Long Strings Library

This is a library of embedded C routines which allow MMBasic on the Micromite and Micromite Plus to use and manipulate strings of unlimited length.

Standard strings in MMBasic are limited to a maximum length of 255 characters. This library duplicates the standard string functions but will work with strings of any length limited only by the amount of available RAM.

Long String Variables

Variables for holding long strings must be defined as floating point arrays. The long string routines do not keep numbers in these arrays but just use them as blocks of memory for holding long strings.

When creating these arrays they should be defined as single dimensioned float arrays with the number of elements set to one quarter the number of characters required for the maximum string length. For example, if three long string variables are required to hold a maximum of 2048 characters each, they should be defined thus:

```
CONST MaxLen = 2048
DIM FLOAT Str1(MaxLen/4),  Str2(MaxLen/4),  Str3(MaxLen/4)
```

These will contain empty strings when created (ie, their length will be zero).

When these variables are passed to the long string functions they should be entered as the variable name followed by empty brackets. For example:

```
SCopy  Str1(), Str2()
```

Note that the long string library routines do not check for overflow in the length of the strings. If an attempt is made to create a string longer than a long string variable's size the outcome will be undefined and could cause the Micromite to crash and restart.

Note that long string variables can be passed as arguments to user defined subroutines and functions. For example:

```
Sub MySub lstr() as float
  PRINT  "Long string length is" SLen(lstr())
END SUB
```

And it could be called like this:

```
MySub str1()
```

Library Routines

There are ten subroutines and five functions in the library and their definitions are listed below. Note the following features of the library:

- They require Version 5.2 or later of MMBasic.
- The parameters to most library routines are long string variables. In some cases a normal MMBasic string or a number is used and this is clearly documented.
- Many routines have source and destination parameters and they can be the same long string variable (although in the case of SCopy that would not achieve anything).
- Where a normal MMBasic string is used as an argument it can be a constant (ie, "Hello"), a normal string variable or a string expression (ie, "Hello" + "World").
- Where a character position in a string is specified or returned the numbering of the characters in the string start from number one.

SAddStr dest, src

Append a normal MMBasic string to a long string variable.

'dest' is a long string variable while 'src' is a normal MMBasic string expression.

Its main purpose is to convert normal MMBasic strings to long strings. For example:

```
SAddStr  Str3(), "This will be added to the end"
```

Another example is the following which will add the contents of a serial communications buffer to a long string variable:

```
DO WHILE LOC(#4) > 0
  SAddStr Str1(),INPUT(250, #4)
LOOP
```

s$ = SGetStr(src, start, len)

Returns part of a long string as a normal MMBasic string.

's$' is the MMBasic string returned by the function. 'src' is the source long string variable, 'start' is the character number in 'src' to start copying from (the first character is numbered one) and 'len' is the maximum number of characters to return. If there are no characters to return 's$' will be an empty string and if there are fewer characters than that 'len' then only they will be copied.

The following is an example of how to output a long string to a serial port using this function:

```
FOR i = 1 TO SLen(Str1()) STEP 250
  PRINT #4, SGetStr(Str1(), i, 250);
NEXT i
```

SCopy dest, src

Copy one long string to another.

'dest' is the destination variable and 'src' is the source variable. Whatever was in 'dest' will be overwritten.

SCat dest, src

Concatenate one long string to another.

'dest' is the destination variable and 'src' is the source variable. 'src' will the added to the end of 'dest' (the destination will not be overwritten).

SClear dest

Will clear the long string variable 'dest'. ie, it will be set to an empty string.

SLeft dest, src, nbr

Will copy the left hand 'nbr' characters from 'src' to 'dest' overwriting whatever was in 'dest'. ie, copy from the beginning of 'src'. 'dest' and 'src' must be long string variables. 'nbr' must be an integer constant or expression.

SRight dest, src, nbr

Will copy the right hand 'nbr' characters from 'src' to 'dest' overwriting whatever was in 'dest'. ie, copy from the end of 'src'. 'dest' and 'src' must be long string variables. 'nbr' must be an integer constant or expression.

SMid dest, src, start, nbr

Will copy 'nbr' characters from 'src' to 'dest' starting at character position 'start' overwriting whatever was in 'dest'. ie, copy from the middle of 'src'. 'nbr' is optional and if omitted the characters from 'start' to the end of the string will be copied. 'dest' and 'src' must be long string variables. 'start' and 'nbr' must be an integer constant or expression.

SUCase str

Will convert any lowercase characters in 'src' to uppercase. 'str' must be long string variable.

SULase str

Will convert any uppercase characters in 'src' to lowercase. 'str' must be long string variable.

SString dest, nbr, char

Will copy 'nbr' characters 'char' to the long string variable 'dest' overwriting whatever was in 'dest'. 'char' is the numeric ASCII value of the character to copy. 'nbr' and 'char' must be an integer constant or expression.

r = SCompare(str1, str2)

Compare the contents of two long string variables.

The returned value 'r' is an integer and will be -1 if 'str1' is less than 'str2'. It will be zero if they are equal in length and content and +1 if 'str1' is greater than 'str2'. The comparison uses the ASCII character set and is case sensitive. 'str1' and 'str2' must be long string variables.

r = SInstr(str, srch)

or

r = SInstr(str, srch, start)

Return the position of a substring in a long string.

The returned value ('r') is an integer and will be zero if the substring cannot be found. 'str' is the string to be searched and must be a long string variable. 'str' is the substring to look for and it must be a normal MMBasic string or expression (not a long string). The search is case sensitive.

Normally the search will start at the first character in 'str' but the second version allows the start position of the search to be specifies as the number 'start'. In both the returned value ('r') and the start position ('start') the first character in the string is number one.

For example, if the string `str` held "Hello world hello and hello":

```
PRINT SInstr(str, "worxx")       Would print 0
PRINT SInstr(str, "Hello")       Would print 1
PRINT SInstr(str, "hello")       Would print 13
PRINT SInstr(14, str, "hello")   Would print 23
```

r = SLen(str)

Will return the length of the string stored in the long string variable 'str'.

Adding the Library to a Program

To add the long string library to MMBasic you must insert the following code somewhere in your BASIC program (you can use copy and paste from this document). The exact spot is not important but at the end of the program is typical.

You only need to insert the particular routines that are used in your program.

```
CSub SAddStr float, string
    00000000
    10800011 00000000 10A0000F 00000000 90A30000 8C820000 00623021 1060000A
    AC860000 24420004 00822021 24A50001 00831821 90A20000 A0820000 24840001
    1483FFFC 24A50001 03E00008 00000000
End CSub


CFunction SGetStr(float, integer, integer) string
    00000000
    3C029D00 27BDFFE0 8C420040 AFB00018 AFA50010 AFA60014 AFBF001C 00808021
    0040F809 24040100 8FA50010 1200001E 8FA60014 10A0001D 8FBF001C 50C0001C
    8FB00018 8CA30000 5060001C A0400000 8E050000 00A3202B 14800017 240400FF
    8CC60000 2CC70100 24A50001 00C7200B 00A32823 00A4302B 0086280A 10A0000A
    A0450000 24630003 02038021 00452821 00401821 92040000 A0640001 24630001
    1465FFFC 26100001 8FBF001C 8FB00018 03E00008 27BD0020 A0400000 8FBF001C
    8FB00018 03E00008 27BD0020
End CFunction


CSub SCopy float, float
    00000000
    1080000C 00000000 10A0000A 2403FFFC 8CA20000 10430007 24830004 00621821
    90A20000 A0820000 24840001 1483FFFC 24A50001 03E00008 00000000
End CSub


CSub SCat float, float
    00000000
    10800010 00000000 10A0000E 00000000 8CA30000 8C820000 00623021 10600009
    AC860000 24420004 00822021 00A31821 90A20004 24A50001 A0820000 14A3FFFC
    24840001 03E00008 00000000
End CSub


CSub SClear float
    00000000 54800001 AC800000 03E00008 00000000
End CSub


CSub SLeft float, float, integer
    00000000
    10800012 00000000 10A00010 00000000 10C0000E 00000000 8CA20000 8CC30000
    0043302B 0046180B 10600008 AC830000 24840004 00A31821 90A20004 24A50001
    A0820000 14A3FFFC 24840001 03E00008 00000000
End CSub
```

```
CSub SRight float, float, integer
    00000000
    10800015 00000000 10A00013 00000000 10C00011 00000000 8CA20000 8CC30000
    0043302B 0046180B AC830000 1060000A 8CA20000 24420004 00431023 00A22821
    00831821 90A20000 A0820004 24840001 1483FFFC 24A50001 03E00008 00000000
End CSub


CSub SMid float, float, integer, integer
    00000000
    1080001F 00000000 10A0001D 00000000 10C0001B 00000000 50E0001D 3C07000F
    8CE70000 8CC20000 50400015 A0800000 8CA60000 00C2182B 14600013 24C60001
    00C23023 240300FF 2CC80100 0068300A 00E6182B 00E3300B 10C00009 AC860000
    24420003 00A22821 00861821 90A20000 A0820004 24840001 1483FFFC 24A50001
    03E00008 00000000 03E00008 A0800000 1000FFE4 24E7423F
End CSub


CSub SUCase float
    00000000
    1080000D 00000000 8C850000 10A0000A 00852821 90820004 2443FF9F 2C63001A
    10600002 2442FFE0 A0820004 24840001 5485FFF9 90820004 03E00008 00000000
End CSub


CSub SLCase float
    00000000
    1080000D 00000000 8C850000 10A0000A 00852821 90820004 2443FFBF 2C63001A
    10600002 24420020 A0820004 24840001 5485FFF9 90820004 03E00008 00000000
End CSub


CSub SString float, integer, integer
    00000000
    1080000E 00000000 10A0000C 00000000 10C0000A 00000000 8CA30000 10600007
    AC830000 00831821 90C20000 A0820004 24840001 5483FFFD 90C20000 03E00008
    00000000
End CSub


CFunction SCompare(float, float) integer
    00000000
    14800006 00000000 2404FFFE 2405FFFF 00801021 03E00008 00A01821 50A0FFFB
    2404FFFE 8C880000 15000008 8CA20000 10400021 00002021 2404FFFF 2405FFFF
    00801021 03E00008 00A01821 50400018 24040001 90830004 90A60004 24840004
    0066382B 14E0FFF4 24A50004 00C3182B 5460000F 24040001 00484023 2442FFFF
    1048FFEB 24840001 10400008 24A50001 90860000 90A30000 00C3382B 14E0FFE6
    0066182B 5060FFF6 2442FFFF 24040001 1000FFD7 00002821 1000FFD5 00002821
End CFunction


CFunction SInstr(float, string, integer) integer
    00000000
    10C00002 00005021 8CCA0000 50800027 00002021 50A00025 00002021 90A90000
    8C8B0000 01695823 016A102B 1440001E 008A2021 10000004 90AC0001 016A102B
    14400019 24840001 90820004 544CFFFB 254A0001 51200010 25440001 00801821
    10000006 24020001 90670005 91080000 15070006 24630001 00C01021 24460001
    0049382B 14E0FFF8 00A64021 5449FFEB 254A0001 25440001 00002821 00801021
    03E00008 00A01821 00002021 00002821 00801021 03E00008 00A01821
End CFunction
```

```
CFunction SLen(float) integer
    00000000
    10800005 00003821 8C860000 00C01021 03E00008 00E01821 00003021 00C01021
    03E00008 00E01821
End CFunction
```