Pragmas

A pragma is a statement in a procedure file that sets a compiler mode or passes other information from the programmer to Igor. The form of a pragma statement is:

```
#pragma keyword [= parameter]
```

The pragma statement must be flush against the left margin of the procedure window, with no indentation.

Igor ignores unknown pragmas such as pragmas introduced in later versions of the program.

Currently, Igor supports the following pragmas:

```
#pragma rtGlobals = value
#pragma version = versionNumber
#pragma IgorVersion = versionNumber
#pragma hide = value
#pragma ModuleName = name
#pragma IndependentModule = name
#pragma TextEncoding = textEncodingNameStr // Igor Pro 7.00
#pragma DefaultTab = {mode, widthInPoints, widthInSpaces} // Igor Pro 9.00
```

The effect of a pragma statement lasts until the end of the procedure file that contains it.

The rtGlobals Pragma

The rtGlobals pragma controls aspects of the Igor compiler and runtime error checking in user-defined functions.

Prior to Igor Pro 3, to access a global (wave or variable) from a user-defined function, the global had to already exist. Igor Pro 3 introduced "runtime lookup of globals" under which the Igor compiler did not require globals to exist at compile time but rather connected references, declared with WAVE, NVAR and SVAR statements, to globals at runtime. Igor Pro 6.20 introduced stricter compilation of wave references and runtime checking of wave index bounds.

You enable and disable these behaviors using an rtGlobals pragma. For example:

```
#pragma rtGlobals = 3   // Strict wave reference mode, runtime bounds checking
```

A given rtGlobals pragma governs just the procedure file in which it appears. The pragma must be flush left in the procedure file and is typically put at the top of the file.

The rtGlobals pragma is defined as follows:

#pragma rtGlobals=0	Specifies the old, pre-Igor Pro 3 behavior. This is no longer supported and acts like rtGlobals=1.
#pragma rtGlobals=1	Turns on runtime lookup of globals. This is the default setting if there is no rtGlobals pragma in a given procedure file.
#pragma rtGlobals=2	Forces old experiments out of compatibility mode. This is superceded by rtGlobals=3. It is described under **Legacy Code Issues** on page IV-113.
#pragma rtGlobals=3	Turns on runtime lookup of globals, strict wave references and runtime checking of wave index bounds. Requires Igor Pro 6.2 or later.

rtGlobals=3 is recommended.

Under strict wave references (rtGlobals=3), you must create a wave reference for any use of a wave. Without strict wave references (rtGlobals=1), you do not need to create a wave reference unless the wave is used in an assignment statement. For example:

```
Function Test()
   jack = 0       // Error under rtGlobals=1 and under rtGlobals=3
   Display jack   // OK under rtGlobals=1, error under rtGlobals=3

   Wave jack      // jack is now a wave reference rather than a bare name
   Display jack   // OK under rtGlobals=1 and under rtGlobals=3
End
```

Even with rtGlobals=3, this compiles without error:

```
Function Test()
   // Make creates an automatic wave reference when used with a simple name
   Make jack
   Display jack   // OK under rtGlobals=1 and rtGlobals=3
End
```

See **Automatic Creation of WAVE References** on page IV-72 for details.

Under runtime wave index checking (rtGlobals=3), Igor reports an error if a wave index is out-of-bounds:

```
Function Test()
   Make/O/N=5 jack = 0  // Creates automatic wave reference

   jack[4] = 123        // OK
   jack[5] = 234        // Runtime error under rtGlobals=3.
                        // Clipped under rtGlobals=1.

   Variable index = 5
   jack[index] = 234    // Runtime error under rtGlobals=3.
                        // Clipped under rtGlobals=1.

   // Create and use a dimension label for point 4 of jack
   SetDimLabel 0,4,four,jack
   jack[%four] = 234    // OK

   // Use a non-existent dimension label.
   jack[%three] = 345   // Runtime error under rtGlobals=3.
                        // Clipped under rtGlobals=1.
   // Under rtGlobals=1, this statement writes to point 0 of jack.
End
```

In Igor Pro 9.00 and later, Igor does runtime wave type checking which reports an error if a wave's type does not match the wave reference's type. In this example, we assign a numeric wave to a text wave reference:

```
#pragma rtGlobals=3
Function DemoRuntimeWaveTypeCheck()
   // WRONG: Assigning a numeric wave to a text wave reference
   WAVE/T tw = NewFreeWave(4,3)

   // Error is detected when the incorrect wave reference is used:
   // "An attempt was made to treat a numeric wave as if it were a text wave"
   String str = tw[0]
End
```

You can disable runtime wave type checking by executing:

```
SetIgorOption ReportWaveTypeMismatch = 0  // Turn runtime type checking off
```

You can ignore it for a specific line of code using GetRTError on the same line as the error:

```
String str = w[0]; int err = GetRTError(1)
```