## Background Tasks and Errors

If a background task procedure contains a bug, it will typically generate an error each time the procedure runs. Normally an error generates an error dialog. If this happened over and over again, it would prevent you from fixing the bug.

Igor handles such repeated errors as follows: The first time an error occurs during the execution of the background task procedure, Igor displays an error dialog. On subsequent errors, Igor prints an error message in the history. After printing 10 such error messages, Igor stops printing messages. When you click a control, execute a command from the command line or execute a command through a menu item, the process starts over.

If the Igor debugger is enabled and Debug on Error is turned on, Igor will break into the debugger each time an error occurs in the background task procedure. You may have to turn Debug on Error off to give you time to stop the background task. You can do this from within the debugger by right-clicking.

## Background Tasks and Dialogs

By default, a background task created by CtrlNamedBackground does not run while a dialog is displayed. You can change this behavior using the CtrlNamedBackground dialogsOK keyword.

If you allow background tasks to run while an Igor dialog is present, you should ensure that your background task does not kill anything that a dialog might depend on. It should not kill waves or variables. It should never directly modify a window (except for a status panel) and especially should never remove something from a window (such as a trace from a graph). Otherwise your background task may kill something that the dialog depends on which will cause a crash.

## Background Task Tips

Background tasks should be designed to execute quickly. They do not run in separate threads threads and they hang Igor's event processing as long as they run. For maximum responsiveness, your task procedure should take no more than a fraction of a second to run even when the period is long. If you have to perform a lengthy computation, let the user know what is going on, perhaps via a message in a status control panel.

Background tasks should never attempt to put up dialogs or directly wait for user input. If you need to get the attention of the user, you should design your system to include a status control panel with an area for messages or some other change in appearance. If you need to wait for the user, you should do so by monitoring global variables set by nonbackground code such as a button procedure in a panel.

Your task procedure should always leave the current data folder unchanged on exit.

## Background Task Example #2

Here is an example that uses many of the concepts discussed above. The task prints a message in the history area at one second intervals five times, performs a "lengthy calculation", and then waits for the user to give the go-ahead for another run.

The task does its own timing and consequently is set to run at the maximum rate (60 times per second). The task procedure, MyBGTask, tests to see if one second has elapsed since the last time it printed a message. In a real application, you might test to see if some external event has occurred.

To try the example, copy the code below to the Procedure window and execute:

```
BGDemo()
```

```
Function BGDemo()
	DoWindow/F BGDemoPanel         // bring panel to front if it exists
	if( V_Flag != 0 )
		return 0                   // panel already exists
	endif

	String dfSav= GetDataFolderDFR()         // so we can leave current DF as we found it
	NewDataFolder/O/S root:Packages
	NewDataFolder/O/S root:Packages:MyDemo  // our variables go here
```

```
    // still here if no panel, create globals if needed
    if( NumVarOrDefault("inited",0) == 0 )
        Variable/G inited= 1

        Variable/G lastRunTicks= 0 // value of ticks function last time we ran
        Variable/G runNumber= 0             // incremented each time we run
        // message displayed in panel using SetVariable...
        String/G message="Task paused. Click Start to resume."

        Variable/G running=0                // when set, we do our thing
    endif

    SetDataFolder dfSav
    NewPanel /W=(150,50,449,163)
    DoWindow/C BGDemoPanel                  // set panel name
    Button StartButton,pos={21,12},size={50,20},proc=BGStartStopProc,title="Start"
    SetVariable msg,pos={21,43},size={300,17},title=" ",frame=0
    SetVariable msg,limits={-Inf,Inf,1},value= root:Packages:MyDemo:message
End

Function MyBGTask(s)
    STRUCT WMBackgroundStruct &s

    NVAR running= root:Packages:MyDemo:running

    if( running == 0 )
        return 0                      // not running -- wait for user
    endif

    NVAR lastRunTicks= root:Packages:MyDemo:lastRunTicks

    if( (lastRunTicks+60) >= ticks )
        return 0                      // not time yet, wait
    endif

    NVAR runNumber= root:Packages:MyDemo:runNumber

    runNumber += 1

    printf "Hello from the background, #%d\r",runNumber

    if( runNumber >= 5 )
        runNumber= 0
        running= 0                    // turn ourself off after five runs

        // run again when user says to
        Button StopButton,win=BGDemoPanel,rename=StartButton,title="Start"

        // Simulate a long calculation after a run
        String/G root:Packages:MyDemo:message="Performing long calculation. Please wait."
        ControlUpdate /W=BGDemoPanel msg
        DoUpdate /W=BGDemoPanel       // Required on Macintosh for control to be redrawn

        Variable t0= ticks
        do
            if (GetKeyState(0) & 32)
                Print "Lengthy process aborted by Escape key"
                break
            endif
        while( ticks < (t0+60*3) )    // delay for 3 seconds

        String/G root:Packages:MyDemo:message="Task paused. Click Start to resume."
    endif

    lastRunTicks= ticks

    return 0
End

Function BGStartStopProc(ctrlName) : ButtonControl
    String ctrlName

    NVAR running= root:Packages:MyDemo:running
    if( CmpStr(ctrlName,"StartButton") == 0 )
        running= 1
        Button $ctrlName,rename=StopButton,title="Stop"
        String/G root:Packages:MyDemo:message=""
        CtrlNamedBackground MyBGTask, proc=MyBGTask, period=1, start
    endif
```