

# Number System Essentials for Embedded Control

**James Kokernak**

Department of Electric Power  
Rensselaer Polytechnic Institute

## 1. Introduction

In Embedded Control, there are 3 basic number systems that we will use. We have been taught to think in terms of decimal, or base 10 systems. This system operates on the same principle as an odometer in a car. Each digit in the number can vary from 0 to 9. Binary operates on the same principle, except we only work with base 2 (i.e., instead of 10 possible digits, we only have 2). Hexadecimal is the same except that we use base 16. Consider counting from 1 to 32 in the three number forms:

<b>Decimal</b>	<b>Binary</b>	<b>Hexadecimal</b>
1	0000 0001	0x01
2	0000 0010	0x02
3	0000 0011	0x03
4	0000 0100	0x04
5	0000 0101	0x05
6	0000 0110	0x06
7	0000 0111	0x07
8	0000 1000	0x08
9	0000 1001	0x09
10	0000 1010	0x0A
11	0000 1011	0x0B
12	0000 1100	0x0C
13	0000 1101	0x0D
14	0000 1110	0x0E
15	0000 1111	0x0F
16	0001 0000	0x10
17	0001 0001	0x11
18	0001 0010	0x12
19	0001 0011	0x13
20	0001 0100	0x14
21	0001 0101	0x15
22	0001 0110	0x16
23	0001 0111	0x17
24	0001 1000	0x18
25	0001 1001	0x19
26	0001 1010	0x1A
27	0001 1011	0x1B
28	0001 1100	0x1C
29	0001 1101	0x1D
30	0001 1110	0x1E
31	0001 1111	0x1F
32	0010 0000	0x20

The process of converting from one number system to the other is pretty straightforward; it just takes repetition to get it straight in your head. There are 6 possible conversions you could be asked to perform. They are described below:

## 2. *Converting from Binary to Decimal*

Remember that an 8 bit representation of a number (binary representation) has bit 0 furthest to the right and bit 7 furthest to the left. The value of a '1' in a given bit position is 2 raised to that power. For example, the binary representation

$$0001\ 0000 = 2^4 = 16, \text{ because there is a 1 in the bit 4 position}$$

$$0001\ 0101 = (2^4)+(2^2)+(2^0) = 16+4+1 = 21, \text{ because there is a 1 in the 4, 2 and 0 bit slots.}$$

You should prove to yourself that  $1111\ 1111 = 255$ , which is the largest number that an 8 bit structure can manage.

Other examples:

$$0100\ 1100 = 76$$

$$1001\ 0001 = 145$$

$$0011\ 1101 = 61$$

## 3. *Converting from Binary to Hexadecimal*

Hexadecimal is convenient and goes well with binary. Why is that? Consider the binary number 0000 1111. What does it equal in decimal? (Answer 15) There are 16 numerical values that can be represented in a series of four binary values. That is why 16 is the base of a hexadecimal (hex) system. This means that a single digit in a hex number corresponds directly to a 4-digit group within a binary number. They both exceed their limits and roll over at the same point. Look at the chart on the first page and see the binary and hex entries from 0 to 16. You will see that as we go from 15 to 16, the rightmost (least significant) digit of the hex number goes to zero, as do the rightmost 4 digits of the binary number. These 4 digits are referred to as a "nibble". This makes the conversion from binary to hex very easy. You will notice that I often bracket the groups of 4 digits in a binary number. You can do a direct conversion of the individual nibbles to come up with the hex representation.

Examples:

1100 0010	Most significant nibble:	1100 = 12 (decimal) = C (Hex)
	Least significant nibble:	0010 = 2 (decimal) = 2 (Hex)

Therefore,  $1100\ 0010 = 0xC2$

Other examples:

$$0101\ 1110 = 0x5D$$

$$0001\ 0101 = 0x15$$

$$1001\ 1000 = 0x98$$

#### 4. *Converting from Decimal to Binary*

This is the biggest pain in my view. In essence, you need to keep subtracting out the largest power of 2 that fits into the number. Powers of 2 associated with the bit positions of a binary number are:

$$128 \quad 64 \quad 32 \quad 16 \quad 8 \quad 4 \quad 2 \quad 1$$

So consider the number 165. Convert it to binary. The largest of the powers of 2 that will fit into 165 is 128, so a 1 goes into that position. The whole exercise works like this:

$$\begin{aligned} &165 \\ &-128\text{-----}\diamond\text{put a 1 into bit 7} \\ &=37\text{-----}\diamond64\text{ doesn't fit into 37, put a 0 into bit 6} \\ &-32\text{-----}\diamond\text{put a 1 into bit 5} \\ &=5\text{-----}\diamond16\text{ and 8 don't fit into 5, put 0's in bits 4 and 3} \\ &-4\text{-----}\diamond\text{put a 1 into bit 2} \\ &=1\text{-----}\diamond2\text{ doesn't fit into 1, so put a 0 into bit 1} \\ &-1\text{-----}\diamond\text{put a 1 into bit 0} \\ &=0 \end{aligned}$$

So our binary representation is 1010 0101

A similar method is divide the number by 2, putting the remainder into successive bit positions beginning with the least significant position.

$165/2 = 82$ rem 1	put a 1 in bit 0
$82/2 = 41$ rem 0	put a 0 in bit 1
$41/2 = 20$ rem 1	put a 1 in bit 2
$20/2 = 10$ rem 0	put a 0 in bit 3
$10/2 = 5$ rem 0	put a 0 in bit 4
$5/2 = 2$ rem 1	put a 1 in bit 5
$2/2 = 1$ rem 0	put a 0 in bit 6
$1/2 = 0$ rem 1	put a 1 in bit 7

So the results is, not suprisingly, the same:  $165 = 1010\ 0101$

This method is the one covered in class. The top method may more clearly demonstrate where the numbers are coming from. With this second method, make sure that you know where to stop dividing. You do this when you get to 2 going into the number 0 times. Look at how to handle an even number below:

Convert 2 to binary:

$$2/2 = 1 \text{ rem } 0 \quad \text{put a 0 in bit 0}$$

$$1/2 = 0 \text{ rem } 1 \quad \text{put a 1 in bit 1}$$

### 5. *Converting from Decimal to Hexadecimal*

For this we need to extract powers of 16 instead of powers of 2. For a standard 8-bit number, we only have two nibbles to consider. Let's consider a 16-bit number to be complete. A 16-bit number has 4 nibbles instead of 2. Each nibble has a power of 16 associated it.

$$16^3 = 4096$$

$$16^2 = 256$$

$$16^1 = 16$$

$$16^0 = 1$$

If we wanted to convert 57 to hex, we could realize that 4096 and 256 are both too large to be divided out. This tells us that 57 fits into an 8 bit structure. We start with 16 to find the hex number:

$$57/16 = 3 \text{ rem } 9 \quad \text{put a 3 into bit 1}$$

$$9/1 = 9 \text{ rem } 0 \quad \text{put a 9 into bit 0}$$

therefore  $57 = 0x39$ , or  $0x0039$  to be consistent with the 16 bit representation.

How about 1250 into hex?

$$1250/256 = 4 \text{ rem } 226 \quad \text{put a 4 into bit 2}$$

$$226/16 = 14 \text{ rem } 2 \quad \text{put a E into bit 1}$$

$$2/1 = 2 \text{ rem } 0 \quad \text{put a 2 into bit 0}$$

$$1250 = 0x04E2$$

For the most part, we use 8-bit numbers, so you just need to divide by 16, put that hex result into

the left digit and put the remainder into the right.

$$92 = 0x5C$$

$$52 = 0x34$$

$$255 = 0xFF$$

## 6. *Converting from Hexadecimal to Decimal*

This is just the reverse of the above procedure. For an 8-bit hex number, you simply multiply the left digit by 16 and add the right digit.

$$0x54 = (16^1)*5+(16^0)*4 = 84$$

$$0xBF = (16^1)*11+(16^0)*15 = 191$$

In the event that we have a 16-bit number, we need to multiply the additional digits by higher powers of 16.

$$\begin{aligned} 0x14A2 &= (16^3)*1+(16^2)*4+(16^1)*10+(16^0)*2 = 4096+1024+160+2 = \\ &= 5282 \end{aligned}$$

## 7. *Converting from Hexadecimal to Binary*

Here we just convert each digit of the hex number into a binary nibble.

$$0x75$$

$$7 = 0111$$

$$5 = 0101 \quad \text{so } 0x75 = 0111 \ 0101$$

$$0xF2 = 1111 \ 0010$$

$$0x3A = 0011 \ 1010$$

$$0xB4C9 = 1011 \ 0100 \ 1100 \ 1001$$

$$0x0307 = 0000 \ 0011 \ 0000 \ 0111$$

## 8. *Checking Your Results*

It is very easy to make mistakes doing these conversions. I made several while working on this guide. There may still be mistakes in here. Once you get comfortable performing the conversions, you should develop the habit of double-checking your results by converting to the third number system and comparing the results. For instance, if you are asked to convert from decimal to binary, convert the decimal number also to hex and make sure the binary and hex numbers agree. It takes a little longer, but you won't make too many mistakes on exams that way. On the next page is a worksheet that you can use as practice. The last page has the solutions.