Controlling The Real World With Computers

Data lines, bits, nibbles, bytes, words, binary and HEX

Data inside a computer, as well as on the board used in this tutorial, is exchanged among the various components by means of metallic conductors called **data lines**. A group of data lines is called a **data buss**. Each data line carries a unit of data called a **bit**. A bit can be on or off. **On** is typically considered to be **5 volts**, and **off** is considered to be **0 volts**, although modern systems often use lower **on** voltages to decrease power consumption.

Data can be represented on paper as a series of ones and zeros. A one means a bit is on, and a zero means it is off. A group of 8 bits is called a **byte**. A byte with a value of 0 would be represented as 00000000. Non-zero bytes can be any combination of 1s and 0s. 01100010 will be used as an example here. In the C language, a byte is called a character and is abbreviated **char**.

When data is represented as a series of ones and zeros, it is said to be a **binary** representation, or to have a **base of 2** because it uses 2 digits.

The **left-end bit** of a number represented in binary is called the **most significant bit**, abbreviated **msb**, and the **right-end bit** is called the **least significant bit**, abbreviated **lsb**.

A little review might be helpful to those who are a little rusty on raising a number to a power. No high math here -- to raise a number to a power just write it down the exponent number of times and multiply. The exponent is the power to which a number is raised. One way to recognize an exponent is by the fact that it is often **raised** when written:

$5^2 = 5 * 5 = 25$

$2^3 = 2 * 2 * 2 = 8$

$4^4 = 4 * 4 * 4 * 4 = 256$

Each bit position has a **weight**. For all numbering systems I am aware of (the mathematicians probably know of others), the right, least-significant position is known as the 1's place. There, the weight is equal to the base raised to the power of 0. Any number raised to the power of 0 is equal to 1.

The exponent is increased by 1 with each move to the left. Thus, the second place from the right has a weight equal to the base raised to the power of 1. Any number raised to the power of 1 is equal to itself. We were taught in grade school that the second place from the right is the 10's place. That's because we were using a base of 10 and we were raising it to the power of 1. Since a base of 2 is used in binary, the second place from the right has a weight of 2 because it is 2 raised to the power of 1. The next weight is $2^2 = 4$, then $2^3 = 8$ and so on.

The exponents are often used to designate a bit in a binary number. **Bit 0** is on the **right end** of the byte and **bit 7** is on the **left end**. Bit 0 is the lsb and bit 7 is the msb. Data bits are often abbreviated using the

letter D -- D0, D1, D2, etc.

Bit	D7	D6	D5	D4	D3	D2	D1	D0
Base^{exponent}	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Weights	128	64	32	16	8	4	2	1

The example binary number above was 01100010. To figure out what the decimal value is, simply add the weights for the bits that are turned on. In this case, bits 6, 5 and 1 are on. The total of their weights equals 64 + 32 + 2 = 98.

A more general description of the procedure is to multiply the position weights by the values at the positions, then add them up. The example 01100010 would be:
(0 * 128) + (1 * 64) + (1 * 32) + (0 * 16) + (0 * 8) + (0 * 4) + (1 * 2) + (0 * 1) = 98.

A common way of representing numbers in a C program is to use **hexadecimal notation**, or **HEX**. It uses a base of 16. Break a byte into two groups of 4 bits each: **nnnn nnnn**. Each group is called a **nibble**. A nibble with all low bits, 0000, is equal to 0. With all of its bits turned on, 1111, a nibble has a value of 15 (8 + 4 + 2 + 1). Thus, we are dealing with the 16 values from 0 through 15, and a base of 16.

Hexadecimal notation is simple. Just use digits for 0 through 9, and A through F for 10 through 15. The following table shows all of the combinations.

Binary	Decimal	Hexadecimal	Binary	Decimal	Hexadecimal
0000	00	0	1000	08	8
0001	01	1	1001	09	9
0010	02	2	1010	10	A
0011	03	3	1011	11	B
0100	04	4	1100	12	C
0101	05	5	1101	13	D
0110	06	6	1110	14	E
0111	07	7	1111	15	F

The right nibble of a byte is the least significant nibble. It's the 1's place because it's 16^0. Next is the 16's place because it's 16^1, then $16^2 = 256$, and so on. To get the decimal value, take the value of the nibbles, multiply by the position weight values and add them up. Thus, the HEX value 9B = (9 * 16) + (11 * 1) = 155.

To show a number is hexadecimal in the C language, prefix it with **0x**. The above would be represented as 0x9B or 0x9b. This particular notation is not case-sensitive, although many things in C are.

The following shows the byte table again, but this time with the weights also expressed in hexadecimal notation, as often seen in C operations.

Bit	D7	D6	D5	D4	D3	D2	D1	D0
Base^{exponent}	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Weights	128	64	32	16	8	4	2	1
HEX Weights	0x80	0x40	0x20	0x10	0x08	0x04	0x02	0x01

A **word** is usually 16 bits, D0 through D15. A table with the bit names and their relationship to the binary base of 2 is below.

Bit	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
Base^{exponent}	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

The following two tables show the bits with their HEX weights.

Bit	D15	D14	D13	D12	D11	D10	D9	D8
HEX Weights	0x8000	0x4000	0x2000	0x1000	0x0800	0x0400	0x0200	0x0100

Bit	D7	D6	D5	D4	D3	D2	D1	D0
HEX Weights	0x0080	0x0040	0x0020	0x0010	0x0008	0x0004	0x0002	0x0001

A word can be broken up into 4 nibbles. It can be represented by showing its 4 nibbles as a 4-place hexadecimal number. For example, the decimal number 19070 can be represented as the hexadecimal number 0x4A7E.
$0x4A7E = (4 * 16^3) + (10 * 16^2) + (7 * 16^1) + (14 * 16^0)$
$= (4 * 4096) + (10 * 256) + (7 * 16) + (14 * 1)$
$= 19070.$

In the C language, a word is most often called an **integer**, abbreviated **int**. An integer can be used to represent numbers that range from negative to positive values, or numbers that have only positive values. In other words, an integer can be **signed** or **unsigned**. A signed integer can have either positive or negative values. An unsigned integer can only be positive. An unsigned 16-bit integer can have values from 0 through 65535. It is often abbreviated simply as **unsigned**.

Bit 15 is used as a sign bit for signed integers. If it is on, the number is negative. If it is off, it is positive. Positive values can range from 0 to 32767. Negative values can range from -1 to -32768. Some examples are shown below. Notice that the signed version is equal to -1 * (65536 - unsigned version). For example, to get the signed number from the unsigned value 49151,
signed = -1 * (65536 - 49151) = -16385.

HEX	8000	BFFF	FFFE	FFFF	0000	3FFF	7FFE	7FFF
Signed	-32768	-16385	-0002	-0001	00000	16383	32766	32767
Unsigned	32768	49151	65534	65535	00000	16383	32766	32767

A **long** word is generally considered to be 4 bytes or 32 bits. A long is used for very large numbers. Longs can also be signed or unsigned. Signed longs have a range from -2,147,483,648 to 2,147,483,647. The maximum unsigned value is 0xFFFFFFFF = 4,294,967,295. The minimum unsigned value is 0.

The following is a self-test over this section. It would be a very good idea to make sure you know the answers to all of the questions since the sections that follow will build on this one.

1) Data inside computers is exchanged among the different components by means of metal conductors called __1__. A group is called a __2__.

A) Data Buss, Weight
B) Nibble, HEX
C) Binary, Bit
D) Data Lines, Data Buss

2) If the voltage is 5 volts, the bit is on. If the bit is off, the voltage is 0 volts.

A) True
B) False

3) A group of 8 bits is a __1__ and is called a __2__ in the C programming language.

A) Data Buss, Nibble
B) Byte, Character (or char)
C) Weight, Bit
D) Unsigned, Signed

4) It is said to be a _____ when data is represented with a base of 2, because of the two digits used.

A) Integer (or int)
B) Most Significant Bit
C) Binary
D) HEX

5) The left-end bit can also be referred to as the __1__. The __2__ is the right-end bit.

A) Word, Integer (or int)
B) Character (or char), Data Buss
C) Data Lines, Long Word
D) Most Significant Bit (or msb), Least Significant Bit (or lsb)

6) In the 1's place, the _____ is equal to the base number raised to the power of 0.

A) Byte
B) Weight
C) Long Word
D) Data Buss

7) _____ is a common way of representing numbers in a C program. It uses a base of 16.

A) Data Lines
B) HEX
C) Word
D) Byte

8) If you break a byte into 2 groups of 4 bits each, then each group is a _____.

A) HEX
B) Binary
C) Nibble
D) Volt

9) A __1__ is usually 16 bits, or 2 bytes. Something with 32 bits, or 4 bytes is usually called a __2__.

A) Signed, Unsigned
B) Byte, Bit
C) Word, Long Word
D) HEX, Binary

10) The __1__ integer can be a positive or negative word, and the __2__ integer can only be a positive word.

A) Nibbles, HEX
B) Byte, Data lines
C) Bit, Integer
D) Signed, Unsigned

Answers

Home ---- ***Next:*** Boolean Logic

Problems, comments, ideas? Please e-mail me