&

[Bitwise Operators]

Description

The bitwise AND operator in C++ is a single ampersand &, used between two other integer expressions. Bitwise AND operates on each bit position of the surrounding expressions independently, according to this rule: if both input bits are 1, the resulting output is 1, otherwise the output is 0.

Another way of expressing this is:

0 0 1 1 operand1

0 1 0 1 operand2

----------

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

In Arduino, the type int is a 16-bit value, so using & between two int expressions causes 16 simultaneous AND operations to occur.

Example Code

In a code fragment like:

int a = 92; // in binary: 0000000001011100

int b = 101; // in binary: 0000000001100101

int c = a & b; // result: 0000000001000100, or 68 in decimal.

Each of the 16 bits in a and b are processed by using the bitwise AND, and all 16 resulting bits are stored in c, resulting in the value 01000100 in binary, which is 68 in decimal.

One of the most common uses of bitwise AND is to select a particular bit (or bits) from an integer value, often called masking. See below for an example (AVR architecture specific).

PORTD = PORTD & B00000011; // clear out bits 2 - 7, leave pins PD0 and PD1 untouched (xx & 11 == xx)