Decoding the Unique Identifiers

As I mentioned in my previous post, I am able to read the ID's of both the cards that came in my kit, however these ID's are raw. We can decode these into different formats.

By reading in the code byte by byte the code is read, by default, in decimal (DEC).

For those who haven't met the different number base's, in our normal day to day lives we use the decimal number system which is base 10, ie 10^0 = 1, 10^1 = 10, 10^2 = 100 . . . Base 10 only uses combinations of the numbers 0 to 9. Other common bases include:

Hexadecimal, base 16 ie 16^0 = 1, 16^1 = 16, 16^2 = 256 . . . Base 16 uses a combination of 0 to 9 and A to F to represent 16 digits.

Binary, base 2 ie 2^0 = 1, 2^1 = 2, 2^2 = 4 . . . Base 2 uses only two digits, 0 and 1.

So using this we can adapt what is sent using the Serial.printcommand by writing the value or variable we want to send, followed by BIN for binary, DEC for decimal, HEX for hexadecimal, as well as others that I haven't covered.
so using

```
int val = 0; // variable to store the data from the serial port

void setup() {
Serial.begin(9600); // connect to the serial port
}
void loop () {
// read the serial port
if(Serial.available() > 0) {
val = Serial.read();
Serial.println(val,DEC/HEX/BIN); //use one Base as appropriate.
}
}
```

We can read the value in any base system supported by Arduino.

DEC
2 53 49 48 48 55 66 69 65 52 50 56 50 13 10 3
2 53 48 48 48 56 70 65 52 66 56 67 51 13 10 3

HEX
2 35 31 30 30 37 42 45 41 34 32 38 32 D A 3
2 35 30 30 30 38 46 41 34 42 38 43 33 D A 3

BIN
10 110101 110001 110000 110000 110111 1000010 1000101 1000001 110100 110010 111000 110010 1101 1010 11
10 110101 110000 110000 110000 111000 1000110 1000001 110100 1000010 111000 1000011 110011 1101 1010 11

Source: http://electronicbyte.cc/2012/10/17/decoding-the-unique-identifiers/