&=

[Compound Operators]

Description

The compound bitwise AND operator &= is often used with a variable and a constant to force particular bits in a variable to the LOW state (to 0). This is often referred to in programming guides as "clearing" or "resetting" bits.

A review of the Bitwise AND & operator:

0 0 1 1 operand1

0 1 0 1 operand2

----------

0 0 0 1 (operand1 & operand2) - returned result

Syntax

x &= y; // equivalent to x = x & y;

Parameters

x: variable. Allowed data types: char, int, long.  
y: variable or constant. Allowed data types: char, int, long.

Example Code

Bits that are "bitwise ANDed" with 0 are cleared to 0 so, if myByte is a byte variable,

myByte & B00000000 = 0;

Bits that are "bitwise ANDed" with 1 are unchanged so,

myByte & B11111111 = myByte;

Notes and Warnings

Because we are dealing with bits in a bitwise operator - it is convenient to use the binary formatter with constants. The numbers are still the same value in other representations, they are just not as easy to understand. Also, B00000000 is shown for clarity, but zero in any number format is zero (hmmm something philosophical there?)

Consequently - to clear (set to zero) bits 0 & 1 of a variable, while leaving the rest of the variable unchanged, use the compound bitwise AND operator (&=) with the constant B11111100

1 0 1 0 1 0 1 0 variable

1 1 1 1 1 1 0 0 mask

----------------------

1 0 1 0 1 0 0 0

bits unchanged

bits cleared

Here is the same representation with the variable’s bits replaced with the symbol x

x x x x x x x x variable

1 1 1 1 1 1 0 0 mask

----------------------

x x x x x x 0 0

bits unchanged

bits cleared

So if:

myByte = B10101010;

myByte &= B11111100; // results in B10101000