```
   SetWindow MyPanel, hook(myHook)=MyHookFunction
End
```

If you run the MakePanelWithHook function on the command line, you see nothing because the panel is hidden. Now select Windows→Other Windows→MyPanel. The following is printed in the history:

```
Show event
Moved while hidden
Resized while hidden
```

The hook function received the Move and Resize events *after* the Show event, but before the window actually became visible.

Hook Functions for Exterior Subwindows

A regular subwindow lives inside a host window and receives events through a hook function attached to its host window.

An exterior subwindow is different because, although it is a subwindow (it is controlled by a host window), unlike a regular subwindow, it has its own actual window and therefore you can attach a hook function directly to it. A hook function attached to the root window does not receive events for an exterior subwindow. To handle events for an exterior subwindow, you must attach a hook function to the exterior subwindow itself. For example:

```
// Make a panel
NewPanel/N=RootPanel

// Make an exterior subwindow attached to RootPanel
NewPanel/HOST=RootPanel/EXT=0

// Attach a hook function to the exterior subwindow
SetWindow RootPanel#P0 hook(myhook)=MyHookFunction
```

Unnamed Window Hook Functions

Unnamed window hook functions are supported for backward compatibility only. New code should use named window hook functions. See **Named Window Hook Functions** on page IV-295.

Each window can have one unnamed hook function. You designate a function as the unnamed window hook function using the **SetWindow** operation with the hook keyword.

The unnamed hook function is called when various window events take place. The reason for the hook function call is stored as an event code in the hook function's infoStr parameter.

The hook function is not called during experiment creation or load time so as to prevent the hook function from failing because the experiment is not fully recreated.

The hook function has the following syntax:

```
Function procName(infoStr)
   String infoStr
   String event= StringByKey("EVENT",infoStr)
   …
   return statusCode    // 0 if nothing done, else 1
End
```

infoStr is a string containing a semicolon-separated list of ***key:value*** pairs:

Key	Value
EVENT	*eventKey* See list of *eventKey* values below.
HCSPEC	Absolute path of the window or subwindow. See **Subwindow Command Concepts** on page III-92.
MODIFIERS	Bit flags as follows: Bit 0: Set if mouse button is down. Bit 1: Set if Shift is down. Bit 2: Set if Option (*Macintosh*) or Alt (*Windows*) is down. Bit 3: Set if Command (*Macintosh*) or Ctrl (*Windows*) is down. Bit 4: Contextual menu click: right-click or Control-click (*Macintosh*), or right-click (*Windows*). See **Setting Bit Parameters** on page IV-12 for details about bit settings.
OLDWINDOW	Previous name of the window or subwindow (for renamed event). Not the old absolute path *hcSpec*, just the name. WINDOW and HCSPEC contain the new name and new *hcSpec*.
WINDOW	Name of the window.

The value accompanying the EVENT keyword is one of the following:

eventKey	**Meaning**
activate	Window has just been activated.
copy	Copy menu item has been selected.
cursormoved	A graph cursor was moved. This event is sent only if bit 2 of the **SetWindow** operation hookevents flag is set.
deactivate	Window has just been deactivated.
enablemenu	Menus are being built and enabled.
hide	Window or subwindows about to be hidden.
hideInfo	The window info panel or window has just been hidden by **HideInfo**.
hideTools	The window tool palette or window has just been hidden by **HideTools**.
kill	Window is being killed. As of Igor Pro version 7, returning 2 as the hook function result no longer prevents Igor from killing the window. Use the killVote event instead.
killVote	Window is about to be killed. Return 2 to prevent that, otherwise return 0. See **Window Hook Deactivate and Kill Events** on page IV-303.
menu	A built-in menu item has been selected.
modified	A modification to the window has been made. See **Modified Events** on page IV-299.
mousedown	Mouse button was clicked. This event is sent only if bit 0 of the **SetWindow** operation hookevents flag is set.

eventKey	Meaning
mousemoved	The mouse moved. This event is sent only if bit 1 of the **SetWindow** operation hookevents flag is set.
mouseup	Mouse button was released. This event is sent only if bit 0 of the **SetWindow** operation hookevents flag is set.
`moved`	Window has just been moved.
`renamed`	Window has just been renamed. The previous name is available under the OLDWINDOW key.
`resize`	Window has just been resized.
`show`	Window or subwindow is about to be unhidden.
`showInfo`	The window info panel or window has just been shown by **ShowInfo**.
`showTools`	The window tool palette or window has just been shown by **ShowTools**.
`subwindowKill`	One of the window's subwindows is about to be killed.

If mouse events are enabled then the following key:value pairs will also be present in `infoStr`:

Key	Value
`MOUSEX`	X coordinate in pixels of the mouse.
`MOUSEY`	Y coordinate in pixels of the mouse.
`TICKS`	Time event happened.

Note that a mouseup event may or may not correspond to a previous mousedown. If the user clicks in the window, drags out and releases the button then the mouseup event will be missing. If the user clicks in another window, drags into this one and then releases then a mouseup will be sent that had no previous mousedown.

In the case of mousedown or mousemoved messages, a nonzero return value will skip normal processing of the message. This is most useful with mousedown.

The `cursormoved` event is not reported if Option (*Macintosh*) or Alt (*Windows*) is held down.

If the `cursormoved` event is enabled then the following key:value pairs will also be present in `infoStr`:

Key	Value
`CURSOR`	Name of the cursor that moved (A through J).
`TNAME`	Name of the trace the cursor is attached to (invalid if `ISFREE=1`).
`ISFREE`	1 if the cursor is "free" (not attached to a trace), 0 if it is attached to a trace or image.
`POINT`	Point number of the trace if not a free cursor. If the cursor is attached to an image, value is the row number of the image. If a free cursor, value is the fraction of the plot width, 0 being the left edge of the plot area, and 1 being the right edge.
`YPOINT`	Column number if the cursor is attached to an image, NaN if attached to a trace. If a free cursor, value is the fraction of the plot height, 0 being the top edge, and 1 being the bottom edge.