```
   Variable autoAbortSecs

   Make/O jack;SetScale x,-5,5,jack
   jack= exp(-x^2)+gnoise(0.1)
   DoWindow Graph0
   if( V_Flag==0 )
      Display jack
      ShowInfo
   endif

   if (UserCursorAdjust("Graph0",autoAbortSecs) != 0)
      return -1
   endif

   if (strlen(CsrWave(A))>0 && strlen(CsrWave(B))>0)// Cursors are on trace?
      CurveFit gauss,jack[pcsr(A),pcsr(B)] /D
   endif
End
```

Alternative to PauseForUser

In normal non-modal operation, when the user invokes a user-defined function, the function executes, terminates, and returns control to the user. The user is in complete control of what happens next. With PauseForUser, the user-defined function is running the whole time and the user's control is restricted control to what PauseForUser allows.

For example, with PauseForUser, the user can not use the help system, search for text in a notebook, activate another window for copy and paste into a table, use the Data Browser, and so on. The user can do the few things PauseForUser permits.

The alternative to PauseForUser is non-modal operation. To facilitate this in a situation where you might consider using PauseForUser, you need to break the task into two or more steps, each implemented by a separate function, and provide the user with the means to invoke each function as he sees fit. The user initiates the first step which runs to completion. The user is then free to do what he wishes before invoking the second step.

To see the non-modal approach in action, execute this in Igor and follow the instructions for running the example:

```
DisplayHelpTopic "Alternatives to PauseForUser"
```

The main difference between the non-modal alternative approach and the PauseForUser approach is that, in the non-modal case, the user is free to do whatever he wants between step one (creating the graph) and step two (performing the fit).

A hidden difference is that the semi-modal operation of PauseForUser relies on tricky and potentially fragile code inside Igor while the non-modal approach uses Igor in a natural way.

PauseForUser Advanced Cursor Example

Now for something a bit more complex. Here we modify the preceding example to include a Cancel button. For this, we need to return information about which button was pressed. Although we could do this by creating a single global variable in the root data folder, we use a slightly more complex technique using a temporary data folder. This technique is especially useful for more complex panels with multiple output variables because it eliminates name conflict issues. It also allows much easier clean up because we can kill the entire data folder and everything in it with just one operation.

```
Function UserCursorAdjust(graphName)
   String graphName

   DoWindow/F $graphName   // Bring graph to front
   if (V_Flag == 0)        // Verify that graph exists
```

```
      Abort "UserCursorAdjust: No such graph."
      return -1
   endif

   NewDataFolder/O root:tmp_PauseforCursorDF
   Variable/G root:tmp_PauseforCursorDF:canceled= 0

   NewPanel/K=2 /W=(139,341,382,450) as "Pause for Cursor"
   DoWindow/C tmp_PauseforCursor          // Set to an unlikely name
   AutoPositionWindow/E/M=1/R=$graphName  // Put panel near the graph

   DrawText 21,20,"Adjust the cursors and then"
   DrawText 21,40,"Click Continue."
   Button button0,pos={80,58},size={92,20},title="Continue"
   Button button0,proc=UserCursorAdjust_ContButtonProc
   Button button1,pos={80,80},size={92,20}
   Button button1,proc=UserCursorAdjust_CancelBProc,title="Cancel"

   PauseForUser tmp_PauseforCursor,$graphName

   NVAR gCaneled= root:tmp_PauseforCursorDF:canceled
   Variable canceled= gCaneled   // Copy from global to local
                                 // before global is killed
   KillDataFolder root:tmp_PauseforCursorDF

   return canceled
End

Function UserCursorAdjust_ContButtonProc(ctrlName) : ButtonControl
   String ctrlName

   KillWindow/Z tmp_PauseforCursor  // Kill self
End

Function UserCursorAdjust_CancelBProc(ctrlName) : ButtonControl
   String ctrlName

   Variable/G root:tmp_PauseforCursorDF:canceled= 1
   KillWindow/Z tmp_PauseforCursor  // Kill self
End

Function Demo()
   Make/O jack;SetScale x,-5,5,jack
   jack= exp(-x^2)+gnoise(0.1)
   DoWindow Graph0
   if (V_Flag==0)
      Display jack
      ShowInfo
   endif
   Variable rval= UserCursorAdjust("Graph0")
   if (rval == -1)            // Graph name error?
      return -1;
   endif
   if (rval == 1)             // User canceled?
      DoAlert 0,"Canceled"
      return -1;
   endif
   CurveFit gauss,jack[pcsr(A),pcsr(B)] /D
End
```