags***] [***objectName***][, ***objectName***]…

The Stack operation stacks the named layout objects in the top page layout.

Parameters

objectName is the name of a graph, table, picture or annotation object in the top page layout.

See Also

The **Tile** operation for details on the flags and parameters.

StackWindows

StackWindows [***flags***] [***windowName*** [, ***windowName***]…]

The StackWindows operation stacks the named windows on the desktop.

See Also

See the **TileWindows** operation for details on the flags and parameters.