Everything you wanted to know about using Hexadecimal and Octal Numbers in Visual Basic 6

Number Systems

No course on programming would be complete without a discussion of the Hexadecimal (Hex) number system. Many beginners have difficulty dealing with Hexadecimal numbers, and I think the reason for that is that no one has ever taken the time to discuss it with them. So I want to take just a few minutes to “de-mystify” this topic for you. In order to discuss the Hexadecimal number system however, I first need to discuss our native number system, the Decimal number system. From there I’ll move on to the less used Binary and Octal systems, and finish up with Hex.

The Decimal System

The Decimal Number system is quite naturally the one you’re most familiar with. By explaining a little bit of the theory behind the Decimal number system, I hope to make it easier for you to understand the other three. (Binary, Octal, and Hex)

Each number system has a set of numbers with which it deals. The Decimal system uses the numbers 0,1,2,3,4,5,6,7,8 and 9. In the Decimal system, if you add the number 1 to the number 9, you 'run out of' numbers. You must then 'start over' with the first number in the system, the zero. For example, if you add 1 to 9 on a piece of paper, you write down the 0, then carry a '1' to the next column. Therefore 9 + 1 gives you 10 (sometimes read as “One Zero Decimal” by math nerds).

That wasn’t too bad, was it? That’s really all you need to know in order to understand the other number systems.

The Binary System

We don’t have to work with the Binary number system at all in Visual Basic. However, since the Binary number system is the native number system of the computer (remember, the computer does all its work with zeroes and ones), you might as well see how it works.

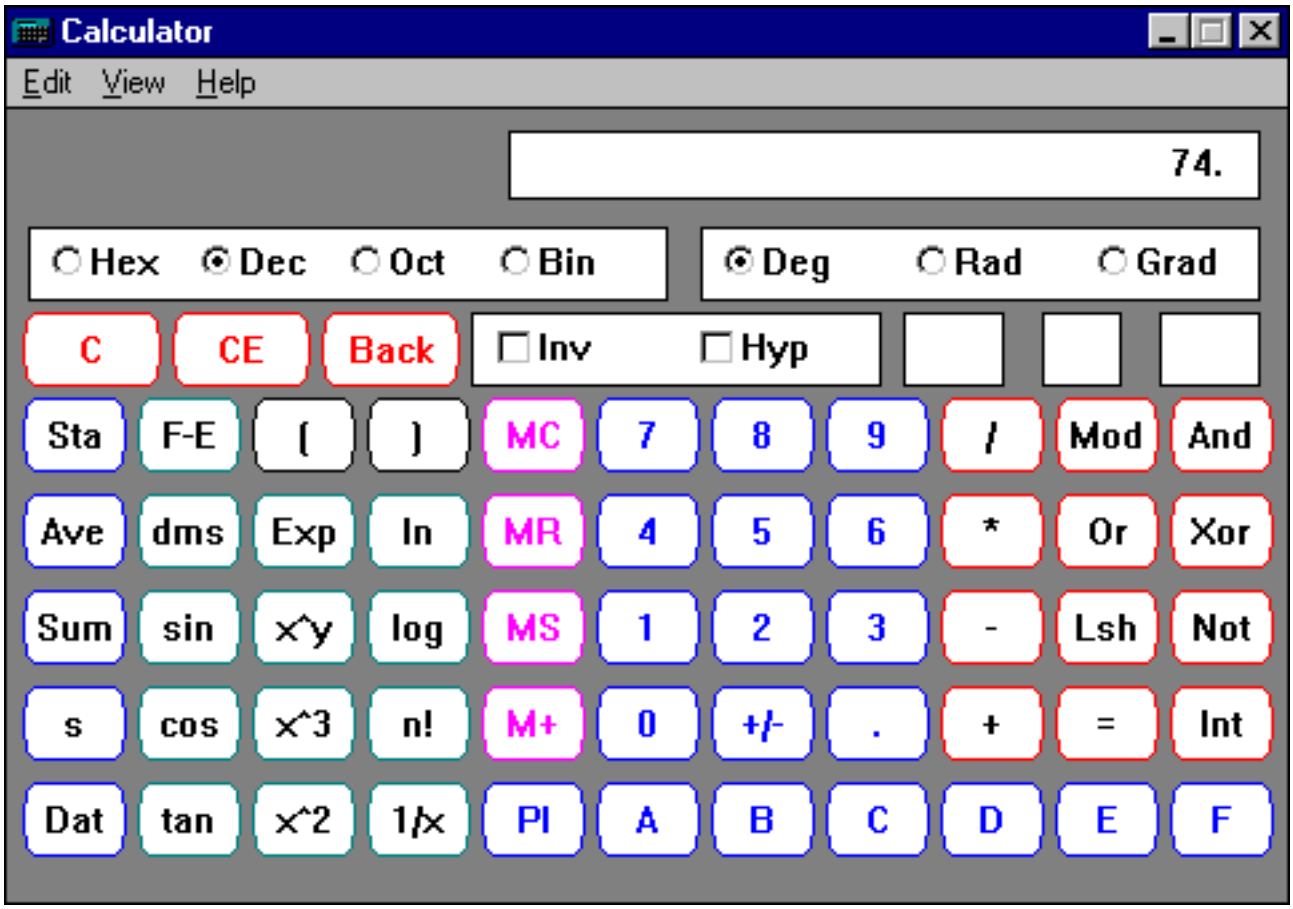
Instead of working with 10 numbers as the Decimal number system does, the Binary number system uses only two numbers: 0 and 1. Just as you can do in the Decimal number system, you can also add numbers in the Binary system. You just don’t get nearly as far before you have to 'carry'. In Decimal arithmetic, you carry whenever your result will be greater than 9. In Binary arithmetic, you must carry whenever your result will be greater than 1. Therefore, in Binary arithmetic, adding 1 to 1 gives you 10, which is not read as 'ten' but as 'One Zero Binary'. Fortunately,

we don’t need to do Binary arithmetic, or Octal or Hexadecimal arithmetic for that matter. The computer does all of that for us.

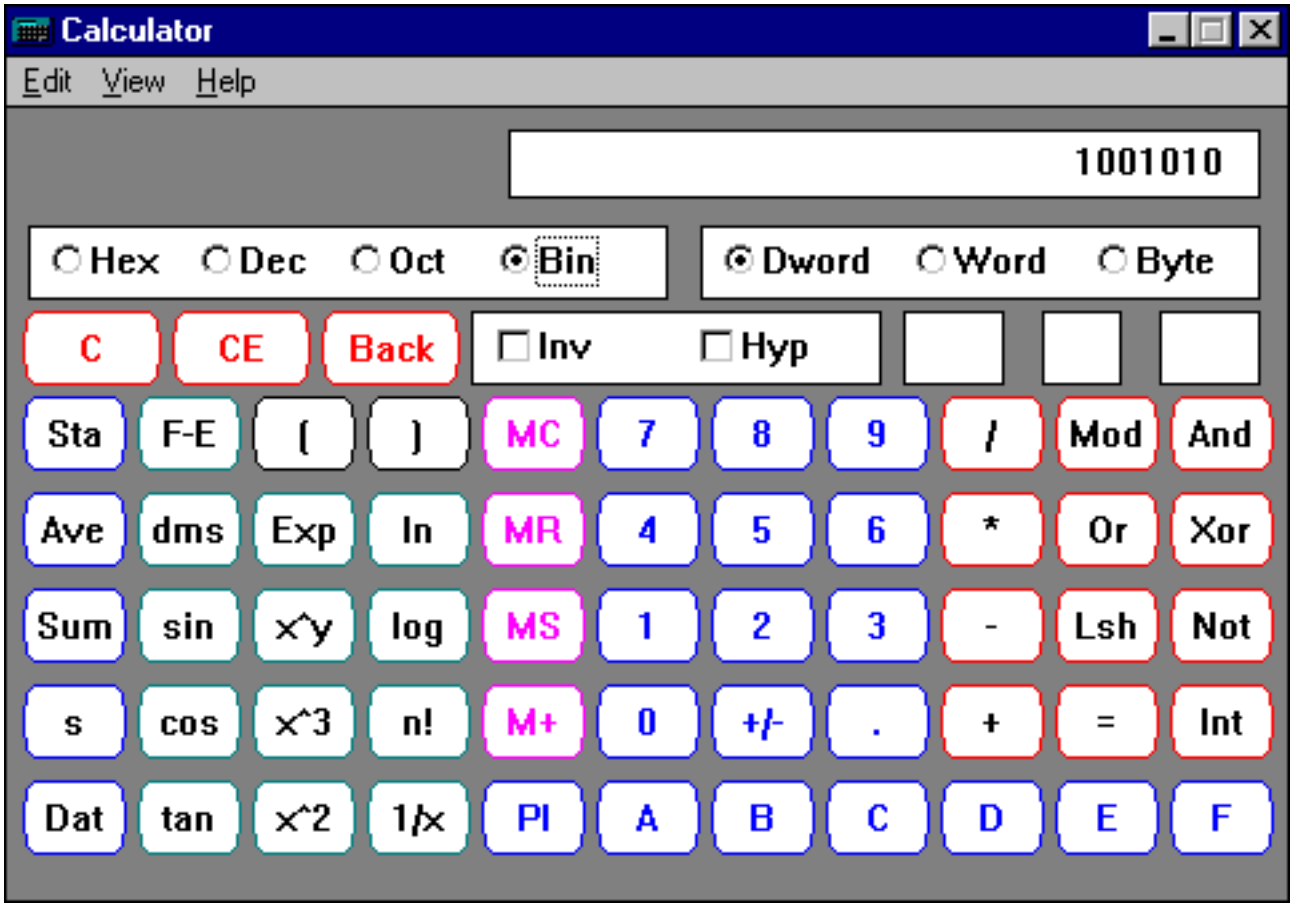
Suppose you need to convert a Binary number to Decimal, and vice versa? Now that you're back up off the floor, let me tell you that the likelihood you'll ever need to do that is remove. However, just in case, I’d like to show you how to convert a Binary Number to a Decimal one.

The easy way is to use the Windows calculator. If you open up the calculator, select View from the Calculator’s menu, and then select Scientific, you can use the calculator to perform this conversion.

For instance, the ASCII chart included as Appendix D of the book indicates that the capital letter 'J' is equal to the Decimal value of '74'. It also is equivalent to the Binary value of '01001010'. Let's see the conversion in action. Just key in the number that you wish to convert---in this case '74'.



And then select the option button corresponding the number system you wish to convert to---in this case Bin for Binary.



As you can see, the Binary equivalent of Decimal '74' is '1001010'. Now I know that you really want to know how this is done. So, let's take a shot at converting this operation on our own. First, the Binary to Decimal conversion.

You're going to need a piece of scrap paper for this one. Write down these numbers at the top of a sheet of paper.

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1

These numbers represent, from right to left, 2 raised to the power of 0, 2 raised to the power of 1, 2 raised to the power of 2, all the way through to 2 raised to the power of 7. Underneath of the first line of numbers is the value equating to the Exponentiation. There are two rules to remember. Any number raised to the power of 0 is 1. And any number raised to the power of 1 is itself.

Now take the number you want to convert (in this case, Binary '01001010'), and place this number underneath of the Powers of two, one by one starting from the left and working your way to the right. Like so.

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1
0	1	0	0	1	0	1	0

Now multiply each number by the value above it and add the results. Like so.

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1
x 0	x 1	x 0	x 0	x 1	x 0	x1	x 0
0	64	0	0	8	0	2	0

0 + 64 + 0 + 0 + 8 + 0 + 2 + 0 = 74

The result of the addition is the Decimal Equivalent of the Binary number **01001010**.

Converting from a Decimal number to a Binary number isn't quite as easy. It requires us to do a form of division. But hang on, it's not as bad as it sounds. Let's convert the Decimal number 74 back to a Binary number. Here are the steps we need to perform.

1. **Start with the Decimal number 74.**

2. **Take the number 74, and attempt to divide 128 into it. It doesn't go, so write 0 down.**

3. **Take the number 74, and attempt to divide 64 into it. It goes once, so write 1 down.**

4. **Subtract 64 from 74, and write the remainder 10 down.**

5. **Take the number 10, and attempt to divide 32 into it. It doesn't go, so write 0 down.**

6. **Take the number 10, and attempt to divide 16 into it. It doesn't go, so write 0 down.**

7. **Take the number 10, and attempt to divide 8 into it. It goes once, so write 1 down.**

8. Subtract 8 from 10, and write the remainder 2 down.

9. Take the number 2, and attempt to divide 4 into it. It doesn't go, so write 0 down.

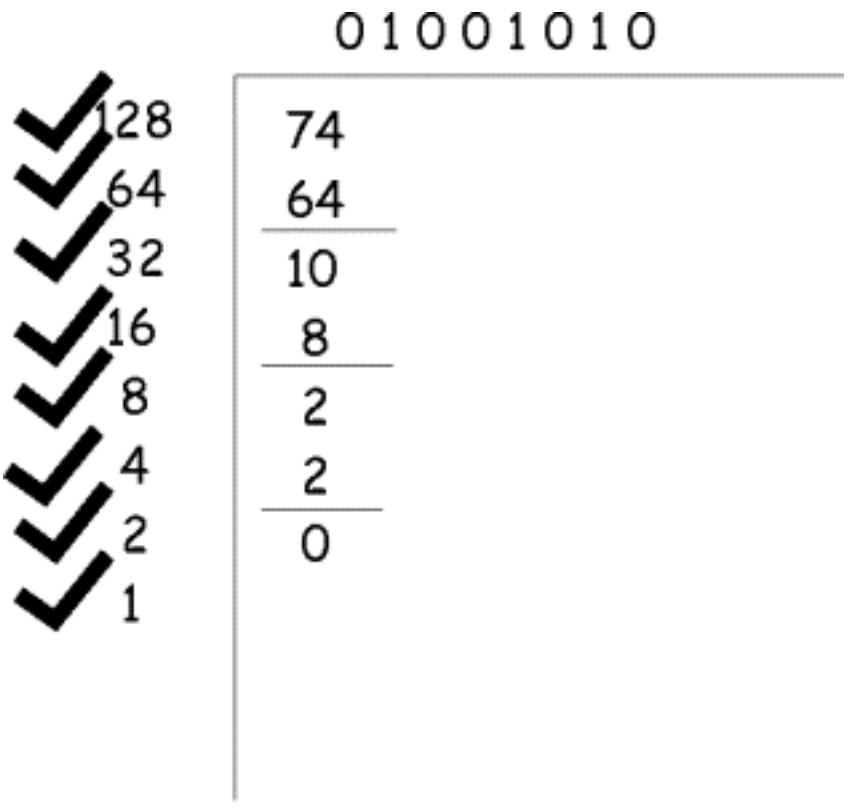
10. Take the number 2, and attempt to divide 2 into it. It goes once, so write 1 down.

11. Subtract 2 from 2, and write the remainder 0 down.

12. Take the number 0, and attempt to divide 1 into it. It doesn't go, so write 0 down.

Numbers written down: 01001010

How do you decide which numbers to divide by? These are the same numbers that are the column headers for the conversion from Binary to Decimal.



The numbers you've written down, from left to right, are the Binary equivalent of Decimal 74!

The largest 8 digit Binary number you can have is 11111111 which equates to Decimal 255.

The Octal System

The Octal number system uses only 8 numbers: 0,1,2,3,4,5,6, and 7. Just as we saw earlier with the Binary and Decimal systems, it is also possible to add numbers in the Octal system. In Octal, you carry whenever your result will be greater than 7. In Octal arithmetic, adding 1 to 7 gives you 10, which is not read as “ten” but as “One Zero Octal”.

Let's continue working with the ASCII values for the capital letter 'J'. We saw that the Decimal equivalent is '74', and the Binary equivalent is '01001010'. What about the Octal equivalent? The ASCII chart in Appendix D indicates that the capital letter “J” is equivalent to a the Octal value of '112'. Again, you can use the Windows calculator to verify that, or you can use the same method we used to convert a Binary number to a Decimal number.

Again, you’re going to need a piece of scrap paper for this one. Write down these numbers at the top of a sheet of paper.

8^3	8^2	8^1	8^0
512	64	8	1

These numbers represent, from right to left, 8 raised to the power of 0, 8 raised to the power of 1, 8 raised to the power of 2, all the way through to 8 raised to the power of 4. Underneath of the first line of numbers is the value equating to the Exponentiation. Again, two rules to remember. Any number raised to the power of 0 is 1. And any number raised to the power of 1 is itself.

Now take the number you want to convert (in this case, Octal '112'), and place this number underneath of the Powers of eight, one by one starting from the left and working your way to the right. Like so.

8^3	8^2	8^1	8^0
512	64	8	1
0	1	1	2

Now multiply each number by the value above it and add the results. Like so.

8^3	8^2	8^1	8^0
512	64	8	1
x 0	x 1	x 1	x 2
0	64	8	2

0 + 64 + 8 + 2 = 74

The result of the addition is the Decimal Equivalent of the Octal number 74.

We can convert from Decimal to Octal in a similar manner to the way we converted from Decimal to Binary. Except that there are fewer steps.

1. **Start with the Decimal number 74.**

2. **Take the number 74, and attempt to divide 512 into it. It doesn't go, so write 0 down.**

3. **Take the number 74, and attempt to divide 64 into it. It goes once, so write 1 down.**

4. **Subtract 64 from 74, and write the remainder 10 down.**

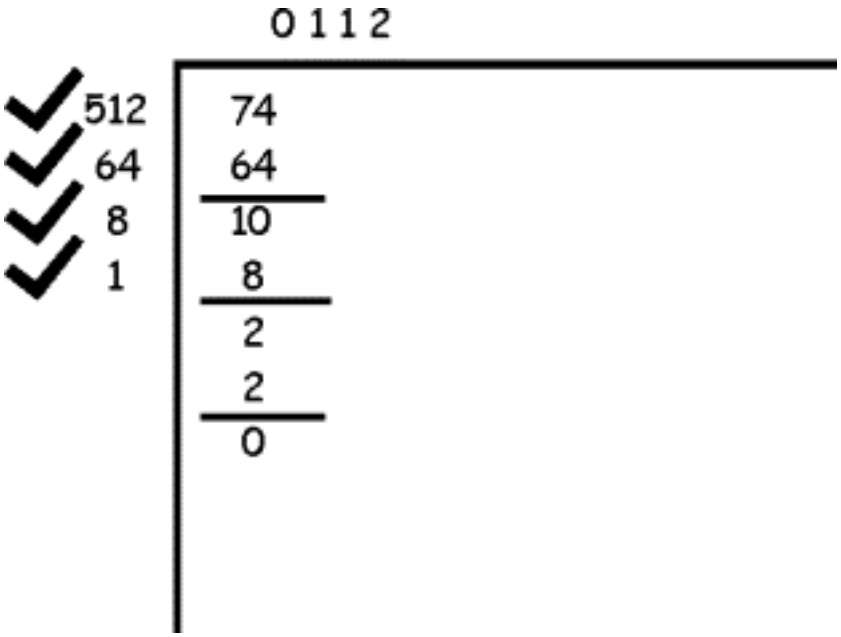
5. **Take the number 10, and attempt to divide 8 into it. It goes once, so write 1 down.**

6. **Subtract 8 from 10, and write the remainder 2 down.**

7. **Take the number 2, and attempt to divide 1 into it. It goes twice, so write 2 down.**

Numbers written down: 0112

How do you decide which numbers to divide by? These are the same numbers that are the column headers in the conversion from Octal to Decimal.



The numbers you've written down, from left to right, are the Octal equivalent of Decimal 74!

The Hexadecimal System

The Hexadecimal number system is a popular form of data representation because a byte (8 bits) can be represented with just two Hexadecimal characters. For systems programmers, this is a big advantage because they don't have to look at eight ones and zeroes and try to decipher what the equivalent character is. Instead, they can look at two hex characters instead. Even more important than the relative ease that Hexadecimal provides to Systems Programmers is the fact that Hexadecimal numbers appear here and there in Visual Basic. Most notably, in objects that have a BackColor or ForeColor property. So you see, a little knowledge of the Hexadecimal number system can come in handy.

Unlike the Binary and Octal number systems that use fewer numbers in their number systems than the Decimal number system, the Hexadecimal system uses more, and I believe this is where much of the confusion and hesitancy of working with Hex comes in. Beginners seem to find the notion of working with more numbers than ten a bit unnerving. But if you just stick to the methods I've shown you so far, you'll be fine.

The Hexadecimal number system uses 16 characters: 0,1,2,3,4,5,6,7,8,9,A,B,C,D, E, and F. Just as we saw with the other number systems, you can add numbers in the Hexadecimal system. Unlike the Decimal system, where you carry when your result is greater than 9, in Hex we don't carry at that point. In Hex,

9 + 1 = A (The Decimal equivalent is 10)

A + 1 = B (The Decimal equivalent is 11)

B + 1 = C (The Decimal equivalent is 12)

C + 1 = D (The Decimal equivalent is 13)

D + 1 = E (The Decimal equivalent is 14)

E + 1 = F (The Decimal equivalent is 15)

F + 1 = 10 (The Decimal equivalent is 16)

As it turns out, it isn't until the result of a Hex addition is greater than 'F' that we carry. Just as we did in the other number systems, we place a zero down and carry the 1. 'F' + 1 is 10, which is not read as 'ten' but as , you guessed it, "One Zero Hexadecimal". Its Decimal equivalent is 16.

Let's continue working with the ASCII values for the capital letter 'J'. We saw that the Decimal equivalent is '74', the Binary equivalent is '01001010', and the Octal value is '112'. What about the Hexadecimal value? According to the ASCII chart in Appendix D, the Hex value is '4A'. How do we convert that to Decimal?

Don't forget, if your patience is wearing thin, you can always use the calculator. But I know that you're starting to get the feeling for this.

Again, you're going to need a piece of scrap paper for this one. Write down these numbers at the top of the paper.

16^2	16^1	16^0
256	16	1

These numbers represent, from right to left, 16 raised to the power of 0, 16 raised to the power of 1, and 16 raised to the power of 2. (For larger Hex numbers, we would need to add additional columns.). Underneath of the first line is the value equating to the Exponentiation. Again, two rules to remember. Any number raised to the power of 0 is 1. And any number raised to the power of 1 is itself.

Now take the number you want to convert (in this case, Hex '4A'), and place this number underneath of the Powers of sixteen, one by one starting from the left and working your way to the right. Like so.

16^2	16^1	16^0

256	16	1
0	4	A

Now multiply each number by the value above it and add the results. Like so.

16^2	16^1	16^0
256	16	1
x 0	x 4	x A
0	64	10

0 + 64 10 = 74

The result of the addition is the Decimal equivalent of the Hexadecimal number 4A. You probably found that this conversion was a little trickier than the previous examples because you had to handle the hex multiplication of

1 X A

Remember, in Hex, the letter A equates to Decimal 10, so that expression is really the same as

1 X 10

We can convert from Decimal to Hex in a similar manner to the way we converted from Decimal to Binary. Except that there are fewer steps.

1. **Start with the Decimal number 74.**

2. **Take the number 74, and attempt to divide 256 into it. It doesn't go, so write 0 down.**

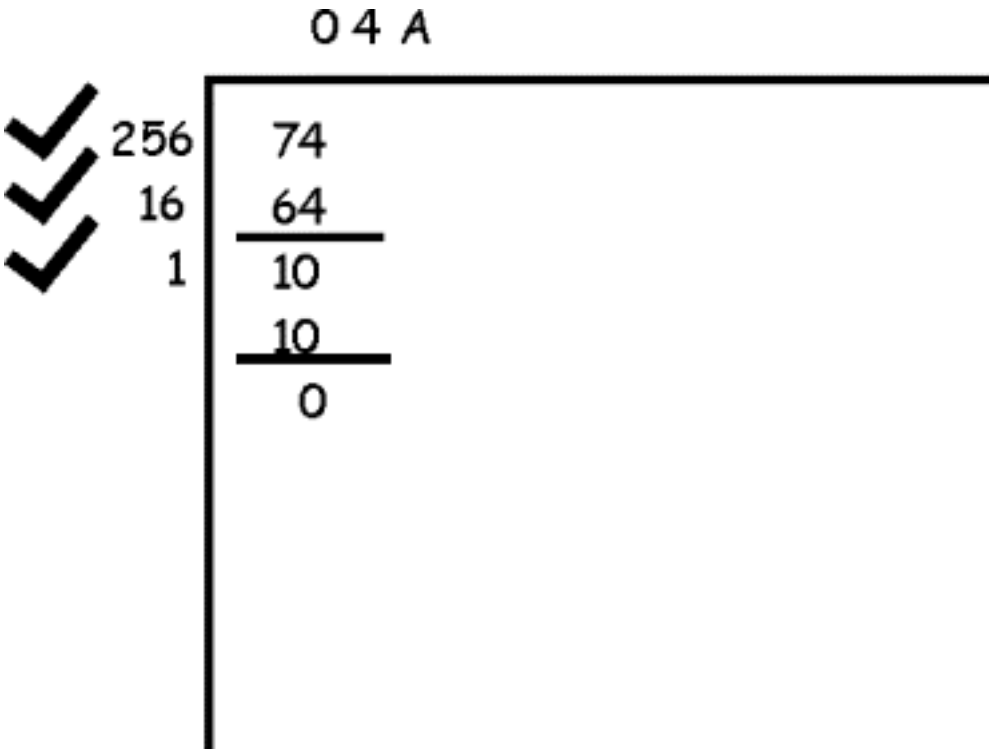
3. **Take the number 74, and attempt to divide 16 into it. It goes into it 4 times, so write 4 down.**

4. **Multiply 4 by 16 (in Decimal), and subtract 64 from 74. Write the remainder 10 down.**

5. **Take the number 10, and attempt to divide 1 into it. It goes ten times. But you can't write 10 down, you have to write the Hex equivalent of 10 which is "A"**

Numbers written down: 04A

How do you decide which numbers to divide by? These are the same numbers that are the column headers in the conversion from Hex to Decimal.



The numbers you've written down, from left to right, are the Hex equivalent of Decimal 74!

Is there an easier way to do these conversions?

Nothing could be easier than using the Windows calculator to perform these conversions.. Unfortunately, you can't use the Windows calculator in code to perform these conversions. There are two functions, **Hex** and **Oct** that will return Decimal values. These are really pretty easy to use, and I'll leave it up to you read up on them in On-line Help.

When am I likely to use Hex?

Most beginners are happy to hear that they're not likely to use Hex all that often. But one place you're bound to run into Hexadecimal values is any property that sets or returns a color related value. You'll know that you're dealing with a Hex number because the number will have an ampersand and the letter H in front of it.

For instance, this value

&H00FFFFFF&

is a value that you might see in the BackColor property of a form. **&H** lets you know that you are dealing with a Hexadecimal Value. If by chance you see this

&O00000071&

you know you are looking at an Octal number because of the leading &O value.

I talk about Visual Basic Color properties in another article.