ZPad$() - pad a number with leading zeroes

Sometimes it is nice to have leading zeroes on a number. MMBasic helpfully removes these for you which means to have them, the number must be returned as a string.

Str$ in MMBasic is deceptively powerful and goes some way to replacing the missing (on Micromites at least) FORMAT$ and USING. It can be tricky to get your head round the functionality so think of this function as a wrapper for a specific form.

This tiddly Function returns a string of the integer value x, padded to (at least) length y with leading zeroes. I say at least because if x already exceeds y then nothing is added.

This is a newer version of this function; it is better as it uses inbuilt “primitive” functionality to do the padding rather than “re-inventing the wheel” by string slicing. Consequently it is very marginally faster - rough tests seem to indicate a 16% improvement in speed. It also eliminates the potential restriction of 127 characters on the desired string.

If the string version of the argument already exceeds the desired length, the original number is returned - e.g. if you ask for 1000 to be padded to 3 digits.

Syntax: =ZPad$(number,totalNumDigits)

Example: Print ZPad$(7,3)' returns “007”

Code:

```
  Function ZPad$(x As Integer,y As Integer)
    ZPad$=Str$(x,y,0,"0")
  End Function
```

See Also

uFormat$() and StrE$() provide flexible formatting of numeric values

Hold your horses! A Note on Str$(): You should probably use Str$ as a stand-alone and not ZPad$() where-ever needed in your program. The complete functionality of Str$ can be a bit difficult to get to grips with, but learning to use it properly is a much better way and better understanding leads to better programs. ZPad$() assumes you only want to pad with zeroes. The overhead of calling this as a function is also undesirable - a loop of 1000 iterations at 48MHz takes 492mS using ZPad$(), the same using Str$ explicitly takes 177mS - two-and-a-half times quicker and a smaller code footprint.

That said, Str$ makes (reasonable) assumptions too. Suppose one where trying to pad the right of a string. Consider the following:

```
Print Str$(53,4,4,"*")
```

you might expect the string

```
**53.****
```

but the pad character (*) is ignored for the third argument (digits after the decimal point) and zero is mandated.

```
**53.0000
```

If you were trying to emulate the action of the old LSet and RSet statements of MicroSoft Basics (ignoring that Str$ only works with floats and integers), you'd still need to write a function similar to the old version of ZPad$() to get LSet. Str$ can do RSet on it's own but not LSet due to the above mandatory 0 padding.