

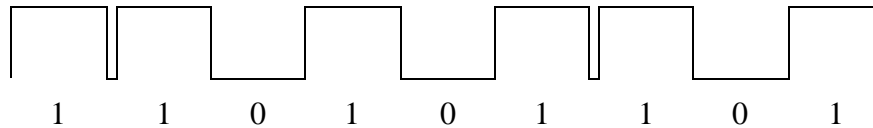
# DIGITAL COMMUNICATION

## *Self-Teaching Guide*

It is said that we live in the age of “digital” technology, but what, really, does “digital” mean?

The word “digit” refers to any of the symbols 0, 1, 2, 3, 4, 5, 6, 7, 8 or 9. We use these symbols for many purposes, but all of them can be lumped under the general heading “to represent or communicate information”.

With the advent of computers, mathematicians and engineers needed to develop a way to store and communicate information using electrical circuits. They decided to do this as simply as possible, based on the idea that an electrical switch can be either “off” or “on”. With the digit “0” being represented by the “off” position, and the digit “1” being represented by the “on” position, any combination of 0’s and 1’s can be stored or transmitted electrically. (In transmission, a “1” is represented by a pulse of electricity, and a “0” is represented by the absence of a pulse.)



The next step was to work out how to translate information into 0’s and 1’s. Numerical information is no problem; long ago mathematicians had worked out a way to convert any decimal number (a number based on *ten* and using all of the *ten* digits 0 through 9) to a binary number (a number based on *two*, using only the *two* digits 0 and 1). Let’s take a look at that.

In decimal numbers, each place represents a power of ten\*: 1, 10, 100, 1000, etc. Thus the number “231” stands for 2 hundreds plus 3 tens plus 1 one:

10,000’s	1,000’s	100’s	10’s	1’s
(10 x 10 x 10 x 10)	(10 x 10 x 10)	(10 x 10)	(10)	(10÷10)
		2	3	1

In binary numbers, each place represents a power of two.

---

\* Power of ten: The result of multiplying or dividing any number of tens.

128's (2x2x2x2x2x2x2)	64's (2x2x2x2x2x2)	32's (2x2x2x2x2)	16's (2x2x2x2)	8's (2x2x2)	4's (2x2)	2's (2)	1's (2+2)
1	1	1	0	0	1	1	1

Thus the binary number *11101001* stands for  $1 \times 128 + 1 \times 64 + 1 \times 32 + 0 \times 16 + 0 \times 8 + 1 \times 4 + 1 \times 2 + 1 \times 1$ , which is the same amount as the decimal number 231.

**Exercise A:** Change these binary numbers into decimal numbers.

128's	64's	32's	16's	8's	4's	2's	1's
1	0	0	1	0	1	0	1

128's	64's	32's	16's	8's	4's	2's	1's
0	1	1	0	1	1	0	0

From these examples you can see how *any* decimal number can be represented by 0's and 1's--though it might take a *lot* of 0's and 1's for large numbers.

But what about textual information, such as the word "CAT"? To solve this problem, computer designers have agreed on a code, in which every letter of the alphabet is represented by its own sequence of 0's and 1's. For example,

C = 01000011

A = 01000001

T = 01010100

Thus the word "CAT" can be represented by this string of 0's and 1's:

CAT = 01000011 01000001 01010100

Actually, *every* upper case and lower case symbol on a typewriter or a computer keyboard has its own digital code. Naturally it's important that all computers use the same code, and so the code below was established. It is called the *American Standard Code for Information Interchange*, and is usually referred to as ASCII (pronounced "asskey") code. Some of this code is presented below.

#### ASCII Code

A	01000001
B	01000010
C	01000011

D	01000100
E	01000101
F	01000110

G	01000111
H	01001000
I	01001001

J	01001010
K	01001011
L	01001100

M	01001101
N	01001110
O	01001111
P	01010000
Q	01010001
R	01010010
S	01010011
T	01010100
U	01010101
V	01010110
W	01010111
X	01011000
Y	01011001

Z	01011010
[	01011011
\	01011100
]	01011101
^	01011110
_	01011111
`	01100000
a	01100001
b	01100010
c	01100011
d	01100100
e	01100101
f	01100110

g	01100111
h	01101000
i	01101001
j	01101010
k	01101011
l	01101100
m	01101101
n	01101110
o	01101111
p	01110000
q	01110001
r	01110010
s	01110011

t	01110100
u	01110101
v	01110110
w	01110111
x	01111000
y	01111001
z	01111010
{	01111011
	01111100
}	01111101
~	01111110
-	01111111

**Exercise B:** Decode this ASCII word.

01000100 01001001 01000111 01001001 01010100 01000001 01001100

## Bits and Bytes

When information is represented by strings of 1's and 0's, you can measure an "amount" of information by simply counting how *many* 1's and 0's. (Of course this measures only a quantity of data, not its usefulness). For the purposes of counting, a "1" or a "0" in digital code is called a *bit*. Thus,

$$1011010 = 7 \text{ bits}$$

$$1011010010101001010 = 19 \text{ bits}$$

$$10001110 11001110 11101101 11000110 = 32 \text{ bits.}$$

Remember that any letter or other keyboard symbol can be represented by a string of *eight* 1's and 0's, which is to say, eight bits. Thus there is often a need to count bits in groups of eight, and so eight bits is defined to equal one *byte*.

$$10001110 11001110 11101101 11000110 = 32 \text{ bits} = 4 \text{ bytes}$$

You can see how this is useful if you want to figure out how much digital code it would take to represent a word, like "CAT". All you have to remember is that each letter requires one byte of digital code.

$$\text{CAT} = 3 \text{ letters} = 3 \text{ bytes}$$

**Exercise C:** Count the bits and bytes in exercise B above.

One page of typewritten text typically consists of around 3000 letters, spaces, and other symbols, which is therefore around 3000 bytes. Since most documents are longer than that, we need a unit that is bigger than "one byte", and that unit is "**kilobyte**".

But before you think you know how many bytes are in a kilobyte, remember this: although people count by tens and multiples of tens such as 100, 1000, etc., computers count by twos and multiples of 2's, such as 4, 8, 16, etc.. For this reason a kilobyte is *not* 1000 bytes, but is rather *1024* bytes, since  $1024 = 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2$ .

Armed with this knowledge, look at this directory of computer documents:

MICHTGT1.DOC	18 KB	EUROPE.DOC	11 KB
MICHTRGT.DOC	30 KB	ENGLISH.DOC	10 KB
DIANE.DOC	7 KB	STDS.DOC	21 KB
MICHTGT5.DOC	17 KB	SCHWRKLB.DOC	26 KB
MICHTGT3.DOC	29 KB	MANIFEST.DOC	7 KB
MICHTGT4.DOC	28 KB	ADVENT2.DOC	30 KB
PULLMAN.DOC	17 KB	AMERICA.DOC	11 KB
EUROPE.DOC	13 KB	LABEL.DOC	14 KB
MICHLOG.DOC	29 KB	LIFE.DOC	14 KB
EUROPACK.DOC	25 KB	LABEL2.DOC	9 KB
CRICKET.DOC	7 KB	ACTIONS.DOC	16 KB

You can see that these documents range from around 7 Kilobytes ( $7 \times 1024 = 7168$  bytes) to around 30 Kilobytes ( $30 \times 1,024 = 30,720$  bytes), or roughly 7000 characters (a little over two full pages) to 30,000 characters (around 10 full pages).

All of these documents have to be kept somewhere, and most often that's a place called a "hard disk".\* There are many hard disks available, but it might help to know that there are just a few features of a hard disk that are especially important: how reliable it is, how much data it will hold, and how fast you can write data onto it or read data from it.

The amount of data a hard disk will hold is measured in "megabytes". But, as we saw with "kilobyte", "mega" doesn't mean a million, it means *1,024* kilobytes, or 1,048,576 bytes.

**Exercise D:** Show that you could put 128,000 documents of size 4 kilobytes on a 500 megabyte computer hard disk.

If you can put 128,000 documents on a 500 megabyte hard disk, you might wonder why anyone would ever need a hard disk with more room than that. But these days computers store other kinds of information as well: pictures and sounds.

## Pictures

---

\* The very first kinds of disks used to store digital data were rather flimsy and came to be called "floppy disks" or just "floppies". When a new kind of disk was created that stored data on a hard surface, it was called a "hard disk". Hard disks could store much more data than floppies, but couldn't be carried around easily, so they were and are used inside computers. Nowadays the disks that you do carry around are enclosed in hard plastic, so they don't flop any more, but they still known as "floppies".

To understand how pictures are stored as 0's and 1's, you need to know a little about how a computer screen works. The screen is actually divided into thousands of tiny regions, each of which can be a particular color. Each tiny region is called a *pixel* (a word which means *picture element*). What the computer stores as a "picture" is actually a batch of instructions that specifies exactly how each pixel is to be colored.

How is this color specified? You probably remember from science or art class that any color can be made by combining one of three basic colors. The basic colors that are used on computer screens are blue, red and green. By mixing different amounts of these basic colors, you can produce any color in the spectrum.

Now let's think about this in digital terms. Suppose you could choose from two intensities of blue: "0" or "1". Similarly you could choose a "0" or "1" intensity of red, and a "0" or "1" intensity of green. You could combine these in several ways:

Blue	Red	Green	
0	0	0	no color = <b>black</b>
0	0	1	<b>green</b>
0	1	0	<b>red</b>
0	1	1	red + green = <b>yellow</b>
1	0	0	<b>blue</b>
1	0	1	blue + green = <b>cyan</b>
1	1	0	blue + red = <b>magenta</b>
1	1	1	blue + red + green = <b>white</b>

In this example we used one bit of information (a "1" or a "0") to describe the intensity of each of the three basic colors, and we created a total of eight different colors.

1 bit descriptions  $\Rightarrow$  8 colors

Suppose we used two bits to specify the intensity of the basic colors?

blue		red		green	
00	none	00	none	00	none
01	low	01	low	01	low
10	medium	10	medium	10	medium
11	high	11	high	11	high

Now there are many more possible combinations; a total of 64 of them:

<b>B</b>	<b>R</b>	<b>G</b>	00	10	10	01	01	00	01	11	11
00	00	00	00	10	11	01	01	01	<b>B</b>	<b>R</b>	<b>G</b>
00	00	01	00	11	00	01	01	10	10	00	00
00	00	10	00	11	01	01	01	11	10	00	01
00	00	11	00	11	10	01	10	00	10	00	10
00	01	00	00	11	11	01	10	01	10	00	11
00	01	01	<b>B</b>	<b>R</b>	<b>G</b>	01	10	10	10	01	00
00	01	10	01	00	00	01	10	11	10	01	01
00	01	11	01	00	01	01	11	00	10	01	10
00	10	00	01	00	10	01	11	01	10	01	11
00	10	01	01	00	11	01	11	10	10	10	00

10	10	01
10	10	10
10	10	11
10	11	00
10	11	01
10	11	10
10	11	11

B	R	G
11	00	00
11	00	01
11	00	10
11	00	11
11	01	00

11	01	01
11	01	10
11	01	11
11	10	00
11	10	01
11	10	10
11	10	11

11	11	00
11	11	01
11	11	10
11	11	11

2 bit descriptions  $\Rightarrow$  64 colors

The more bits you allow yourself to use to describe the intensity of the three basic colors, the more color combinations you can produce.

**Exercise E:** Show that if three bits are used to describe the intensity of a basic color, then eight different intensities can be specified.

If you made a chart showing all of the possibilities, you would find that the eight levels of intensity made possible by three bit definitions would yield 512 possible color combinations.

3 bit descriptions  $\Rightarrow$  512 colors

These days most computers use 8 bit definitions, which allows 256 levels of intensity for each of the primary colors. The total number of color combinations then works out to be 16,777,216, which is enough for just about any picture you'd ever want to display.

8 bit descriptions  $\Rightarrow$  16,777,216 colors

Now let's start adding this all up for one full-screen picture. For each pixel on the computer screen, you will need 24 bits of information (8 bits for each of the red, blue, and green intensity levels). That's three bytes.

How many pixels are there? It varies depending on the size and quality of your computer monitor, but even the smallest and cheapest will contain 640 columns and 480 rows of pixels, for a total of  $640 \times 480 = 307,200$ . So 3 bytes per pixel  $\times$  307,200 pixels = 921,600 bytes of information to represent *one* full screen picture--almost a megabyte!

So on a computer, one picture may not be worth a thousand words, but it *is* equal to about one million letters worth of digital code.

**Exercise F:** How many bytes of information would be required to specify a picture that is rectangular in shape, 240 pixels by 320 pixels, drawing from the full possible set of over 16 million colors?

Of course not all pictures take up the whole screen, but you can start to see why 500 megabytes of hard disk space might cramp your style if you were doing much work with images. And in fact you will find that computer hard disks these days commonly run in the **gigabyte** range (1 gigabyte = 1024 megabytes).

## Speed

In this era of the Internet, many of the images you look at won't be found on your own computer at all—they are on some other computer, thousands of miles away. Once computer engineers started hooking computers together, they found they needed a way to describe how *fast* digital data could be transferred from one location to another. What they decided to use was “bits per second”, commonly abbreviated as “bps”<sup>\*\*</sup>.

If you know that your connection to another computer can transfer information at a rate of, say, 14,400 bits per second, then it's pretty easy to compute how long it would take to send an entire full-screen picture.

$$1 \text{ picture} = 921,600 \text{ bytes} = 7,372,800 \text{ bits.}$$

$$7,372,800 \text{ bits} \div 14,400 \text{ bits per second} = 512 \text{ seconds}$$

$$= 8 \text{ minutes, } 32 \text{ seconds}$$

Eight minutes is a long time to wait for a picture to appear--this is why so much work has gone into finding faster and faster ways of transmitting digital information, and currently many computers are connected to the Internet at speeds of at least 28,800 bps, or higher.

Even that speed wouldn't be fast enough, except for some very clever tricks that are used to reduce the amount of bits needed to represent a picture. These are too complex to describe here, as they depend on some rather fancy mathematics, but fortunately they allow computers to “compress” the digital information that represents a picture down to a fraction of its original size.

Which is a good thing, because people aren't satisfied with just sending and receiving pictures and words any more. They want video, with sound.

## Video and Sound

**Exercise G:** A video consists of a stream of (usually) 30 pictures per second. Ignoring the sound information, how many bits would it take to represent 1 minute of full screen (640 x 480 pixels) video? How long would it take to transmit this information at 28,800 bits per second?

The answer to the above exercise tells you that you could wait a long time to see even a very short video sent over the Internet. Again, fortunately, there are some clever schemes that make it possible to compress this information a great deal. Still, much

---

\* You will sometimes see another term also used to describe rate of transmission of digital information: *baud rate*. However, baud rate does not translate well into bits per second (believe it or not, a baud rate of 1200 could mean anything from 600 to 4800 bits per second!). So stick with bits per second and you'll be better protected from misunderstandings.

faster transmission methods are going to have to be implemented before computers can be turned into televisions.

As to sound, it's frankly too complicated a subject to try to squeeze in here. Suffice it to say that sounds are basically vibrations, and that those vibrations can be represented by digital code in much the same way that colors are.

## Summary

We have seen that, by various schemes, ways have been found to represent numbers, text, pictures, sound, and video information all in the same way: using only the digits "0" and "1". The word "**digital**", then, is used to describe either a) information made of 0's and 1's, or b) equipment which works with this kind of information.

*Why* has there been this stampede to digital? Mostly because of this one advantage: while it used to take a teletype for text, a telephone or radio for sound, and a television for pictures and video, now all you need is one piece of equipment to access all of those different types of information: a computer.

**Exercise H:** Explain and give an example of each of these concepts.

Digital

Bit

Byte

Megabyte

Hard disk

Floppy

Pixel

bps

