Conversions library for PIC18 processors

version 1.14

The library, prepared some time ago with yo2lio's help for PIC18 processors, has been now ported to mP PRO, optimised and a bit expanded - this time it's prepared as a direct replacement of the official Conversions library (__Lib_Conversions.mcl). Due to use of assembly, the library produces smaller and faster (up to several times) code than the official one and is also more extensive, introducing new functions and procedures. For those that want just smaller and faster code, but do not want to bother with changing names of routines, the library contains all routines present in the official version. Important: this library requires presence of another replacement lib, namely the strings library.

Routines added to the library do not follow the way the official routines work. For example, procedure Word2str converts number of type word to string of minimal length - not, like the official one, WordToStr, always to string of length 5. If one needs such formatting, one may use more general procedure, Word2strN, which may produce string of any (reasonable) length with spaces filling the beginning of string. (Word2strN('123',st,5) gives st:=' 123' just like WordToStr.) The advantage of such approach is that strings of the same length may be produced (and displayed in simple way) from variables of different types, like:

```
Byte2strN(b,st,6);
Word2strN(w,st,6);
Int2strN(i,st,6);
```

If one needs to, one may replace spaces with other characters (like zeroes) using procedure StrReplChr from Strings library replacement (or use ready procedures, like Byte2strWithZeros). Additionally, Conversions lib routines converting strings to numbers detect errors and set the result to maximum value, if error occurs. They are also flexible and accept strings that only end with numbers – no need to 'clean' the string before conversion. For cases where it's necessary to ensure that a string contains only a number, function strIsNumber is provided.

To differentiate from the official lib routines, different naming convention was assumed for the added ones. Here is the list of additional routines in conversions library:

Byte2Hex	**Hex2Byte**
Word2Hex	**Hex2Word**
dWord2Hex	
Byte2Bin	
Word2Bin	
dWord2Bin	
Bcd2Dec_8	**Dec2Bcd_8**
Bcd2Dec_16	**Dec2Bcd_16**

strU	**Val**
strS	
Byte2str	**str2byte**
Byte2strN	
Byte2strWithZeros	
Short2str	**str2short**
Short2strN	
Short2strWithZeros	
Word2str	**str2word**
Word2strN	
Word2strWithZeros	
Int2str	**str2int**
Int2strN	
Int2strWithZeros	
long2str	**str2long**
Long2strN	
Long2strWithZeros	
dWord2str	**str2dWord**
dWord2strN	
dWord2strWithZeros	
Float2str	**str2float**
strIsNumber	
strAddSpaces	
IP2str	**str2IP**
MAC2str	**str2MAC**

You'll find their descriptions below (or click on a name of a routine to jump to it's prototype).

Innovation of the library is introduction of function Val and procedures str (there are separate routines for signed and unsigned numbers) resembling those from Delphi.

Function Val converts string of length 1-11 to number of any type (but real) - defined by assignment instruction, like

```
byte_var:=Val('123',Res);
integer_var:=Val('-1234',Res);
dWord_var:=Val('12345678',Res);
```

Res (variable of type byte), like in Delphi, is zero when conversion is successful, indicates position of unrecognizable character (if spotted), or equals $FF, if the number is out of dWord type range, or negative limit for Longint type.

If one uses conversions from string to numbers of different type, Val is the perfect choice, saving code space. On the other hand, if one uses only one type of variables, code will be faster when function corresponding to the type is used (like str2byte for numbers ranging from 0 to 255).

The str procedures, strU & strS, convert any (but real) unsigned/signed number to string using only required string space (i.e. working with bytes one may use just string[3] as output in strU). One may also wish to obtain strings of specified length, like

```
strU(number, output_string, 15);
```

where the number in string will be preceded by spaces.

Hopefully, some day mE will include Val and str in the compiler and there'll be just one str procedure. Compiler may easily determine the type of used variable, which is not possible in a procedure.

The library declares a version string, called Lib_Conversions_ver (constant string [4]) that may be called from user code for verification.

For those that use Florin's Ethernet libs a small library was added that should be used instead of his Additional Strings Library (which cannot be used in parallel with the Conversions lib). It contains few routines that differ in naming from the Strings lib replacement and are needed by Ethernet libs. It's naturally called the same: pic_additional_string_library.mcl.

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_conversions.mcl** there and rename it,
- unpack the replacement lib and move the **__Lib_conversions .mcl** file to the **.../Uses/P18** directory.

Have fun,
janni

String conversions library for PIC18 processors

version 1.14
date 20.07.13
Revision history:

1.00 - adjusted for PRO 1.50 beta
1.01 - rewritten as replacement for official lib and adjusted for strings lib v 1.03
1.02 - changed str2byte, str2short, str2word, str2int, Val; added StrIsNumber;
optimised Word2str & dWord2str
1.03 - corrected str2IP
1.04 - FSR loading moved to assembly,
optimised Bcd2Dec_16
1.05 - corrected str2float, upgraded for ver>3.2
1.06 - corrected str2short, str2int & Val
1.07 - added dWord2Hex, series of ..ToHex and ..ToStringWithZeroes
replacements for official lib routines (mP v 4.10)
1.08 – corrected mistake in Short-, Int- & Longint2StrWithZeros procedures
changed strAddSpaces; minor parameter changes
1.09 – changed Float2str (and FloatToStr), corrected IP2str & str2.. routines
1.10 - compiled for v4.80 beta
1.11 – corrected Byte2str procedure
1.12 – compiled with mP PRO v5.00
1.13 – compiled with v5.61, corrected Float2str
1.14 – modified Byte2str, Word2Str, dWord2Str, Dec2Bcd_8, Dec2Bcd_16

Library routines:

------------------- official lib replacements -------------

procedure ByteToHex(input: byte; var output: string[2]);

procedure ShortToHex(input: short; var output: string[2]);

procedure WordToHex(input: word; var output: string[4]);

procedure IntToHex(input: integer; var output: string[4]);

procedure dWordToHex(input: dWord; var output:string[8]);

procedure LongWordToHex(input: dWord; var output:string[8]);

procedure LongIntToHex(input: longint; var output: string[8]);

procedure ByteToStr(input: byte; var output: string[3]);

procedure ShortToStr(input: short; var output: string[4]);

procedure WordToStr(input: word; var output: string[5]);

procedure WordToStrWithZeros(input: word; var output: string[5]);

procedure IntToStr(input: integer; var output: string[6]);

procedure IntToStrWithZeros(input: integer; var output: string[6]);

procedure dWordToStr(input: dWord; var output:string[10]);

procedure dWordToStrWithZeros(input: dWord; var output:string[10]);

procedure LongWordToStr(input: dWord; var output:string[10]);

procedure LongWordToStrWithZeros(input: dWord; var output:string[10]);

procedure LongintToStr(input: longint; var output: string[11]);

procedure LongintToStrWithZeros(input: longint; var output: string[11]);

procedure FloatToStr(input: real; var output: string[17]);
(with the replacement of MathDouble lib, better use the Real2str procedure placed in Trigon replacement)

function StrToInt(var input:string[6]): integer;

function StrToWord(var input:string[5]): word;

function Bcd2Dec(bcdnum: byte): byte;

function Dec2Bcd(decnum: byte): byte;

function Bcd2Dec16(bcdnum : word): word;

function Dec2Bcd16(decnum: word): word;

------------------- representation conversion -------------

procedure Byte2Bin(number:byte; var output:string[8]);
converts byte/short to binary representation in output string

procedure Word2Bin(number:word; var output:string[16]);
converts word/integer to binary representation in output string

procedure dWord2Bin(number:dword; var output:string[32]);
converts dword/longint to binary representation in output string

procedure Byte2Hex(number:byte; var output:string[2]);
converts byte/short to hex representation in output string

function Hex2Byte(var hex_in:string[2]):byte;
converts hexadecimally coded number (2 chars) in hex_in to byte; accepts lower- and uppercase letters

procedure Word2Hex(number:word; var output:string[4]);
converts word/integer to hex representation in output string

function Hex2Word(var hex_in:string[4]):word;
converts hexadecimally coded number (4 chars) in hex_in to word; accepts lower- and uppercase letters

procedure dWord2Hex(number:dWord; var output:string[8]);
converts dWord/longint to hex representation in output string

function Bcd2Dec_8(number: byte):byte;
converts BCD coded number to byte

function Dec2Bcd_8(number: byte):word;
codes number of type byte (0..255) in BCD

function Bcd2Dec_16(number: word):word;
converts BCD coded number to word

function Dec2Bcd_16(number: word):dWord;
codes number of type word (0..65535) in BCD

-------------------- number to string ---------------------
procedure strU(number:dWord; var output:string; len:byte);
converts any (but real) unsigned number to string using only required string space, i.e. working with bytes one may use just string[3] as output; len is the minimum resulting string length - if larger then the number representation, the number will be preceded by spaces

procedure strS(number:longint; var output:string; len:byte);
converts any (but real) signed number to string using only required string space, i.e. working with short type one may use just string[4] as output; len is the minimum resulting string length - if larger then the number representation, the number will be preceded by spaces

procedure strAddSpaces(var output:string; len:byte; minus:boolean);
inserts spaces at the beginning of output up to length indicated by len; minus sign is added at front if minus=true; the resulting string will have length of at least len characters

procedure Byte2str(number: byte; var output: string[3]);
converts number of type byte to string containing up to 3 characters; uses only necessary string space

procedure Byte2strN(number:byte; var output:string; len:byte);
converts number of type byte to string containing minimum len characters - spaces are used to fill the string before representation of number

procedure Byte2strWithZeros(number:byte; var output:string[3]);
converts number of type byte to string of three characters - zeros fill the string before representation of number

procedure Short2str(number:short; var output:string[4]);
converts number of type short to string containing up to 4 characters; uses only necessary string space

procedure Short2strN(number:short; var output:string; len:byte);
converts number of type short to string containing minimum len characters - spaces are used to fill the string before representation of number

procedure Short2strWithZeros(number:short; var output:string[4]);
converts number of type short to string of four characters - zeros (with minus sign, if negative) fill the string before representation of number

procedure Word2str(number:word; var output:string[5]);
converts number of type word to string containing up to 5 characters; uses only necessary string space

procedure Word2strN(number:word; var output:string; len:byte);
converts number of type word to string containing minimum len characters - spaces are used to fill the string before representation of number

procedure Word2strWithZeros(number:word; var output:string[5]);
converts number of type word to string of five characters - zeros fill the string before representation of number

procedure Int2str(number:integer; var output:string[6]);
converts number of type integer to string containing up to 6 characters; uses only necessary string space

procedure Int2strN(number:integer; var output:string; len:byte);
converts number of type integer to string containing minimum len characters - spaces are used to fill the string before representation of number

procedure Int2strWithZeros(number:integer; var output:string[6]);
converts number of type integer to string of six characters - zeros (with minus sign, if negative) fill the string before representation of number

procedure dWord2str(number:dWord; var output:string[10]);
converts dWord number to string using only necessary string space - up to 10 characters

procedure dWord2strN(number:dWord; var output:string; len:byte);
converts number of type dWord to string containing minimum len characters - spaces are used to fill the string before representation of number

procedure dWord2strWithZeros(number:dWord; var output:string[10]);
converts number of type dWord to string of ten characters - zeroes fill the string before representation of number

procedure Long2str(number:longint; var output:string[11]);
converts Longint number to string using only necessary string space - up to 11 characters

procedure Long2strN(number:longint; var output:string; len:byte);
converts number of type longint to string containing minimum len characters - spaces are used to fill the string before representation of number

procedure Long2strWithZeros(number:longint; var output:string[11]);
converts number of type longint to string of eleven characters - zeros (with minus sign, if negative) fill the string before representation of number

procedure Float2Str(number: real; var output: string[17]; fract: byte);
converts value of type real to string of up to 17 characters; fract is the number of decimal places (with the replacement of MathDouble lib, better use the Real2str procedure placed in Trigon replacement lib)

-------------------- string to number ---------------------

All functions of this type (except Val) return maximum positive value of given type number (FF for byte, etc.), if conversion was not successful. Strings of size longer than specified in given function will also be accepted if the number is preceded by non-number characters (like space). Any characters may appear before the '-' or '+' sign at the beginning of a number. To check whether a string contains only a number, use StrIsNumber function. Result of str2float is unpredictable, if conversion was unsuccessful.

function strIsNumber(var Str_in:string[20]): boolean;
checks whether S consists entirely of chars from ['0'..'9','+','-']; '+' or '-' may appear only as first character.

function Val(var Str_in:string[11]; var Res:byte):longint;
converts string of length 1-11 to number of any type; Res equals zero if the conversion is successful; if the number is out of dWord type range, or negative limit for Longint, or Str_in is empty, Res=255; if unrecognizable character is spotted, Res will give it's position in the input string (index+1) - this is in practice true only for last char as Val tolerates non-numeric characters preceding number

function str2byte(var Str_in:string[3]):byte;
converts string of length 1-3 to number of type byte

function str2short(var Str_in:string[4]):short;
converts string of length 1-4 to number of type short

function str2word(var Str_in:string[5]):word;
converts string of length 1-5 to number of type word

function str2int(var Str_in:string[6]):integer;
converts string of length 1-6 to number of type integer

function str2dWord(var Str_in:string[10]):dWord;
converts string of length 1-10 to number of type str2dWord (this function uses Val, so direct use of the latter is advised)

function str2long(var Str_in:string[11]):longint;
converts string of length 1-11 to number of type Longint (this function uses Val, so direct use of the latter is advised)

function str2float(var Str_in: string[17]): real;
converts string of length 1-17 to number of type real (with the replacement of MathDouble lib, better use the str2real procedure placed in Trigon replacement lib)

------------------- other conversions ---------------------------

procedure IP2str(var IP_in: array[4] of byte; var Str_out: String[15]);
converts IP address (array[4] of byte) into string. Lenght of Str_out may vary from 7 to 15.

procedure MAC2str(var MAC_in: array[6] of byte; var Str_out: String[12]);
converts MAC (array[6] of byte) into String[12].

procedure str2IP(var Str_in: String[15]; var IP_out :array[4] of byte);
converts Str_in (of length from 7 to 15 and including three '.') to IP address

procedure str2MAC(var Str_in: String[12]; var MAC_out: array[6] of byte);
converts Str_in (of length 12) to MAC address