Hex Translator

```
//http://www.bouncycastle.org/
//MIT X11 License

using System;

namespace Org.BouncyCastle.Utilities.Encoders
{
///

/// A hex translator.
///
public class HexTranslator
{
private static readonly byte[] hexTable =
{
(byte)'0', (byte)'1', (byte)'2', (byte)'3', (byte)'4',
(byte)'5', (byte)'6', (byte)'7',
(byte)'8', (byte)'9', (byte)'a', (byte)'b', (byte)'c',
(byte)'d', (byte)'e', (byte)'f'
};

///

/// Return encoded block size.
///
/// 2
public int GetEncodedBlockSize()
{
return 2;
}

///

/// Encode some data.
///
/// Input data array. /// Start position within input data
```

```
array. /// The amount of data to process. /// The output data array. /// The offset within the output data array to start writing from. /// Amount of data encoded.
public int Encode(
byte[] input,
int inOff,
int length,
byte[] outBytes,
int outOff)
{
for (int i = 0, j = 0; i < length; i++, j += 2) {
outBytes[outOff + j] = hexTable[(input[inOff] >> 4) & 0x0f];
outBytes[outOff + j + 1] = hexTable[input[inOff] & 0x0f];

inOff++;
}

return length * 2;
}

///

/// Returns the decoded block size.
///
/// 1
public int GetDecodedBlockSize()
{
return 1;
}

///

/// Decode data from a byte array.
///
/// The input data array. /// Start position within input data array. /// The amounty of data to process. /// The output data array. /// The position within the output data array to start writing from. /// The amount of data written.
public int Decode(
```

```
byte[] input,
int inOff,
int length,
byte[] outBytes,
int outOff)
{
int halfLength = length / 2;
byte left, right;
for (int i = 0; i < halfLength; i++) { left = input[inOff + i
* 2]; right = input[inOff + i * 2 + 1]; if (left < (byte)'a')
{ outBytes[outOff] = (byte)((left - '0') <
```