

FunctionList

```
isNumeric = 0
elseif (type & 4)           // test for VARIABLE bit
    typeStr += "Variable "
elseif (type & 0x100)        // test for DFREF bit
    typeStr += "Data folder reference "
    isNumeric = 0
elseif (type & 0x200)        // test for STRUCTURE bit
    typeStr += "Struct "
    isNumeric = 0
elseif (type & 0x400)        // test for FUNCREF bit
    typeStr += "FuncRef "
    isNumeric = 0
endif

// print "real" or "complex" for numeric objects only
if (isNumeric)
    if (type & 1)           // test for COMPLEX bit
        typeStr += "cmplx "
    else
        typeStr += "real "
    endif
endif

if( !isReturnType && (type & 0x1000) ) // test for PASS BY REFERENCE bit
    typeStr += "reference "
endif

return typeStr
End
```

See Also

The **StringByKey** and **NumberByKey** functions.

StringByKey, **NumberByKey**, and **FunctionList** functions.

Regular Modules on page IV-236 and **Independent Modules** on page IV-238.

FunctionList

FunctionList(*matchStr*, *separatorStr*, *optionsStr*)

The FunctionList function returns a string containing a list of built-in or user-defined function names satisfying certain criteria. This is useful for making a string to list functions in a pop-up menu control. Note that if the procedures need to be compiled, then FunctionList will not list user-defined functions.

Parameters

Only functions having names that match *matchStr* string are listed. Use "*" to match all names. See **WaveList** for examples.

separatorStr is appended to each function name as the output string is generated. *separatorStr* is usually ";" for list processing (See **Processing Lists of Waves** on page IV-198 for details on list processing).

Use *optionsStr* to further qualify the list of functions. *optionsStr* is a string containing keyword-value pairs separated by commas. Available options are:

KIND: <i>nk</i>	Controls the kinds of functions returned.
<i>nk</i> =1:	List built-in functions.
<i>nk</i> =2:	List normal and override user-defined functions.
<i>nk</i> =4:	List external functions (defined by an XOP).
<i>nk</i> =8:	List only curve fitting functions; must be summed with 1, 2, 4, or 16. For example, use 10 to list user-defined fitting functions.
<i>nk</i> =16:	Include static user-defined functions; requires WIN: option, must be summed with 1, 2, or 8. To list only static functions, subtract the non-static functions using RemoveFromList.

SUBTYPE:*typeName*

Lists functions that have the type *typeName*. That is, you could use ButtonControl as *typeName* to list only functions that are action procedures for buttons.

VALTYPE: <i>nv</i>	<p>Restricts list to functions whose return type is a certain kind.</p> <p><i>nv</i>=1: Real-valued functions. <i>nv</i>=2: Complex-valued functions. <i>nv</i>=4: String functions. <i>nv</i>=8: WAVE functions <i>nv</i>=16: DFREF functions.</p> <p>Use a sum of these values to include more than one type. The return type is not restricted if this option is omitted.</p>
NPARAMS: <i>np</i>	Restricts the list to functions having exactly <i>np</i> parameters. Omitting this option lists functions having any number of parameters.
NINDVARS: <i>ni</i>	Restricts the list to fitting functions for exactly <i>ni</i> independent variables. NINDVARS is ignored if you have not elected to list curve fitting functions using the KIND option. Functions for any number of independent variables are listed if the NINDVARS option is omitted.
NRETURN: <i>nRet</i>	<p>Restricts list to user-defined functions that use the Multiple Return Syntax on page IV-36 to return <i>nRet</i> values.</p> <p>NRETURN is ignored if KIND:8 is specified.</p> <p>NRETURN was added in Igor Pro 9.00.</p>
PARAM_n_TYPE: <i>pType</i>	<p>Restricts list to functions whose <i>n</i>th input parameter type is a certain kind as described by <i>pType</i>.</p> <p>PARAM_n_TYPE is ignored if KIND:8 is specified.</p> <p>PARAM_n_TYPE was added in Igor Pro 9.00.</p> <p>Use PARAM_0_TYPE for the first parameter, PARAM_1_TYPE for the second parameter, and so on. You can use any number of PARAM_n_TYPE keyword=value pairs in a given FunctionList call.</p> <p>For each parameter, set <i>pType</i> to a parameter type code as described under Return Type and Parameter Type Codes on page V-286 below.</p> <p>See also Return Type and Parameter Type Code Examples on page V-287 below.</p>
RETURN_n_TYPE: <i>rType</i>	<p>Restricts list to user-defined functions whose <i>n</i>th Multiple Return Syntax return type is a certain kind described by <i>rType</i>.</p> <p>RETURN_n_TYPE is ignored if KIND:8 is specified.</p> <p>RETURN_n_TYPE was added in Igor Pro 9.00.</p> <p>Use RETURN_0_TYPE for the first return value, RETURN_1_TYPE for the second return value, and so on. You can use any number of RETURN_n_TYPE keyword=value pairs in a given FunctionList call.</p> <p>For each return value, set <i>rType</i> to a return type code as described under Return Type and Parameter Type Codes on page V-286 below.</p> <p>See also Return Type and Parameter Type Code Examples on page V-287 below.</p>
WIN: <i>windowTitle</i>	<p>Lists functions that are defined in the procedure window with the given title. “Procedure” is the title of the built-in procedure window.</p> <p>Note: Because the <i>optionsStr</i> keyword-value pairs are comma separated and procedure window names can have commas in them, the WIN:keyword must be the last one specified.</p>
WIN: <i>windowTitle</i> [<i>independentModuleName</i>]	

FunctionList

Lists functions that are defined in the named procedure window that belongs to the independent module *independentModuleName*. See **Independent Modules** on page IV-238 for details. Requires SetIgorOption IndependentModuleDev=1, otherwise no functions are listed.

Requires *independentModuleName*=ProcGlobal or SetIgorOption independentModuleDev=1, otherwise no functions are listed.

Note: The syntax is literal and strict: the window title must be followed by one space and a left bracket, followed directly by the independent module name and a closing right bracket.

WIN:[*independentModuleName*]

Lists functions that are defined in any procedure file that belongs to the named independent module.

Requires *independentModuleName*=ProcGlobal or SetIgorOption independentModuleDev=1, otherwise no functions are listed.

Note: The syntax is literal and strict: 'WIN:' must be followed by a left bracket, followed directly by the independent module name and a closing right bracket, like this:

```
FunctionList( . . . , "WIN: [myIndependentModuleName]" )
```

Return Type and Parameter Type Codes

These codes are used with the PARAM_N_TYPE and RETURN_N_TYPE keywords:

Type	Code	Code in Hex
Complex	1	0x1
Single Precision	2	0x2
Variable	4	0x4
Double Precision	4	0x4
Byte	8	0x8
16-bit Integer	16	0x10
32-bit Integer	32	0x20
Unsigned	64	0x40
/WAVE	128	0x80
Data folder reference	256	0x100
Structure	512	0x200
Function reference	1024	0x400
Pass by reference parameter	4096	0x1000
String	8192	0x2000
Wave	16384	0x4000
/Z	32768	0x8000

Special Return Type and Parameter Type Codes

These special codes are used with the PARAM_N_TYPE and RETURN_N_TYPE keywords:

Any real-valued numeric wave -2

Any complex-valued numeric wave -3