Aborting Macros

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

You can abort macro 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 when the Abort operation is executed. Here is an example:

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

Converting Macros to Functions

If you have old Igor procedures written as macros, as you have occasion to revisit them, you can consider converting them to functions. In most cases, this is a good idea. An exception is if the macros are so complex that there would be a substantial risk of introducing bugs. In this case, it is better to leave things as they are.

If you decide to do the conversion, here is a checklist that you can use in the process.

1. Back up the old version of the procedures.
2. Change the defining keyword from Macro or Proc to Function.
3. If the macro contained Prompt statements, then it was used to generate a missing parameter dialog. Change it to generate a simple input dialog as follows:
   a. Remove the parameters from the parameter list. The old parameter declarations now become local variable declarations.
   b. Make sure that the local variable for each prompt statement is initialized to some value.
   c. Add a DoPrompt statement after all of the Prompt statements.
   d. Add a test on V_Flag after the DoPrompt statement to see if the user canceled.
4. Look for statements that access global variables or strings and create NVAR and SVAR references for them.
5. Look for any waves used in assignment statements and create WAVE references for them.
6. Compile procedures. If you get an error, fix it and repeat step 6.