TI-83 Basic Home
68k TI-Basic Home
TI-Nspire Basic Page
Program Archives
Discussion Forums

Preparation

The Calculators
Why TI-Basic?
Using This Guide
TI-Basic Starter Kit
TI-Basic FAQ

Main Content

Commands
Development
Techniques
Miscellaneous

References

Resources
Downloads
Showcases
Sidebar
Token Size
Text Sprites
Binary & Hex
TI-83+ Font
Error Conditions
File Extensions
Key Codes
Abbreviations
Glossary

11269997

Binary, Hexadecimal and Octal number system

Binary, hexadecimal, and octal refer to different number systems. The one that we typically use is called decimal. These number systems refer to the number of symbols used to represent numbers. In the decimal system, we use ten different symbols: 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9. With these ten symbols, we can represent any quantity. For example, if we see a 2, then we know that there is two of something. For example, this sentence has 2 periods on the end..

When we run out of symbols, we go to the next digit placement. To represent one higher than 9, we use 10 meaning one unit of ten and zero units of one. This may seem elementary, but it is crucial to understand our default number system if you want to understand other number systems.

For example, when we consider a binary system which only uses two symbols, 0 and 1, when we run out of symbols, we need to go to the next digit placement. So, we would count in binary 0, 1, 10, 11, 100, 101, and so on.

This article will discuss the binary, hexadecimal, and octal number systems in more detail and explain their uses.

Fold

Table of Contents

How a Number System Works

Number systems are used to describe the quantity of something or represent certain information. Because of this, I can say that the word "calculator" contains ten letters. Our number system, the decimal system, uses ten symbols. Therefore, decimal is said to be **Base Ten**. By describing systems with bases, we can gain an understanding of how that particular system works.

When we count in Base Ten, we count starting with zero and going up to nine in order.

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, …

Once we reach the last symbol, we create a new placement in front of the first and count that up.

8, 9, **1**0, 11, 12, … , 19, **2**0, …

This continues when we run out of symbols for that placement. So, after 99, we go to 100.

The placement of a symbol indicates how much it is worth. Each additional placement is an additional power of 10. Consider the number of 2853. We know this number is quite large, for example, if it pertains to the number of apples in a basket. That's a lot of apples. How do we know it is large? We look at the number of digits.

Each additional placement is an additional power of 10, as stated above. Consider this chart.

10^3	**10^2**	**10^1**	**10^0**
digit	digit	digit	digit
***1000**	***100**	***10**	***1**

Each additional digit represents a higher and higher quantity. This is applicable for Base 10 as well as to other bases. Knowing this will help you understand the other bases better.

Binary

Binary is another way of saying Base Two. So, in a binary number system, there are only two symbols used to represent numbers: 0 and 1. When we count up from zero in binary, we run out of symbols much more frequently.

0, 1, …

From here, there are no more symbols. We do not go to 2 because in binary, a 2 doesn't exist. Instead, we use 10. In a binary system, 10 is equal to 2 in decimal.

We can count further.

Binary	0	1	10	11	100	101	110	111	1000	1001	1010
Decimal	0	1	2	3	4	5	6	7	8	9	10

Just like in decimal, we know that the more digits there are, the larger the number. However, in binary, we use powers of two. In the binary number 1001101, we can create a chart to find out what this really means.

2^6	**2^5**	**2^4**	**2^3**	**2^2**	**2^1**	**2^0**
1	0	0	1	1	0	1
64+0+0+8+4+0+1						
87						

Since this is base two, however, the numbers don't get quite as large as it does in decimal. Even still, a binary number with 10 digits would be larger than 1000 in decimal.

The binary system is useful in computer science and electrical engineering. Transistors operate from the binary system, and transistors are found in practically all electronic devices. A 0 means no current, and a 1 means to allow current. With various transistors turning on and off, signals and electricity is sent to do various things such as making a call or putting these letters on the screen.

Computers and electronics work with bytes or eight digit binary numbers. Each byte has encoded information that a computer is able to understand. Many bytes are stringed together to form digital data that can be stored for use later.

Octal

Octal is another number system with less symbols to use than our conventional number system. Octal is fancy for Base Eight meaning eight symbols are used to represent all the quantities. They are 0, 1, 2, 3, 4, 5, 6, and 7. When we count up one from the 7, we need a new placement to represent what we call 8 since an 8 doesn't exist in Octal. So, after 7 is 10.

Octal	0	1	2	3	4	5	6	7	10	11	12…	17	20…	30…	77	100
Decimal	0	1	2	3	4	5	6	7	8	9	10…	15	16…	24…	63	64

Just like how we used powers of ten in decimal and powers of two in binary, to determine the value of a number we will use powers of 8 since this is Base Eight. Consider the number 3623 in base eight.

8^3	8^2	8^1	8^0
3	6	2	3
1536+384+16+3			
1939			

Each additional placement to the left has more value than it did in binary. The third digit from the right in binary only represented 2^{3-1}, which is 4. In octal, that is 8^{3-1} which is 64.

Hexadecimal

The hexadecimal system is Base Sixteen. As its base implies, this number system uses sixteen symbols to represent numbers. Unlike binary and octal, hexadecimal has six additional symbols that it uses beyond the conventional ones found in decimal. But what comes after 9? 10 is not a single digit but two… Fortunately, the convention is that once additional symbols are needed beyond the normal ten, letters are to be used. So, in hexadecimal, the total list of symbols to use is 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, and F. In a digital display, the numbers B and D are lowercase.

When counting in hexadecimal, you count 0, 1, 2, and so on. However, when you reach 9, you go directly to A. Then, you count B, C, D, E, and F. But what is next? We are out of symbols! When we run out of symbols, we create a new digit placement and move on. So after F is 10. You count further until you reach 19. After 19, the next number is 1A. This goes on forever.

Hexadecimal	9	A	B	C	D	E	F	10	11…	19	1A	1B	1C…	9F
Decimal	9	10	11	12	13	14	15	16	17	25	26	27	28	15

Digits are explained as powers of 16. Consider the hexadecimal number 2DB7.

16^3	16^2	16^1	16^0
2	D	B	7
8192+3328+176+7			
11703			

As you can see, placements in hexadecimal are worth a whole lot more than in any of the other three number systems.

Conversion

It is important to know that 364 in octal is **not** equal to the normal 364. This is just like how a 10 in binary is certainly not 10 in decimal. 10 in binary (this will be written as 10_2 from now on) is equal to 2. 10_8 is equal to 8. How on earth do we know this? What is $20C.38F_{16}$, and how do we find out?

Here is why it is important to understand how the number systems work. By using our powers of the base number, it becomes possible to turn any number to decimal and from decimal to any number.

Base to Decimal

So, we know that 364_8 is not equal to the decimal 364. Then what is it? There is a simple method in converting from any base to the decimal base ten. If you remember how we dissected the numbers above, we used powers, such as 2^4, and ended up with a number we understand. This is exactly what we do to convert from a base to decimal. We find out the true value of each digit according to their placement and add them together.

In a formula, this algorithm looks like:

$$V_{10} = v_p B^p + v_{p-1} B^{p-1} + \ldots + v_1 B + v_0 \tag{1}$$

Where V_{10} is the decimal value, v is the digit in a placement, p is the placement from the right of the number assuming the rightmost placement is 0, and B is the starting base. Do not be daunted by the formula! We are going to go through this one step at a time.

So, let us say we had the simple hexadecimal number 2B. We want to know what this number is in decimal so that we can understand it better. How do we do this?

Let us use the formula above. Define every variable first. We want to find V_{10}, so that is unknown. The number $2B_{16}$ has two positions since it has two digits. p therefore is one less than that[1], so p is 1. The number is in base 16, so B is 16. Finally, we want to know what v is, but there are multiple v's. You have v_1 and v_0. This refers to the value of the digit in the subscripted position. v_1 refers to the digit in position one (the second digit from the right). So, v_1 is 2. v_0 is the first digit which is B. In the case of the conversion, you must convert all the letters to what they are in decimal. B is 11 in decimal, so v_0 is 11.

Now, plug all this into the formula:

$$\begin{aligned} V_{10} &= 2(16^1) + 11(16^0) \\ V_{10} &= 2(16) + 11(1) \\ V_{10} &= 32 + 11 \\ V_{10} &= 43 \end{aligned} \tag{2}$$

Therefore, $2B_{16}$ is equal to 43.

Now, let me explain how this works. Remember how digit placement affects the actual value? For example, in the decimal number 123, the "1" represents 100 which is $1*10^2$. The "2" is 20, or $2*10^1$. Likewise, in the number $2B_{16}$, the "2" is $2*16^1$, and the B is $11*16^0$.

We can determine the value of numbers in this way. For the number 364_8, we will make a chart that exposes the decimal value of each individual digit. Then, we can add them up so that we have the whole. The number has three digits, so starting from the right, we have position 0, position 1, and position 2. Since this is base eight, we will use powers of 8.

8^2	8^1	8^0
3	6	4

Now, 8^2 is 64. 8^1 is 8. 8^0 is 1. Now what?

Remember what we did with the decimal number 123? We took the value of the digit *times* the respective power. So, considering this further…

3*64	6*8	4*1
192	48	4

Now, we add the values together to get 244. Therefore, 364_8 is equal to 244_{10}.

In the same way that for 123, we say there is one group of 100, two groups of 10, and three groups of 1, for octal and the number 364, there are three groups of 64, six groups of 8, and four groups of 1.

Decimal to Base

Just like how we can convert from any base to decimal, it is possible to convert decimal to any base. Let us say that we want to represent the number 236_{10} in binary, octal, and hexadecimal. What we need to do is pretty much reverse whatever we did above. There isn't really a good formula for this, but there is an algorithm that you can follow which will help accomplish what we want.

$$\begin{aligned} (1)\ \ Let\ P = \operatorname{int}(\sqrt[B]{V}) \\ (2)\ \ Let\ v = \operatorname{int}(V \div B^P) \\ (v\ is\ the\ next\ digit\ to\ the\ right) \\ (3)\ \ Make\ V = V - vB^p \\ (4)\ \ Repeat\ steps\ 1\ through\ 3\ until\ p = 0 \end{aligned} \tag{3}$$

This algorithm may look confusing at first, but let us go through an example to see how it can be used. We want to represent 236 in binary, octal, and hexadecimal. So, let's try getting it to binary first.

The first step is to make p equal to $\operatorname{int}(\sqrt[B]{V})$. B is the base we want to convert to which is 2. The V is the number we want to convert, 236. Essentially, we are taking the square root of 236 and disregarding the decimal part. Doing this makes p become 7.

Step two says to let v equal our number V divided by B^p. B^p is 2^7, or 128, and the integer part of 236 divided by 128 is 1. Therefore, our first digit on the left is 1. Now, we actually change V to become V minus the digit times the B^p. So, V will now be 236-128, or 108.

We simply repeat the process until the p becomes a zero. When p becomes zero, we complete the steps a last time and then end.

So, since V is now 108, p becomes 6. 108 divided by 2^6 is 1. The 1 goes to the right of the 1, so now we have 1$\underline{1}$. V becomes 44 since 108-64 is 44.

How?

Now you might be asking yourself how to *read* these numbers. Well, that’s not so difficult. First, I’ll give a general mathematical explanation which can be fit into one formula:

$$V = vB^P \tag{4}$$

In human language: the value of the cipher in the number is equal to the value of the cipher on its own multiplied by the base of the number system to the power of the position of the cipher from left to right in the number, starting at 0. Read that a few times and try to understand it.

Thus, the value of a digit in binary **doubles** every time we move to the left. (see table below)

From this follows that every hexadecimal cipher can be split up into 4 binary digits. In computer language: a nibble. Now take a look at the following table:

Binary Numbers						
8	**4**	**2**	**1**		**Hexadecimal Value**	**Decimal Value**
0	0	0	0		0	0
0	0	0	1		1	1
0	0	1	0		2	2

0	0	1	1		3	3
0	1	0	0		4	4
0	1	0	1		5	5
0	1	1	0		6	6
0	1	1	1		7	7
1	0	0	0		8	8
1	0	0	1		9	9
1	0	1	0		A	10
1	0	1	1		B	11
1	1	0	0		C	12
1	1	0	1		D	13
1	1	1	0		E	14
1	1	1	1		F	15

Another interesting point: look at the value in the column top. Then look at the values. You see what I mean? Yeah, you’re right! The bits switch on and off following their value. The value of the first digit (starting from the right), goes like this: 0,1,0,1,0,1,0,1,0,1,… Second digit: 0,0,1,1,0,0,1,1,0,0,1,1,0,0… Third digit (value=4): 0,0,0,0,1,1,1,1,0,0,0,0,1,1,1,1,… And so on…

Now, what about greater numbers? Therefore we’ll need an extra digit. (but I think you figured that out by yourself). For the values starting from 16, our table looks like this:

Binary Numbers							
16	**8**	**4**	**2**	**1**		**Hexadecimal Value**	**Decimal Value**
1	0	0	0	0		10	16
1	0	0	0	1		11	17
1	0	0	1	0		12	18
1	0	0	1	1		13	19
1	0	1	0	0		14	20
1	0	1	0	1		15	21
1	0	1	1	0		16	22
1	0	1	1	1		17	23
1	1	0	0	0		18	24
1	1	0	0	1		19	25
1	1	0	1	0		1A	26
1	1	0	1	1		1B	27
1	1	1	0	0		1C	28
1	1	1	0	1		1D	29
1	1	1	1	0		1E	30
1	1	1	1	1		1F	31

For octals, this is similar, the only difference is that we need only 3 digits to express the values 1->7. Our table looks like this:

Binary Numbers					
4	**2**	**1**		**Octal Value**	**Decimal Value**
0	0	0		0	0
0	0	1		1	1
0	1	0		2	2

0	1	1		3	3
1	0	0		4	4
1	0	1		5	5
1	1	0		6	6
1	1	1		7	7

Conversion.

In the latter topic I explained the logic behind the binary, hexadecimal and octal number systems. Now I'll explain something more practical. If you fully understood the previous thing you can skip this topic.

From decimal to binary

- Step 1: Check if your number is odd or even.
- Step 2: If it's even, write 0 (proceeding backwards, adding binary digits to the left of the result).
- Step 3: Otherwise, if it's odd, write 1 (in the same way).
- Step 4: Divide your number by 2 (dropping any fraction) and go back to step 1. Repeat until your original number is 0.

An example:
Convert 68 to binary:

- 68 is even, so we write 0.
- Dividing 68 by 2, we get 34.
- 34 is also even, so we write 0 (result so far - 00)
- Dividing 34 by 2, we get 17.
- 17 is odd, so we write 1 (result so far - 100 - remember to add it on the left)
- Dividing 17 by 2, we get 8.5, or just 8.
- 8 is even, so we write 0 (result so far - 0100)
- Dividing 8 by 2, we get 4.
- 4 is even, so we write 0 (result so far - 00100)
- Dividing 4 by 2, we get 2.
- 2 is even, so we write 0 (result so far - 000100)
- Dividing 2 by 2, we get 1.
- 1 is odd, so we write 1 (result so far - 1000100)
- Dividing by 2, we get 0.5 or just 0, so we're done.
- Final result: 1000100

From binary to decimal

- Write the values in a table as shown before. (or do so mentally)
- Add the value in the column header to your number, if the digit is turned on (1).
- Skip it if the value in the column header is turned off (0).
- Move on to the next digit until you've done them all.

An example:
Convert 101100 to decimal:

- Highest digit value: 32. Current number: 32
- Skip the "16" digit, its value is 0. Current number: 32
- Add 8. Current number: 40
- Add 4. Current number: 44
- Skip the "2" and "1" digits, because their value is 0.
- Final answer: 44

From decimal to hexadecimal.

THIS IS ONLY ONE OF THE MANY WAYS!

- Convert your decimal number to binary

- Split up in nibbles of 4, starting at the end
- Look at the first table on this page and write the right number in place of the nibble

(you can add zeroes at the beginning if the number of bits is not divisible by 4, because, just as in decimal, these don't matter)

An example:
Convert 39 to hexadecimal:

- First, we convert to binary (see above). Result: 100111
- Next, we split it up into nibbles: 0010/0111 (Note: I added two zeroes to clarify the fact that these are nibbles)
- After that, we convert the nibbles separately.
- Final result: 27

From hexadecimal to decimal

*Check the formula in the first paragraph and use it on the ciphers in your hexadecimal number. (this actually works for any conversion to decimal notation)

An example:
Convert 1AB to decimal:

- Value of B = 16^0×11. This gives 11, obviously
- Value of A = 16^1×10. This gives 160. Our current result is 171.
- Value of 1 = 16^2×1. This gives 256.
- Final result: 427

From decimal to octal

- Convert to binary.
- Split up in parts of 3 digits, starting on the right.
- Convert each part to an octal value from 1 to 7

Example: Convert 25 to octal

- First, we convert to binary. Result: 11001
- Next, we split up: 011/001
- Conversion to octal: 31

From octal to decimal

Again, apply the formula from above

Example: convert 42 to decimal

- Value of 2=8^0×2=2
- Value of 4=8^1×4=32
- Result: 34

Fun Facts

OK, these may not be 100% "fun", but nonetheless are interesting.

- Do you tend to see numbers beginning with 0x? This is common notation to specify hexadecimal numbers, so you may see something like:

CHAR

```
0x000000
0x000002
0x000004
```

This notation is most commonly used to list computer addresses, which are a whole different story.

- This is pretty obvious, but you can "spell" words using hexadecimal numbers. For example:
  - CAB = 3243 in decimal notation.

End

Did you understand everything? If you think so, test yourself:

Bin	Dec	Hex
…	…	3A
…	76	…
101110	…	…
…	88	…
1011110	…	…
…	…	47

Make some exercises yourself, if you want some more.

Footnotes

1. It is one less because the rightmost position is 0, not 1. So p is always one less than the number of digits.