AbortOnRTE Keyword

The AbortOnRTE (abort on runtime error) keyword can be used to raise an abort whenever a runtime error occurs. Use AbortOnRTE immediately after a command or sequence of commands for which you wish to catch a runtime error.

See **AbortOnRTE** on page V-18 for further details. For a usage example see **Simple try-catch-endtry Example** on page IV-49.

AbortOnValue Keyword

The AbortOnValue keyword can be used to abort function execution when a specified abort condition is satisfied. When AbortOnValue triggers an abort, it can also return a numeric abort code that you can use to characterize the cause.

See **AbortOnValue** on page V-19 for further details. For a usage example see **Simple try-catch-endtry Example** on page IV-49.

try-catch-endtry Flow Control

The try-catch-endtry flow control construct has the following syntax:

```
try
   <code>
catch
   <code to handle abort>
endtry
```

The try-catch-endtry flow control construct can be used to catch and respond to abnormal conditions in user functions. Code within the try and catch keywords tests for abnormal conditions and calls Abort, AbortOnRTE or AbortOnValue if appropriate. An abort also occurs if the user clicks the Abort button in the status bar or presses the **User Abort Key Combinations**.

When an abort occurs, execution immediately jumps to the first statement after the catch keyword. This catch code handles the abnormal condition, and execution then falls through to the first statement after endtry.

If no abort occurs, then the code between catch and endtry is skipped, and execution then falls through to the first statement after endtry.

When an abort occurs within the try-catch area, Igor stores a numeric code in the V_AbortCode variable. The catch code can use V_AbortCode to determine what caused the abort.

See **try-catch-endtry** on page V-1047 for further details.

Simple try-catch-endtry Example

We start with a simple example that illustrates the most common try-catch use. It catches and handles a runtime error.

```
Function DemoTryCatch0(name)
   String name             // Name to use for a new wave

   try
      Make $name           // Generates error if name is illegal or in use
      AbortOnRTE           // Abort if an error occurred

      // This code executes if no error occurred
      Printf "The wave '%s' was successfully created\r", name
   catch
      // This code executes if an error occurred in the try block
      Variable err = GetRTError(1)     // Get error code and clear error
      String errMessage = GetErrMessage(err, 3)
      Printf "While executing Make, the following error occurred: %s\r", errMessage
```

```
   endtry
End
```

Copy the code to the Procedure window of a new experiment and then execute:

```
DemoTryCatch0("wave0")
```

Since wave0 does not exist in a new experiment, DemoTryCatch0 creates it.

Now execute the same command again. This time the call to Make generates a runtime error because wave0 already exists. The call to AbortOnRTE generates an abort, because of the error generated by Make. The abort causes the catch code to run. It reports the error to the user and clears it to allow execution to continue.

Without the clearing of the runtime error, via the GetRTError call, the error would cause procedure execution to halt and Igor would report the error to the user via an error dialog.

Complex try-catch-endtry Example

The following example demonstrates how aborting flow control works. Copy the code to the Procedure window and execute the `DemoTryCatch1` function with 0 to 6 as input parameters. When you execute `DemoTryCatch1(6)`, the function loops until you click the Abort button at the right end of the status bar.

```
Function DemoTryCatch1(a)
   Variable a

   Print "A"
   try
      Print "B1"
      AbortOnValue  a==1 || a==2,33
      Print "B2"
      DemoTryCatch2(a)
      Print "B3"
      try
         Print "C1"
         if( a==4 || a==5 )
            Make $""; AbortOnRTE
         endif
         Print "C2"
      catch
         Print "D1"
         // will be runtime error so pass along to outer catch
         AbortOnValue a==5, V_AbortCode
         Print "D2"
      endtry
      Print "B4"
      if( a==6 )
         do
         while(1)
      endif
      Print "B5"
   catch
      Print "Abort code= ", V_AbortCode
      if( V_AbortCode == -4 )
         Print "Runtime error= ", GetRTError(1)
      endif
      if( a==1 )
         Abort "Aborting again"
      endif
      Print "E"
   endtry
   Print "F"
End
```