Expression Evaluation

An expression is a combination of literal values, variable references, wave references, function calls, parentheses and operators. Expressions appear on the right-hand side of assignment statements and as parameters in commands. For example:

```
Variable v = ((123 + someVariable) * someWave[3]) / SomeFunction()
```

In most cases Igor evaluates expressions using double-precision floating point. However, if the destination is an integer type, then Igor uses integer calculations.

Integer Expressions

Prior to Igor Pro 7, all calculations were performed in double-precision floating point. Double-precision can represent integers of up to 53 bits precisely. Integers larger than 53 bits are represented approximately in double-precision.

Igor Pro 7 or later can perform integer calculations instead of floating point. You trigger this by assigning a value to a local variable declared using the integer types int, int64 and uint64. Calculations are done using 32 bits or 64 bits depending on the type of the integer.

When an integer type variable is the destination in an expression in a function, the right-hand side is compiled using integer math. This avoids the limitation of 53 bits for double precision and may also provide a speed advantage. If you use a function such as sin that is inherently floating point, it is calculated as a double and then converted to an integer. You should avoid using anything that causes a double-to-integer conversion when the destination is a 64-bit integer.

This example shows various ways to use integer expressions:

```
Function Example()
   int a = 0x101              // 0x introduces a hexadecimal literal number
   int b = a<<2
   int c = b & 0x400
   printf "a=%x, b=%x, c=%x\r",a, b, c
End
```

This prints

```
a=101, b=404, c=400
```

To set bits in an integer, use the left-shift operator, <<. For example to set bit 60 in a 64-bit integer, use 1<<60. This is the integer equivalent of 2^60, but you can not use ^ because exponentiation is not supported in integer expressions.

To print all the bits in a 64-bit integer, use Printf with the %x or %d conversion specification. You can also use a Print command as long as the compiler can clearly see the number is an integer from the first symbol. For example:

```
Function Example()
   int64 i = 1<<60   // 1152921504606846976 (decimal), 1000000000000000 (hex)
   Printf "i = %0.16X\r", i
   Printf "i = %d\r", i
   Print "i =", i
   Print "i =", 0+i  // First symbol is not an integer variable or wave
End
```

This prints:

```
  i = 1000000000000000
  i = 1152921504606846976
  i = 1152921504606846976
  i = 1.15292e+18
```