



Understanding HEX and BIN Code

With Xbox-Scene growing so rapidly, floods of noobs will want to progress their skills and knowledge ☺ As a warning programming and interpreting different types of code is not for everyone. But for those of you who would like to progress and learn how to do things themselves, understanding and interpreting code can be beneficial for game hacking, map editing, and item injection and reconstruction. It is very quick and easy to learn.

One form of code (notational system) is hex. Hex editing can change just about any feature in many games. (A classic example being Halo).

Just to clarify, this tutorial isn't a how-to on how to hex edit Halo, there are many posts in the forums for that. I'm sure that leftyfb, shanafan, and Shadowcat wouldn't mind helping you with that. This is just to give you an understanding of different types of code, how to interpret and get values from the code. Hopefully this will help you when running into hex during your game hacking or what not.

Binary Code

Unlike our decimal system, the binary system only uses two values, 1 and 0.

You have probably seen binary code somewhere before,

An example, 11010011.

Each digit you have is referred to as a bit. The example above has 8 bits. Binary code can have any number of bits, for example, computers usually run on 32 bit patterns. Unlike reading regular numerical values from left to right, you read binary code from left to right. (The bolded bit would be read first).

Each bit is expressed in powers of two.

So,	Bit Position	→	8	7	6	5	4	3	2	1
	Power of 2	→	2 ⁷	2 ⁶	2 ⁵	2 ⁴	2 ³	2 ²	2 ¹	2 ⁰
	Corresponding Value	→	128	64	32	16	8	4	2	1

(In Our Decimal System)

As you can see as you increase in bit position, the next power of 2 will just double the previous corresponding value.

So whenever you see a 1 in a position, you just add it corresponding value to the sum.

An example, 1101 would mean 8+4+1, or 8+4+0+1, which equals to 13.

Binary is just that simple.

NOTE For those of you who would like to learn something called Two's Complement, which allows for negative numbers, pm me for those instructions.

I am trying to keep it basic, since breaking hex code doesn't really require Two's Complement.

Now for the great Halo code,

Hex (Hexadecimal) Code

I showed you how to do binary code just to show you hex. In a nutshell, hex is just a way to simplify binary code. Hex breaks up every four bits and gives it a corresponding digit. Just as some FYI hex uses the power of 16.

Instead of this: 101011111110110, you now have this, AFF6

0 (Zero) through 9 represents the first ten values

A represents the value 10

B represents the value 11

C represents the value 12

D represents the value 13



E represents the value 14
F represents the value 15

You only have to worry up to F since the highest value in binary, 1111 equals 15.

NOTE There are a exceptions to the above rule, such as the binary 10000 is represented as 10 in hex, and so on, pm me for a complete list up to 20 if you need it. But for the average coding this is not necessary.

An example of changing binary(s) to hex is,
1101 0101 (the underlining is given just to show separation)
then getting each value of each cluster of four bits: 13 5
which corresponds in hex to: D 5

Another example would be:

11011010 00110110

can be broken like this : 1101 1010 0011 0110, just to see each cluster of four bits better,
then getting each value of each cluster of four bits: 13 10 3 6,
which corresponds in hex to: DA 36

*As a side note there can be more that two cluster of four bits in each set of binary, so in actuality you could see this, 2A8E 466F, it can get vary complex.

You can also take the hex code and break it back down into binary code. I hope this helps you better understand what hex actually is, and what it means in terms of values.

Tutorial By: [crazyj0202](#)