Pop-up items support all of the special characters available for user-defined menu definitions (see **Special Characters in Menu Item Strings** on page IV-133) with the exception that items in pop-up menus are limited to 50 bytes, keyboard shortcuts are not supported, and special characters must be enabled.

See Also

Prompt, **DoPrompt**, and **Pop-Up Menus in Simple Dialogs** on page IV-145.

See **WaveList**, **TraceNameList**, **ContourNameList**, **ImageNameList**, **FontList**, **MacroList**, **FunctionList**, **StringList**, and **VariableList** for functions useful in generating lists of Igor objects.

Chapter III-14, **Controls and Control Panels** for details about control panels and controls.

PopupContextualMenu

PopupContextualMenu [**/C=(*xpix*, *ypix*) /N /ASYN[=*func*]**] ***popupStr***

The PopupContextualMenu operation displays a pop-up menu.

The menu appears at the current mouse position or at the location specified by the /C flag.

The content of the menu is specified by *popupStr* as a semicolon-separated list of items or, if you include the /N flag, by a user-defined menu definition referred to by the name contained in *popupStr*.

If you omit the /ASYN flag, the menu is tracked and the operation does not return until the user makes a selection or cancels the menu by clicking outside of its window.

If you include /ASYN, the menu is displayed and the operation returns immediately. When the user makes a selection, then the result is sent to the specified function or to the user-defined menu's execution text. You can use /ASYN to allow a background task to continue while a contextual menu is popped up. /ASYN requires Igor Pro 7.00 or later.

Parameters

If *popupStr* specifies the pop-up menu's items (/N is not specified), then *popupStr* is a semicolon-separated list of items such as "yes;no;maybe;", or a string expression that returns such a list, such as **TraceNameList**.

The menu items can be formatted and checkmarked, like user-defined menus can. See **Special Characters in Menu Item Strings** on page IV-133.

If /N is specified, *popupStr* must be the name of a user-defined menu that also has the popupcontextualmenu keyword. See Example 3.

Flags

/ASYN — When used with /N: The user-defined menu is displayed and operation returns immediately. The result of menu selection is handled by the user-defined menu's execution text. See **User-Defined Menus** on page IV-125.

/ASYN=*func* — When used without /N: The user-defined menu is displayed and operation returns immediately. The result of menu selection or cancellation is handled by calling the named function, which must have the following format:

```
Function func(popupStr, selectedText, menuItemNum)
   String popupStr
   String selectedText
   Variable menuItemNum
```

In Igor Pro 8.05 and later, if the user cancels the menu selection, selectedText will be "" and menuItemNum will be 0.

/C=(*xpix*, *ypix*) — Sets the coordinates of the menu's top left corner.

Units are in pixels relative to the top-most window or the window specified by /W, like the MOUSEX and MOUSEY values passed to a window hook. See the window hook example, below and **SetWindow**.

If /C is not specified, the menu's top left corner appears at the current mouse position.

/N — Indicates that *popupStr* contains the name of a menu definition instead of containing a list of menu items.

/W=*winName* — The /C coordinates are relative to the top/left corner of the named window or subwindow. If you omit /W, /C uses the top-most window having focus.

When identifying a subwindow with *winName*, see **Subwindow Syntax** on page III-92 for details on forming the window hierarchy.

/W was added in Igor Pro 7.00.

Details

If you omit /N and /ASYN, PopupContextualMenu sets the following variables:

`V_flag=0` — User cancelled the menu without selecting an item, or there was an error such as an empty *popupStr*.

`V_flag=>=1` — 1 if the first menu item was selected, 2 for the second, etc.

`S_selection` — "" if the user cancelled or error, else the text of the selected menu item.

If you include /N and omit /ASYN, PopupContextualMenu sets the following variables in a manner similar to **GetLastUserMenuInfo**:

V_kind — The kind of menu that was selected:

V_kind	Menu Kind
0	Normal text menu item, including **Optional Menu Items** (see page IV-130) and **Multiple Menu Items** (see page IV-131).
3	"*FONT*"
6	"*LINESTYLEPOP*"
7	"*PATTERNPOP*"
8	"*MARKERPOP*"
9	"*CHARACTER*"
10	"*COLORPOP*"
13	"*COLORTABLEPOP*"

See **Specialized Menu Item Definitions** on page IV-132 for details about these special user-defined menus.

V_flag — -1 if the user didn't select any item, otherwise V_flag returns a value which depends on the kind of menu the item was selected from:

V_kind	V_flag Meaning
0	Text menu item number (the first menu item is number 1).
3	Font menu item number (use S_selection, instead).
6	Line style number (0 is solid line)
7	Pattern number (1 is the first selection, a SW-NE light diagonal).
8	Marker number (1 is the first selection, the X marker).
9	Character as an integer, = char2num(S_selection). Use S_selection instead.
10	Color menu item (use V_Red, V_Green, V_Blue, and V_Alpha instead).
13	Color table list menu item (use S_selection instead).

S_selection — The menu item text, depending on the kind of menu it was selected from:

V_kind	S_selection Meaning
0	Text menu item text.
3	Font name or "default".
6	Name of the line style menu or submenu.
7	Name of the pattern menu or submenu.
8	Name of the marker menu or submenu.
9	Character as string.
10	Name of the color menu or submenu.
13	Color table name.

In the case of **Specialized Menu Item Definitions** (see page IV-132), S_selection will be the title of the menu or submenu, etc.

V_Red, V_Green, V_Blue, V_Alpha

If a user-defined color menu ("*COLORPOP*" menu item) was selected then these values hold the red, green, and blue values of the chosen color. The values range from 0 to 65535 - see **RGBA Values**.

Will be 0 if the last user-defined menu selection was not a color menu selection.

If you include /N and /ASYN, PopupContextualMenu sets the following variables:

V_flag=0	There was an error such as an empty *popupStr* or *popupStr* did not name a compiled user-defined menu.
V_flag=-1	No error. The named user menu was valid and no item was selected yet.
S_selection	""

If you include /N and omit /ASYN, PopupContextualMenu sets the following variables:

V_flag=0	There was an error such as an empty *popupStr*.
V_flag=-1	No error. *popupStr* was valid and no item was selected yet.
S_selection	""

Examples

Example 1 - *popupStr* contains a list of menu items

```
// Menu formatting example
String checked= "\\M0:!" + num2char(18) + ":"  // checkmark code
String items= "first;\M1-;"+checked+"third;"   // 2nd is divider, 3rd is checked
PopupContextualMenu items
switch( V_Flag )
   case 1:
      // do something because first item was chosen
      break;
   case 3:
      // do something because first item was chosen
      break;
endswitch
```

Example 2 - *popupStr* contains a list of menu items

```
// Window hook example
SetWindow kwTopWin hook=TableHook, hookevents=1      // mouse down events
Function TableHook(infoStr)
   String infoStr

   Variable handledEvent=0
   String event= StringByKey("EVENT",infoStr)
   strswitch(event)
      case "mousedown":
         Variable isContextualMenu= NumberByKey("MODIFIERS",infoStr) & 0x10
         if( isContextualMenu )
            Variable xpix= NumberByKey("MOUSEX",infoStr)
            Variable ypix= NumberByKey("MOUSEY",infoStr)
            PopupContextualMenu/C=(xpix,ypix) "yes;no;maybe;"
            strswitch(S_selection)
               case "yes":
                  // do something because "yes" was chosen
                  break
               case "no":
                  break
               case "maybe":
                  // do something because "maybe" was chosen
                  break
```

```
            endswitch
            handledEvent=1
        endif
    endswitch
    return handledEvent
End
```

Example 3 - popupStr contains the name of a user-defined menu

```
// User-defined contextual menu example

// dynamic menu (to keep WaveList items updated), otherwise not required.
// contextualmenu keyword is required, and implies /Q for all menu items.
//
// NOTE: Actions here are accomplished by the menu definition's
// execution text, such as DoSomethingWithColor.
// See Example 4 for another approach.
//
Menu "ForContext", contextualmenu, dynamic
    "Hello", Beep
    Submenu "Color"
        "*COLORPOP*", DoSomethingWithColor()
    End
    Submenu "Waves"
        WaveList("*",";",""), /Q, DoSomethingWithWave()
    End
End

Function DoSomethingWithColor()
    GetLastUserMenuInfo
    Print V_Red, V_Green, V_Blue, V_Alpha
End

Function DoSomethingWithWave()
    GetLastUserMenuInfo
    WAVE w = $S_value
    Print "User selected "+GetWavesDataFolder(w,2)
End

// Use this code in a function or macro:
PopupContextualMenu/N "ForContext"
if( V_flag < 0 )
    Print "User did not select anything"
endif
```

Example 4 - popupStr contains the name of a user-defined menu

```
// User-defined contextual menu example

Menu "JustColorPop", contextualmenu
    "*COLORPOP*(65535,0,0)", ;// initially red, empty execution text
End

// Use this code in a function or macro
PopupContextualMenu/C=(xpix, ypix)/N "JustColorPop"
if( V_flag < 0 )
    Print "User did not select anything"
else
    Print V_Red, V_Green, V_Blue, V_Alpha
endif
```

Example 5 - popupStr contains a list of menu items, asynchronous popup result

```
Function YourFunction()
    // Use this code in a function or macro:
    PopupContextualMenu/ASYN=Callback "first;second;third;"
    (YourFunction continues...)
End

// Routine called when/if popup menu item is selected. selectedItem=1 is the first item.
Function Callback(String list, String selectedText, Variable selectedItem)
```