LTrim and RTrim Functions (VB work-a-like)

The following functions return the input string with leading or trailing white-space characters (&H20 & &H09) removed.

If you need the functions for MM+, MMX, PiCromite, or ARMMite you should use the versions shown below. If you are using a MicroMite Mk2 (28 or 48 pin MX170), the below are fully compatible, but you are better off using these versions. Being CFunctions, they are hugely faster (at least 10x and often nearer 100x) - not `precisely` VB compatible as they only recognise a CHR$(32) as white space - unlikely to be a problem.

Using CONSTs and a temporary string for the compare slice gives a good speed increase. =LTrim$(RTrim$("a"))takes only 11.5mS at 48MHz on MX170

Syntax:

```
  LTrim$(expression)
  RTrim$(expression)
```

Example:

```
  a$=RTrim$(userinput$)
```

Code:

```
'preamble
    Const sT$=Chr$(9), sS$=" "

    Function LTrim$(a$)
        Local Integer m
        For m=1 To Len(a$)
            If Not(Mid$(a$,m,1)=" " Or Mid$(a$,m,1)=Chr$(9)) Then
LTrim$=Mid$(a$,m): Exit Function
        Next
        LTrim$=""
    End Function

    Function RTrim$(a$)
        Local Integer n
        For n=Len(a$) TO 1 Step -1
            If Not(Mid$(a$,n,1)=" " Or Mid$(a$,n,1)=Chr$(9)) Then
RTrim$=Left$(a$,n): Exit Function
        Next
        RTrim$=""
    End Function
```

For those needing reduced timings without resorting to CSubs, in the alternative RTrim$() below, some tricks which speed up things up by about 25% are used; Instead of removing padding from the right by repeated string slicing, it analyzes bytes - stopping when it hits a non-space. It then POKEs that position into the variable descriptor to truncate the string in one go. This method is fairly

advanced MMBasic and tweakers should be wary of changing the POKE commands. It is good to try to understand what is happening though.

```
    Function RTrim$(aa$)
        Local Integer n,x
        local q$
        q$=aa$
        For n=Peek(Var q$,0) TO 0 Step -1
            x=Peek(Var q$, n)
            If Not(x=&o40 Or x=&o11) Then Poke Var q$, 0, n: Exit For
        Next
        RTrim$=q$
    End Function
```

As with most optimizations resulting in an increased code footprint, there is a sweet-spot where the larger size can work against the workload. This seems to be a string of six spaces where the new version is only slightly faster than the old (the inverse is true).

Note also, the pad characters are described in octal which is the fastest radix for the interpreter.

Such a trick is not possible with LTrim$() because we would need to copy the useful part of the string down to start at position one in the string descriptor, likely voiding any gains and quite possibly making things a lot worse on large strings.