Strings library for PIC18 processors

version 1.13

The library, prepared some time ago with yo2lio's help for PIC18 processors, has been ported to mP PRO and expanded quite a bit by adding several Delphi-like routines. It serves as a direct replacement of mE's string library file **__Lib_String.mcl**. Naturally, all the official strings lib routines are included (even those not presented in help) and some have an analogue added that works with constant strings or ROM (flash) memory:

memChr	
memCmp	
memCpy	**memCpyC**
memMove	
memSet	
StrCat	**StrCatC**
StrCat2	
StrChr	
StrCmp	
StrCpy	**StrCpyC**
StrLen	
StrnCat	
StrnCat2	
StrnCpy	**StrnCpyC**
StrSpn	
StrCSpn	
StrnCmp	
StrPbrk	
StrRchr	
StrStr	
Ltrim	
Rtrim	
strAppendPre	
strAppendSuf	
Length	**LengthC**

Differences compared to the official lib:

- *StrnCpy will always terminate the destination string with null char,*
- *Comparison functions, like memCmp, StrCmp, and StrnCmp, produce results of type integer, instead of short. That was also corrected lately in official lib.*
- *Rtrim procedure does not have the flaw of overwriting memory in the rare event where RAM contains 32 just before the string being corrected and the string is empty or contains only spaces.*
- *StrStr and StrCmp are free of errors the official lib routines contain.*

The replacement routines are more effective, both in size and in execution time, than the originals - that was the main reason to write them.

Other routines of the library are extensions of the official library or Delphi-like additions:

StrChrN
StrRchr
StrNchr
StrStrN

StrReplChr
StrInsertChr
StrRemoveChr
StrCutChr
StrReplace
StrSplit
StrAddCRLF

SetLength	
StringOfChar	
StrScan	**StrScanC**
Copy	**CopyC**
	CopyCL
StrLCopy	**StrLCopyC**
StrLCat	**StrLCatC**
StrEnd	
TrimLeft	
TrimRight	
Trim	
UpperCase	
LowerCase	
CompareStr	
StrLcomp	
Pos	**PosC**
PosEx	**PosExC**
Insert	
Delete	

Procedure CopyCL represents a new concept by making possible use of an 'array' of strings of different length. The 'array' may be thought of as text consisting of lines and is in fact declared as single string with LF char used as a separator, like

Code:
```
const txtc='ab'+#10+
           'cdef'+#10+
           'ghi';
```

Access to individual strings (lines) is granted by an index and procedure CopyCL copies chosen constant string to a string variable. Such approach is much more effective than using regular array of strings, which in mP have to be declared of equal length.

There is a version string declared, called Lib_String_ver (constant string [4]) that may be called from user code for verification.

Other notes:

- functions StrLen and Length are identical - use only one of them. Same is true for routines LTrim and TrimLeft, as well as for Rtrim and TrimRight.
- last corrections to the official lib made some of the C-like routines practically identical with the Delphi-like ones, so ,again, to save memory, please use only one of these.

Important: This version requires presence of Math library replacement.

Manual library installation:

- find mP PRO installation directory and subdirectory **.../Uses/P18**,
- find original library file **__Lib_string.mcl** there and rename it,
- unpack the replacement lib and move the **__Lib_string.mcl** file to the **.../Uses/P18** directory.

Have fun,
janni

Strings library for PIC18 processors
version 1.13
date 03.07.14
Revision history:

1.00 - adjusted for PRO 1.40, StrReplace corrected (ReplAll logic reversed)
1.01 - added Pascal routines: Length, LengthC, SetLength, StringOfChar, StrScan, StrScanC, Copy, CopyC,CopyCL, StrLCopy, StrLCopyC, StrLCat, StrLCatC, StrEnd, TrimLeft, TrimRight, Trim, UpperCase, LowerCase, CompareStr, StrLComp, Pos, Insert, Delete, and new official ones: Ltrim & Rtrim
1.02 - adjusted for PRO 1.50 beta
1.03 - order of parameters in StrAppendPre changed, parameters in mem__ routines changed
1.04 - adjusted to newest cmd line compiler, changed memCpyC parameter p1 to ^byte
1.05 - FSR loading moved to assembly
1.06 - updated for new (4.10) compiler version
1.07 - updated for new (4.15) compiler version; added PosEx, PosExC; more extensions for strings longer than 255 chars
1.08 - memMove corrected
1.09 - optimised Pos, PosC, PosEx, PosExC, StrStr, and StrStrN
1.10 - compiled with v4.80 beta
1.11 - compiled with v5.00
1.12 - corrected StrStrN, PosEx, PosExC
1.13 - corrected StrStrN, PosEx, PosExC, compiled with v6.40

Library routines:

procedure memSet(p:^byte; ch:char; n:word);
fills the first n bytes of memory with ch, starting at address p

procedure memCpy(p1,p2:^byte; n:word);
copies n bytes starting from memory address p2 to memory area beginning at p1. If these memory buffers overlap, the memcpy function cannot guarantee that bytes are copied before being overwritten. If these buffers do overlap, use the memMove function.

procedure memCpyC(p1:^byte; p2:dWord; n:word);
copies n bytes starting from Flash address p2 to RAM memory area beginning at p1

procedure memMove(p1,p2:^byte; n:word);
copies n bytes starting from memory address p2 to memory area beginning at p1. Memory buffers may overlap.

function memCmp(p1,p2:^byte; n:word): integer;
compares two memory areas starting at addresses p1 and p2 for n bytes and returns a value indicating their relationship as follows:

Value	Meaning
< 0	p1 "less than" p2
= 0	p1 "equal to" p2
> 0	p1 "greater than" p2

The returned value is determined by the difference between the values of the first pair of bytes that differ in the areas being compared.

function memChr(p:^byte; ch:char; n:word): word;
locates the first occurrence of ch in the initial n bytes of memory area starting at address p. Function returns the offset of this occurrence from the memory address p or $FFFF (maxWord) if no instance of ch was found.

function StrLen(var inst:string):word;
calculates length of inst (without the null terminating character)

function Length(var S:string):word;
calculates length of inst (without the null terminating character)

function LengthC(atS:^ const char):word;
calculates length of constant string atS (without the null terminating character)

procedure SetLength(var S:string; NewLength: word);
sets the string termination character at proper place preserving the string contents up to this position - WARNING: the contents may be undefined

procedure StringOfChar(var S:string; ch:char; Count:word);
fills S with Count of ch characters and adds string termination character

procedure StrCat(var S1,S2**:string);**
concatenates S2 to S1

procedure StrCatC(var S1:string; S2:^const char);
concatenates constant string from ROM at address S2 to string S1

procedure StrnCat(var S1,S2**:string; n:byte);**
concatenates at most n characters of S2 to S1

procedure StrCat2(var l1,S1,S2:string);
copies S1 and S2 into l1; added for compatibility with original library

procedure StrnCat2(var p1,p2:string);
identical to StrCat; introduced for compatibility with original library

procedure StrLCat(var S1,S2:string; MaxLen:word);
appends up to a specified maximum number of characters from S2 to S1

procedure StrLCatC(var S1:string; S2:^const char; MaxLen:word);
appends at most MaxLen characters from constant string at address S2 to string S1

procedure StrSplit(var S1,S2:string; n:word);
splits S1 in 2 strings: S1 - from 0 to n-1 and S2 - from n to end

procedure StrAppendSuf(var S:string; ch:char);
adds ch at the end of inst

procedure StrAppendPre(ch:char; var S:string);
inserts ch at the beginning of S

procedure StrAddCRLF(var S:string);
appends CR+LF to the input string, S

procedure StrInsertChr(var S:string; ch:byte; n:word);
inserts character at specified position - after n-th character (after insertion index of the new character will equal n and length of string will increase by 1)

procedure StrRemoveChr(var S:string; n:word);
removes character of specified index n from S (after removal the length of string will decrease by 1)

function StrScan(var S:string; ch:char):boolean;
returns FSR0Ptr pointer to first occurrence of a specified character in a string S. If ch does not occur in S, StrScan returns false with FSR0Ptr pointing to next memory location after the null terminating char. The null terminator is considered to be part of the string.

function StrScanC(atS:^const char; ch:char):boolean;
returns TBLPTR pointing to first occurrence of a specified character in a constant string atS. If ch does not occur in S, StrScanC returns false with TBLPTR pointing to next memory location after the null terminating char. The null terminator is considered to be part of the string.

procedure StrCpy(var S1,S2:string);
copies S2 to S1

procedure StrCpyC(var S1:string; S2:^const char);
copies constant string at S2 to S1

procedure StrnCpy(var S1,S2:string; size:word);
copies at most size characters from S2 to S1; if S2 contains fewer characters than n, only available characters are copied. Unlike in the official version, S1 is always terminated with null char

procedure StrnCpyC(var S1:string; S2:^const char; n:byte);
copies at most first n characters from constant string at S2 to S1; S1 is always terminated with null char

procedure Copy(var Dest,Source:string; Index,Count: word);
copies Count characters from Source to Dest starting at Source[Index]. If Index is larger than the length of Source, Copy returns empty Dest string. If Count specifies more characters than are available, only the characters from Source[Index] to the end of Source are copied.

procedure CopyC(var Dest:string; Source:^const char; Index,Count: word);
copies Count characters from constant Source to Dest starting at Source[Index]. If Index is larger than the length of Source, CopyC returns empty Dest string. If Count specifies more characters than are available, only the characters from Source[Index] to the end of Source are copied.

procedure CopyCL(var Dest:string; Source:^const char; LineNo: byte);
copies line of the index LineNo (starting from 0) from constant Source to Dest string. If there is no line of this number, CopyCL returns empty Dest string.

procedure StrLCopy(var Dest,Source:string; MaxLen:word);
copies at most Maxlen characters from Source to Dest

procedure StrLCopyC(var Dest:string; Source:^const char; MaxLen:word);
copies at most MaxLen characters from constant at Source to Dest

procedure StrEnd(var S:string);
returns FSR0Ptr pointing to the null character at the end of S

procedure StrCutChr(var S:string; ch: byte);
removes all characters equal ch from front of the string

procedure Ltrim(var S: string);
trims the leading spaces of the input string

procedure Rtrim(var S: string);
trims the trailing spaces of the input string.

procedure TrimLeft(var S: string);
trims the leading spaces of the input string

procedure TrimRight(var S: string);
trims the trailing spaces of the input string.

procedure Trim(var S: string);
trims the leading and up to 255 trailing spaces of the input string
procedure UpperCase(var S: string);
converts S to uppercase. Character values not in the range a..z are unaffected

procedure LowerCase(var S: string);
converts S to lowercase. Character values not in the range A..Z are unaffected

function StrChr(var S:string; ch:char):word;
returns the position (index) of the first character ch found in S; if no matching character was found, function returns $FFFF (maxWord); the null character terminating S is included in the search.

function StrChrN(var S:string; ch:char; n:word):word;
operates as StrChr but starts search from index n in S; n should be smaller than length of S.

function StrRchr(var S:string; ch:char):word;
same as StrChr but last occurence is reported

function StrNchr(var S:string; ch:char):word;
returns number of occurences of ch in S

procedure StrReplChr(var S:string; ch1,ch2: char);
replaces all occurences of ch1 with ch2 in S

function StrCmp(var S1,S2:string):integer;
lexicographically compares the contents of strings and returns a value indicating their relationship:

Value	Meaning
< 0	S1 "less than" S2
= 0	S1 "equal to" S2
> 0	S1 "greater than" S2

The value returned by function is determined by the difference between the values of the first pair of bytes that differ in the strings being compared. Terminating char (0) is taken into account thus allowing to compare string lengths, if this is the only difference.

function StrnCmp(var S1,S2:string; n:word):integer;
same as StrCmp, but only for first n characters of both strings

function CompareStr(var S1,S2:string):integer;
compares S1 to S2, with case-sensitivity. The return value is less than 0 if S1 is less than S2, 0 if S1 equals S2, or greater than 0 if S1 is greater than S2. The compare operation is based on the 8-bit ordinal value of each character. Terminating char (0) is taken into account thus allowing to compare string lengths, if this is the only difference.

function StrLComp(var S1,S2:string; MaxLen:word):integer;
same as CompareStr but compares at most MaxLen chars of S1 to S2

function StrSpn(var S1,S2:string):word;
computes the length of the maximum initial segment of the string S1 which consists entirely of the characters from the string S2; function returns the length of the segment

function StrCSpn(var S1,S2:string):word;
computes the length of the maximum initial segment of the string S1 which consists entirely of characters not from the string S2; function returns the length of the segment

function StrStr(var S1,S2:string):word;
locates the first occurrence of the string S2 in the string S1 (excluding the terminating null character); function returns a number indicating the position of the first occurrence; if no string was found or S2 is a null string, function returns $FFFF (maxWord).

function StrStrN(var S1,S2:string; n:word):word;
performs the same task as StrStr but from index n in S1; n should be smaller than length of S1

function Pos(var Substr,S:string):word;
searches for Substr within S and returns a value that is the index of the first character of Substr within S; if Substr is not found or Substr is a null string, function returns $FFFF (maxWord).

function PosC(var Substr:string; atS:^const char):word;
searches for Substr within constant atS and returns a value that is the index of the first character of Substr within string atS; if Substr is not found or Substr is a null string, function returns $FFFF (maxWord).

function PosEx(var Substr,S:string; Offset:word):word;
searches for Substr within S starting from Offset and returns a value that is the index of the first character of Substr within S; if Substr is not found or Substr is a null string, function returns $FFFF (maxWord); Offset should be smaller than length of S.

function PosExC(var Substr:string; atS:^const char; Offset:word):word;
searches for Substr within string constant atS starting from Offset and returns a value that is the index of the first character of Substr within string atS; if Substr is not found or Substr is a null string, function returns $FFFF (maxWord); Offset should be smaller than length of the string constant.

procedure Insert(var Source,S:string; Index:word);
Insert merges Source into S at the position S[index]. Length of S will be adjusted if necessary

procedure Delete(var S:string; Index,Count:word);
removes a substring of Count characters from string S starting with S[Index]. If Index is larger than the length of S, no characters are deleted. If Count specifies more characters than remain starting at the S[Index], Delete removes the rest of the string.

function StrReplace(var S1:string; S2,S3:string[255]; ReplAll:boolean):byte;
depending on ReplAll, replaces first or all occurrences of S2 in S1 with S3; result gives the number of replacements made

function StrPbrk(var S1,S2:string):word;
searches S1 for the first occurrence of any character from S2; returns an index of the matching character in S1; if S1 contains no characters from S2, function returns $FFFF (maxWord).