See Also
IgorBeforeNewHook and **SetIgorHook**.

Window User Data

The window user data feature provides a way for packages that create or manage windows to store per-window settings. You can store arbitrary data with a window using the userdata keyword with the **SetWindow**.

Each window has a primary, unnamed user data that is used by default.

You can also store an unlimited number of different user data strings by specifying a name for each different user data string. The name can be any legal Igor name. It should be distinct to prevent clashes between packages.

Packages should use named user data.

You can retrieve information from the default user data using the **GetWindow**. To retrieve named user data, you must use the **GetUserData**.

Here is a simple example of user data using the top window:

```
SetWindow kwTopWin,userdata= "window data"
Print GetUserData("","","")
```

Although there is no size limit to how much user data you can store, it does have to be stored as part of the recreation macro for the window when experiments are saved. Consequently, huge user data can slow down experiment saving and loading.

You can also attach user data to traces in graphs using the userData keyword of the ModifyGraph operation and to controls using the userData keyword of the various control operations.

Window Hook Functions

A window hook function is a user-defined function that receives notifications of events that occur in a specific window. Your window hook function can detect and respond to events of interest. You can then allow Igor to also process the event or inform Igor that you have handled it.

This section discusses window hook functions that apply to a specific window. For information on general events hooks, see **User-Defined Hook Functions** on page IV-280.

To handle window events, you first write a window hook function and then use the **SetWindow** operation to install the hook on a particular window. This example shows how you would detect arrow key events in a particular window. To try it, paste the code below into the procedure window and then execute `DemoWindowHook()`:

```
Function MyWindowHook(s)
   STRUCT WMWinHookStruct &s

   Variable hookResult = 0 // 0 if we do not handle event, 1 if we handle it.

   switch(s.eventCode)
      case 11:              // Keyboard event
         switch (s.keycode)
            case 28:
               Print "Left arrow key pressed."
               hookResult = 1
               break
            case 29:
               Print "Right arrow key pressed."
```