

Additional String Util Libraries

- **Procedure mem_cpy(addr1, addr2, n : word);**
// Copies n bytes from the memory area starting at the address addr2 to addr1.
// Ex. mem_cpy(@data1, @data2, 4);

- **Function mem_Cmp(addr1,addr2, n : word): integer;**
// compares two memory areas starting at addresses addr1 and addr2 for n bytes
// and return 0 if equal or difference if is not.
// Ex. result16s := mem_cmp(@data1, @data2, 4);

- **Procedure mem_set(addr1 : word; value : byte; n : word);**
// Fill n bytes from memory starting at address addr1 with value.
// Ex. mem_set(@data1, 255, 4);

- **Function mem_chr(addr1 : word; chr : byte; n : word) : word;**
// Search for chr in first n bytes from memory starting at address addr1 and
// return position index.
// Ex. result16 := mem_chr(@data1, '4', 7);

- **Function Str_Len(var data_Str : String[4095]) : word;**
// Returns the length, in bytes, of the String data_Str
// Ex. result16 := Str_Len(data1);

- **Procedure Str_Cut_Chr(var data_Str : String[4095]; chr_ : byte);**
// Remove all specified chr from front of the String data_Str
// Ex. Str_Cut_Chr(data1, '');

- **Procedure Str_Cat(var data_Str1, data_Str2 : String[4095]);**
// Appends data_Str2 to data_Str1.
// Ex. Str_Cat(data1,data2);

- **Function Str_Cpy(var data_Str1, data_Str2 : String[4095]) : word;**
// Copies the value of String data_Str2 to the String data_Str1
// Return number of copied elements
// Ex. Str_Cpy(data1, data2);

- **Function Str_Chr(var data_Str : String[4095]; chr_ : byte) : word;**
// Returns the position of the first character chr found in data_Str
// Ex. result16 := Str_Chr(data1, '6');

- **Procedure Str_Replace_Chr(var data_Str : String[4095]; chr1, chr2 : byte);**
// Replase all chr1 with chr2 in data_Str
// Ex. Str_Replase_Chr(data1,','0');

- **Procedure Str_Split(var inst1,inst2 : string[4095]; n : word);**
// Split inst1 in 2 strings after n char
- **Procedure Str_Insert_Chr(var inst1 : string[4095]; chr_ : byte; n : word);**
// Insert Chr at position n
- **Procedure Str_AppendSuf(var str_ : string[4095]; ch : char);**
// adds ch at the end of str_
- **Procedure Str_AppendPre(var str_ : string[4095]; ch : char);**
// inserts ch at the beginning of str_
- **Procedure Byte2StrWithZeros(data_in : byte; var data_Str : String[3]);**
// Convert byte value to String[3].
// Ex. Byte2StrWithZeros(10, data1); // data1 will be '010'
- **Procedure Byte2Str(data_in : byte; var data_Str : String[3]);**
// Convert byte value to String, lenght of String is variable from 1 to 3.
// Ex. Byte2Str(10, data1); // data1 will be '10'
- **Procedure Short2StrWithZeros(data_in : short; var data_Str : String[4]);**
// Convert short value to String[4].
// Ex. Short2StrWithZeros(-10, data1); // data1 will be '-010'
- **Procedure Short2Str(data_in : short; var data_Str : String[4]);**
// Convert short value to String, lenght of String is variable from 1 to 4.
// Ex. Short2Str(-10, data1); // data1 will be '-10'
- **Procedure Word2StrWithZeros(data_in : word; var data_Str : String[5]);**
// Convert word value to String[5].
// Ex. Word2StrWithZeros(645, data1); // data1 will be '00645'
- **Procedure Word2Str(data_in : word; var data_Str : String[5]);**
// Convert word value to String, lenght of String is variable from 1 to 5.
// Ex. Word2Str(645, data1); // data1 will be '645'
- **Procedure Int2StrWithZeros(data_in : integer; var data_Str : String[6]);**
// Convert integer value to String[6].
// Ex. Int2StrWithZeros(-645, data1); // data1 will be '-00645'
- **Procedure Int2Str(data_in : integer; var data_Str : String[6]);**
// Convert integer value to String, lenght of String is variable from 1 to 6.
// Ex. Int2Str(-645, data1); // data1 will be '-645'

- **Procedure LongWord2StrWithZeros(data_in : dWord; var data_Str : String[10]);**
// Convert dWord value to String[10].
// Ex. LongWord2StrWithZeros(11645, data1); // data1 will be '0000011645'

- **Procedure LongWord2Str(data_in : dWord; var data_Str : String[10]);**
// Convert dWord value to String, length of String is variable from 1 to 10.
// Ex. LongWord2Str(11645, data1); // data1 will be '11645'

- **Procedure Long2StrWithZeros(data_in : longint; var data_Str : String[11]);**
// Convert longint value to String[11].
// Ex. Long2StrWithZeros(-11645, data1); // data1 will be '-0000011645'

- **Procedure Long2Str(data_in : longint; var data_Str : String[10]);**
// Convert longint value to String, length of String is variable from 1 to 11.
// Ex. Long2Str(-11645, data1); // data1 will be '-11645'

- **Procedure Float2Str(data_in : Real; var data_Str : String[17]; digits : byte);**
// Convert Real value to String, length of String is variable from 1 to 17, max digits 4.
// Ex. Float2Str(-116.12345, data1, 0); // data1 will be '-116'
// Ex. Float2Str(-116.12345, data1, 3); // data1 will be '-116.123'
// Ex. Float2Str(-116.12345, data1, 4); // data1 will be '-116.1234'

- **Procedure Byte2Hex(data_hex : byte; var hex : String[2]);**
// Convert Byte to Hex
// Ex. Byte2Hex(255, data_out);

- **Function Hex2Byte(var hex : String[2]) : byte;**
// Convert Hex String format (must be String[2]) into Byte
// Ex. res_8 := Hex2Byte('AA');

- **Procedure Word2Hex(data_hex : word; var hex : String[4]);**
// Convert Word to Hex
// Ex. Word2Hex(1000, data_out);

- **Function Hex2Word(var hex : String[4]) : word;**
// Convert Hex String format (must be String[4]) into Word
// Ex. res_16 := Hex2Word('AA55');

- **Function Str2Byte(var byte_in : String[3]) : byte;**
// Convert Byte String format (must be String[1] to String[3]) into Byte
// Ex. res_8 := Str2Byte('6');
// Ex. res_8 := Str2Byte('66');
// Ex. res_8 := Str2Byte(' 66');
// Ex. res_8 := Str2Byte(' 6');

```

- Function Str2Word(var word_in : String[5]) : word;
  // Convert Word String format (must be String[1] to String[5]) into Word
  // Ex. res_16 := Str2Word('9');
  // Ex. res_16 := Str2Word(' 669');
  // Ex. res_16 := Str2Word('96');
  // Ex. res_16 := Str2Word(' 669');

- Function Str2LongWord(var byte_in : String[10]) : dWord;
  // Convert dWord String format (must be String[1] to String[10]) into dWord
  // Ex. res_32 := Str2LongWord('1234567890');

- Function Str2Short(var byte_in : String[4]) : short;
  // Convert Short String format (must be String[1] to String[4]) into short
  // Ex. res_8s := Str2Short('-6');

- Function Str2Int(var word_in : String[6]) : integer;
  // Convert Word String format (must be String[1] to String[6]) into integer
  // Ex. res_16s := Str2Int('-12345');

- Function Str2Long(var byte_in : String[11]) : Longint;
  // Convert Longint String format (must be String[1] to String[11]) into Longint
  // Ex. res_32s := Str2dLong('-1234567890');

- Function Str2Float(var byte_in : String[17]) : real;
  // Convert Float String format (must be String[1] to String[17]) into Float,
  // sign + 10bytes + '.' + 5 bytes
  // Ex. res_float := Str2Float('-123.12345');

- Procedure IP2Str(var user_IPaddr : array[4] of byte; var Str_out : String[15]);
  // Convert IP ( array[4] of byte ) into String. Length of String is variable from 7 to 15.
  // Ex. IP2Str(IP_address,data_out);

- Procedure MAC2Str(var MAC_addr : array[6] of byte; var Str_out : String[12]);
  // Convert MAC ( array[6] of byte ) into String[12].
  // Ex. MAC2Str(MAC_address, data_out);

- Procedure Str2IP(var Str_in : String[15]; var user_IPaddr : array[4] of byte);
  // Convert String (length of String must be from 7 to 15 and must include 3 '.')
  // to array[4] of byte
  // Ex. Str2IP('192.168.1.155', IP_address);

- Procedure Str2MAC(var Str_in : String[12]; var MAC_addr : array[6] of byte);
  // Convert String (length of String must be 12) to array[6] of byte
  // Ex. Str2MAC('00AA80FF457F', MAC_address);

```

- **Procedure Str2IP_(var Str_in : String[15]; IPaddr_ : word);**
// Convert String (length of String must be from 7 to 15 and must include 3 '.')
// to array[4] of byte and save to RAM starting at address IPaddr_
// Ex. Str2IP_('192.168.1.155', @IP_address);

- **Procedure Str2MAC_(var Str_in : String[12]; MAC_addr_ : word);**
// Convert String (length of String must be 12) to array[6] of byte
// and save to RAM starting at address MAC_addr_
// Ex. Str2MAC_('00AA80FF457F', @MAC_address);

- **Function Bcd2Dec(number : byte) : byte;**

- **Function Bcd2Dec16(number : word) : word;**

- **Function Dec2Bcd(number : byte) : byte;**

- **Function Dec2Bcd16(number : word) : word;**

- **Procedure PIC_additional_string_library_version(var version : string[\$FF]);**
// return version of library
// Ex. PIC_additional_string_library_version(data_out);
// data_out will be 'PIC_A_S_L V4.0 12-May-2008'

- **Procedure CpyFlash2Mem(Faddr : dword; Maddr_,nb : word);**
// Copies nb bytes from the Flash area starting at the address Faddr
// to RAM address Maddr_.
// Ex. CpyFlash2Mem(@cd1,@data1,3);

- **Function CpyFlashString2Mem(Faddr : dword; Maddr_ : word) : word;**
// Copies String from the Flash area starting at the address Faddr to RAM area starting
// at address Maddr_.
// Return number of copied elements
// Ex. CpyFlashString2Mem(@cd1,@data1);

- **Function CmpFlashWithMem(Faddr : dword; Maddr_,nb : word) : integer;**
// Compare nb bytes from the Flash area starting at the address Faddr with RAM area
// starting at address Maddr_, return 0 if equal or difference if not .
// Ex. result16s := CmpFlashWithMem(@cd1,@data1,3);

- **Function CmpFlashStringWithMem(Faddr : dword; Maddr_ : word) : integer;**
// Compare String from the Flash area starting at the address Faddr with RAM area
// starting at address Maddr_, return 0 if equal or difference if is not .
// Ex. result16s := CmpFlashStringWithMem(@cd1,@data1);

- Function CmpFlashWithFlash(Faddr1,Faddr2,nb : dword) : integer;
// Compare two Flash memory areas starting at addresses Faddr1 and Faddr2 for
// nb bytes, return 0 if equal or difference if not .
// Ex. result16s := CmpFlashWithFlash(@cd1,@cd2,5);

Author : Florin Andrei Medrea, Software Version V4.0
Copyrights (c) 2008 - YO2LIO, All Rights Reserved