Updates During Function Execution

An update is an action that Igor performs which consists of:

- Reexecuting formulas for dependent objects whose antecedents have changed (see Chapter IV-9, **Dependencies**)
- Redrawing graphs, tables and Gizmo plots which display waves that have changed
- Redrawing page layouts containing graphs, tables, Gizmo plots, or annotations that have changed
- Redrawing windows that have been uncovered

When no procedure is executing, Igor continually checks whether an update is needed and does an update if necessary.

When a user-defined function is executing, Igor does no automatic updates at all. You can force an update by calling the **DoUpdate** operation (see page V-168). Call DoUpdate if you don't want to wait for the next automatic update which will occur when function execution finishes.

Aborting Functions

There are two ways to prematurely stop procedure execution: a user abort or a programmed abort. Both stop execution of all procedures, no matter how deeply nested.

You can abort function execution by pressing the **User Abort Key Combinations** or by clicking the Abort button in the status bar. You may need to press the keys down for a while because Igor looks at the keyboard periodically and if you don't press the keys long enough, Igor will not see them.

A user abort does not directly return. Instead it sets a flag that stops loops from looping and then returns using the normal calling chain. For this reason some code will still be executed after an abort but execution should stop quickly. This behavior releases any temporary memory allocations made during execution.

A **programmed abort** occurs during procedure execution according to conditions set by the programmer.

The simplest programmed abort occurs when the **Abort** operation (see page V-18) is executed. Here is an example:

```
if (numCells > 10)
   Abort "Too many cells! Quitting."
endif
// code here doesn't execute if numCells > 10
```

Other programmed aborts can be triggered using the AbortOnRTE and AbortOnValue flow control keywords. The try-catch-endtry flow control construct can be used for catching and testing for aborts. See **Flow Control for Aborts** on page IV-48 for more details.

If your code uses Igor pre-emptive threads that can run for a long time, you should detect aborts and make your threads quit. See **Aborting Threads** on page IV-337 for details.

Igor Pro 7 Programming Extensions

The following programming features were added in Igor Pro 7. They bring Igor programming closer to the C language. If you use these features, your procedures will not compile in Igor Pro 6.

Double and Complex Variable Types

You can use `double` and `complex` in functions as aliases for `Variable` and `Variable/C`:

```
Function foo()
   double d = 4
   complex c = cmplx(2,3)
```