µC/LIB™
Embedded Library

Release Notes
V1.35.00

Micriµm

For the Way Engineers Work

Revision History

Version	Date	Description
V1.35.00	2011 Jun	New features and improvements
V1.34	2010 Dec	Improvements
V1.33	2010 Oct	Bug fixes and improvements
V1.32	2010 Apr	New ports and improvements
V1.31	2009 Dec	New features, bug fixes, and improvements
V1.30	2009 Jun	New features, bug fixes, and improvements
V1.29	2009 Apr	New features and improvements
V1.28	2009 Jul	New features and improvements
V1.27	2009 Jan	New features, bug fixes, and improvements
V1.26	2008 Nov	New features, bug fixes, and improvements
V1.25	2008 Jul	New features and improvements
V1.24	2007 May	Improvements
V1.23	2007 Mar	Bug fixes and improvements
V1.22	2006 Sep	Improvements
V1.21	2006 Aug	New features and improvements
V1.20	2006 Jun	New features and improvements
V1.19	2006 Apr	Improvements
V1.18	2005 Oct	Bug fixes and improvements First version with release history
V1.17	2005 Jul	Improvements
V1.16	2005 Jun	Improvements
V1.15	2005 May	Improvements
V1.14	2005 Apr	Improvements
V1.13	2005 Feb	Improvements
V1.12	2004 Dec	Improvements
V1.11	2004 Nov	Improvements
V1.10	2004 Sep	Improvements
V1.00	2004 Feb	First release

900-uC-LIB-003

Required Modules

Version 1.35.00

µC/CPU version 1.27

Version 1.34

µC/CPU version 1.27

Version 1.33

µC/CPU version 1.27

Version 1.32

µC/CPU version 1.22

Version 1.31

µC/CPU version 1.22

Version 1.30

µC/CPU version 1.22

Version 1.29

µC/CPU version 1.22

Version 1.28

µC/CPU version 1.22

Version 1.27

µC/CPU version 1.20

Version 1.26

µC/CPU version 1.19

Version 1.25

µC/CPU version 1.18

Version 1.24

µC/CPU version 1.17

Version 1.23

µC/CPU version 1.16

Version 1.22

µC/CPU version 1.15

Version 1.21

µC/CPU version 1.14

Version 1.20

µC/CPU version 1.14

Version 1.19

µC/CPU version 1.14

Version 1.18

µC/CPU version 1.13

Version 1.17

µC/CPU version 1.12

Version 1.16

µC/CPU version 1.12

New Features

Version 1.35.00

V1.35.00-001
Added `DEF_NULL` to assign or validate `NULL` pointer values.

V1.35.00-002
Added new octet defines:

`DEF_OCTET_TO_BIT_NBR_BITS` `DEF_OCTET_TO_BIT_SHIFT`	number of bits to encode/decode octets-to-bits
`DEF_OCTET_TO_BIT_MASK`	mask value to encode/decode octets-to-bits

V1.35.00-003
Added new bit macros:

`DEF_BITxx()`	create bit mask of specified bit size with specified bit set
`DEF_BIT_MASK_xx()`	shift a bit mask of specified bit size
`DEF_BIT_FIELD_xx()`	create and shift a contiguous bit field of specified bit size

V1.35.00-004
Added new memory data value macros:

`MEM_BIG_TO_LITTLE_??()`	Convert big- endian data values to little-endian data values
`MEM_LITTLE_TO_BIG_??()`	Convert little-endian data values to big- endian data values
`MEM_???_TO_HOST_??()`	Convert big-/little-endian data values to host-endian data values
`MEM_HOST_TO_???_??()`	Convert host-endian data values to big-/little-endian data values
`MEM_VAL_COPY_GET_INTU_???()`	copy and decode data values from any memory address to any other memory address for any sized data values
`MEM_VAL_COPY_SET_INTU_???()`	copy and encode data values from any memory address to any other memory address for any sized data values
`MEM_VAL_COPY()`	copy data values from any memory address to any other memory address for any sized data values

See also 'New Features V1.21-001.

V1.35.00-005
Added new value validation macros:

`DEF_CHK_VAL_MIN()`	validates a value as greater than or equal to a specified minimum value
`DEF_CHK_VAL_MAX()`	validates a value as less than or equal to a specified maximum value
`DEF_CHK_VAL()`	validates a value as greater than or equal to a specified minimum value and less than or equal to a specified maximum value

Version 1.34

N/A

Version 1.33

V1.33-001
Added `LIB_STR_CFG_FP_MAX_NBR_DIG_SIG` to (optionally) configure the maximum number of floating-point number significant digits to format/parse. See also 'Improvements V1.33-003'.

Version 1.32

N/A

Version 1.31

V1.31-001
Added new Boolean-related defines:

```
DEF_INVALID
DEF_VALID
```

V1.31-002
Added new string functions:

`Str_Char_Last_N()`	searches a string for a character starting from the end of the string limited to a maximum number of characters
`Str_Str_N()`	searches a string for a sub-string limited to a maximum number of characters

See also 'New Features V1.20-001, V1.26-003, & V1.30-004'.

Version 1.30

V1.30-001
Added new template configuration file `lib_cfg.h`.

V1.30-002
Added `LIB_MEM_CFG_OPTIMIZE_ASM_EN` to enable/disable assembly-optimized memory functions.
See also 'Changes V1.30-001'.

V1.30-003
Added new math module functions:

`Math_Init()`	initializes mathematical library
`Math_Rand()`	generates a (pseudo-) random number
`Math_RandSeed()`	generates the next (pseudo-) random number after a specified seed value
`Math_RandSetSeed()`	sets the next (pseudo-) random number seed value

V1.30-004
Added new string functions:

`Str_Len_N()`	calculates a string's length limited to a maximum number of characters

See also 'New Features V1.20-001, V1.26-003, & V1.31-001'.

Version 1.29

V1.29-001
Added new time-related defines:

```
DEF_TIME_NBR_DAY_PER_WK
DEF_TIME_NBR_DAY_PER_YR
DEF_TIME_NBR_DAY_PER_YR_LEAP

DEF_TIME_NBR_HR_PER_WK
DEF_TIME_NBR_HR_PER_YR
DEF_TIME_NBR_HR_PER_YR_LEAP

DEF_TIME_NBR_MIN_PER_WK
DEF_TIME_NBR_MIN_PER_YR
DEF_TIME_NBR_MIN_PER_YR_LEAP

DEF_TIME_NBR_SEC_PER_WK
DEF_TIME_NBR_SEC_PER_YR
DEF_TIME_NBR_SEC_PER_YR_LEAP
```

Version 1.28

V1.28-001
Added `LIB_MEM_CFG_HEAP_BASE_ADDR` to (optionally) specify the heap memory base address.

Version 1.27

V1.27-001
Added new memory allocation function:

`Mem_PoolClr()`   clear a memory pool

See also 'Changes V1.26-001' & 'New Features V1.25-001'.

Version 1.26

V1.26-001
Added new memory allocation function:

`Mem_HeapAlloc()`   get memory from the heap

See also 'Changes V1.26-001' & 'New Features V1.25-001'.

V1.26-002
Added new ASCII module functions and macros:

`ASCII_IsDigOct()`   indicates whether a character is an octal digit
`ASCII_IS_DIG_OCT()`

See also 'New Features V1.25-002'.

V1.26-003
Added new string compare functions:

`Str_CmpIgnoreCase()`   compares two strings, ignoring case
`Str_CmpIgnoreCase_N()`   compares two strings, ignoring case, up to a maximum number
                                           of characters

See also 'New Features V1.20-001, V1.30-004, & V1.31-001'.

V1.26-004a
Added new string format functions:

`Str_FmtNbr_Int32U()`   formats an unsigned number into a string
`Str_FmtNbr_Int32S()`   formats a   signed number into a string

V1.26-004b
Added new string parse functions:

```
Str_ParseNbr_Int32U()                    parses an unsigned number from a string
Str_ParseNbr_Int32S()                    parses a       signed number from a string
```

Version 1.25

V1.25-001
Added new memory allocation functions:

```
Mem_PoolCreate()                         create a memory pool
Mem_PoolBlkGet()                         get  a memory block from     a memory pool
Mem_PoolBlkFree()                        free a memory block back to a memory pool
```

See also 'Changes V1.26-001'.

V1.25-002
Added new ASCII module functions and macros:

```
ASCII_IsAlpha()                          indicates whether a character is alphabetic
ASCII_IS_ALPHA()


ASCII_IsAlnum()                          indicates whether a character is alphanumeric
ASCII_IS_ALNUM()                         (see also 'Changes V1.27-001')


ASCII_IsLower()                          indicates whether a character is lowercase
ASCII_IS_LOWER()


ASCII_IsUpper()                          indicates whether a character is uppercase
ASCII_IS_UPPER()


ASCII_IsDig()                            indicates whether a character is a decimal digit
ASCII_IS_DIG()


ASCII_IsDigHex()                         indicates whether a character is a hexadecimal digit
ASCII_IS_DIG_HEX()


ASCII_IsBlank()                          indicates whether a character is blank
ASCII_IS_BLANK()


ASCII_IsSpace()                          indicates whether a character is a space
ASCII_IS_SPACE()
```

`ASCII_IsPrint()` `ASCII_IS_PRINT()`	indicates whether a character is printable
`ASCII_IsGraph()` `ASCII_IS_GRAPH()`	indicates whether a character is graphic
`ASCII_IsPunct()` `ASCII_IS_PUNCT()`	indicates whether a character is punctuation
`ASCII_IsCtrl()` `ASCII_IS_CTRL()`	indicates whether a character is a control
`ASCII_ToLower()` `ASCII_TO_LOWER()`	converts uppercase to lowercase
`ASCII_ToUpper()` `ASCII_TO_UPPER()`	converts lowercase to uppercase
`ASCII_Cmp()`	compares two characters (case insensitive)

See also 'Changes V1.25-001'.

Version 1.24

V1.24-001
Added new CPU-related integer defines:

```
DEF_INT_CPU_NBR_BITS
DEF_INT_CPU_MASK
DEF_INT_CPU_U_MIN_VAL
DEF_INT_CPU_U_MAX_VAL
DEF_INT_CPU_S_MIN_VAL
DEF_INT_CPU_S_MAX_VAL
DEF_INT_CPU_S_MIN_VAL_ONES_CPL
DEF_INT_CPU_S_MAX_VAL_ONES_CPL
```

Version 1.23

N/A

Version 1.22

N/A

Version 1.21

V1.21-001
Added new memory data value macros:

`MEM_VAL_GET_???()`	decode data values from any memory address
`MEM_VAL_SET_???()`	encode data values to any memory address
`MEM_VAL_COPY_GET_???()`	copy and decode data values from any memory address to any other memory address
`MEM_VAL_COPY_SET_???()`	copy and encode data values from any memory address to any other memory address
`MEM_VAL_COPY_???()`	copy data values from any memory address to any other memory address

See also 'New Features V1.35.00-004'.

Version 1.20

V1.20-001
Added new string functions:

`Str_Copy_N()`	copies a string limited to a maximum number of characters
`Str_Cat_N()`	concatenates two strings limited to a maximum number of characters
`Str_Char_N()`	searches a string for a character limited to a maximum number of characters

See also 'New Features V1.26-003, V1.30-004, & V1.31-001'.

Version 1.19

N/A

Version 1.18

N/A

Improvements

Version 1.35.00

V1.35.00-001
Updated µC/LIB's CERT-C and MISRA-C compliance:

V1.35.00-001a1
Added 'u' qualifier back to certain unsigned integer constants. This reverts the removal of all unsigned integer constants, requiring instead that unsigned constants used in signed expressions must be cast to appropriate signed data types. See also 'Improvements V1.34-001b & V1.31-001a1'.

V1.35.00-001a2
Removed 'L' qualifier from certain long integer constants. This reverts the return of certain long integer constants. See also 'Improvements V1.33-001b & V1.31-001a2'.

V1.35.00-001b
Modified `DEF_BIT_IS_CLR()` and `DEF_BIT_IS_SET_ANY()` to explicitly test masked values for zero. See also 'Changes V1.35.00-001b'.

V1.35.00-002
Modified `MEM_VAL_SET_xxx()` to cast bit mask to appropriate integer data type size.

V1.35.00-003
Refactored `Mem_PoolBlkFree()` to validate memory block address before validating if the memory pool is full.

V1.35.00-004
Modified the following functions to invalidate `len_max` for non-positive values:

```
Str_Cat_N()
Str_Cmp_N()
Str_CmpIgnoreCase_N()
Str_Char_N()
Str_Char_Last_N()
Str_Str_N()
```

V1.35.00-005
Modified `Str_FmtNbr_Int32()` to consistently compare decimal digit values for less than 10 versus less than or equal to 9.

Version 1.34

V1.34-001
Updated µC/LIB's CERT-C and MISRA-C compliance:

V1.34-001a
Removed the following standard library headers from being #include'd in `lib_str.h`:

```
<ctype.h>
<errno.h>
<limits.h>
<stdio.h>
<stdlib.h>
```

V1.34-001b
Removed 'u' qualifier from certain integer constants. This reverts a previously implemented improvement only for certain integer constants that may be used in both signed and unsigned expressions. See also 'Improvements V1.31-001a1'.

V1.34-001c
Added `const` modifier to all appropriate API function pointer arguments. See also 'Changes V1.34-001'.

V1.34-002
Modified the following functions to reconfigure any optional `NULL` return pointers to point to an unused local variable in order to remove `NULL` pointers from scope:

```
Mem_HeapAlloc()
Mem_PoolCreate()
Str_ParseNbr_Int32()
```

Version 1.33

V1.33-001
Updated µC/LIB's CERT-C and MISRA-C compliance:

V1.33-001a
Modified functions to trap `NULL` 'p_err' pointers with µC/CPU's new `CPU_SW_EXCEPTION()` macro.

V1.33-001b
Added 'L' qualifier to certain long integer constants. This reverts a previously incorrect assumption about certain integer data type and constant promotions. See also 'Improvements V1.31-001a2'.

V1.33-001c
Removed `Str_IsPrint()` and `Str_ToLong()` standard library string macros.

V1.33-002
Modified `Str_Copy_N()` to allow copies of 0 size. See also 'Changes V1.33-002'.

V1.33-003

Modified `Str_FmtNbr_32()` to limit the maximum number of floating-point number significant digits to format. See also 'New Features V1.33-001'.

V1.33-004

Modified `Str_FmtNbr_32()` and `Str_FmtNbr_Int32()` to always prepend possible negative sign immediately prior to the formatted number's (`nbr`) most significant digit if lead character (`lead_char`) is not an alphanumerical digit; otherwise, prepends possible negative sign prior to any alphanumerical lead characters.

V1.33-005

Improved the following functions to check for heap or segment memory request overflows:

```
Mem_HeapAlloc()
Mem_PoolCreate()
Mem_PoolSegCalcTotSize()
Mem_PoolSegAlloc()
```

V1.33-006

Added 64-bit integer `#define`'s in `lib_def.h` .

Version 1.32

V1.32-001

Updated µC/LIB's CERT-C and MISRA-C compliance:

V1.32-001a

Encapsulated all macros defined as code blocks within `do..while(0)` conditions.

V1.32-002

Removed `Mem_PoolSegAlloc()`'s critical sections since `Mem_PoolSegAlloc()` is always called with critical sections already acquired.

Version 1.31

V1.31-001

Updated µC/LIB's CERT-C and MISRA-C compliance:

V1.31-001a1

Appended unsigned 'u' qualifier to all unsigned integer constants.

V1.31-001a2

Removed redundant 'L' qualifier from all long integer constants.

V1.31-001b

Replaced all instances of '???' comments with '&&&' (to avoid possible usage of C trigraphs).

V1.31-001c

Refactored the following functions to copy any function arguments into local variables before modifying:

```
Mem_HeapAlloc()
Mem_PoolCreate()

Str_Len_N()
Str_Copy_N()
Str_Cat_N()
Str_Cmp_N()
Str_CmpIgnoreCase_N()
Str_Char_N()
```

V1.31-002

Improved the following string functions to call their corresponding length-limited functions:

```
Str_Char_Last()    calls Str_Char_Last_N()
Str_Str()          calls Str_Str_N()
```

See also 'New Features V1.31-002' & 'Improvements V1.26-001'.

V1.31-003

Improved the following functions to terminate, and return errors when possible, if any strings point or overlap with the NULL address (i.e. the terminating NULL character is not found prior to the string pointer overflowing to the NULL address):

```
Str_Copy_N()
Str_Cat_N()
Str_Char_N()
Str_Char_Last_N()
Str_Str_N()
```

See also 'Corrections V1.31-001'.

Version 1.30

V1.30-001
Improved the following bit macros to be called from within conditional expressions:

```
DEF_BIT_SET()
DEF_BIT_CLR()
```


Version 1.29

V1.29-001
Improved the configuration of optional memory allocation argument checking.

Version 1.28

V1.28-001
Replaced all 'cpu_sr' local variable declarations with µC/CPU's new `CPU_SR_ALLOC()` macro.

Version 1.27

N/A

Version 1.26

V1.26-001
Improved the following string functions to call their corresponding length-limited functions:

```
Str_Copy()    calls Str_Copy_N()
Str_Cat()     calls Str_Cat_N()
Str_Cmp()     calls Str_Cmp_N()
Str_Char()    calls Str_Char_N()
```

See also 'New Features V1.20-001' & 'Improvements V1.31-002'.

V1.26-002a
Improved unsigned integer macro definitions by explicitly declaring unsigned constant.

V1.26-002b
Improved signed integer macro definitions by avoiding twos-complement arithmetic underflow.

Version 1.25

N/A

Version 1.24

V1.24-001
Added `LIB_VERSION` to indicate current library module software version number.

V1.24-002
Improved several `DEF_BIT_???()` macros to handle overflow boundary conditions.

V1.24-003
Added several `LIB_STR_???` common string defines.

Version 1.23

V1.23-001
Removed `malloc()` and all other references to standard library memory functions.

Version 1.22

N/A

Version 1.21

N/A

Version 1.20

V1.20-001
Improved ARM assembly port files to be compatible for both ARM and Thumb modes.

Version 1.19

N/A

Version 1.18

V1.18-001
Added macro function headers for all `lib_def.h` macros.

V1.18-002
Improved consistency for all `lib_str.c` functions.

Changes

Version 1.35.00

V1.35.00-001a
Modified `DEF_BIT_IS_SET()` and `DEF_BIT_IS_CLR()` to return `DEF_NO` for `NULL` masks (i.e., masks of value 0).

V1.35.00-001b
Modified `DEF_BIT_IS_CLR()`, `DEF_BIT_IS_SET_ANY()`, and `DEF_BIT_IS_CLR_ANY()` to test masked values with equality (instead of inequality) to zero or specified mask. See also 'Improvements V1.35.00-001b'.

Version 1.34

V1.34-001
Modified the following functions to add the `const` modifier to all appropriate pointer arguments:

```
Mem_Copy()
Mem_Cmp()
```

```
Str_Len()
Str_Len_N()
Str_Copy()
Str_Copy_N()
Str_Cat()
Str_Cat_N()
Str_Cmp()
Str_Cmp_N()
Str_CmpIgnoreCase()
Str_CmpIgnoreCase_N()
Str_Char()
Str_Char_N()
Str_Char_Last()
Str_Char_Last_N()
Str_Str()
Str_Str_N()
```

```
Str_ParseNbr_Int32U()
Str_ParseNbr_Int32S()
```

V1.34-002
Modified `Mem_HeapAlloc()` and `Mem_PoolCreate()` to invalidate 0 (zero) as a valid value for arguments `align` and `blk_align`, respectively, which defaults to no alignment. Only a positive number of octets that specify the word boundary alignment are validated.

V1.34-003a

Modified the following functions to format an invalid string for any invalid arguments, error conditions, or if the number to format (`nbr`) has more significant integer digits than the number of digits to format (`nbr_dig`):

```
Str_FmtNbr_Int32U()
Str_FmtNbr_Int32S()
Str_FmtNbr_32()
```

The invalid string is formatted with `nbr_dig` and `nbr_dp` number of question marks (`'?'`).

V1.34-003b

Whenever an invalid string is formatted for any reason, string format functions also return a `NULL` pointer.

V1.34-004

Modified the following functions to invalidate any lead character (`lead_char`) that is a valid number digit with the exception of zero (`'0'`):

```
Str_FmtNbr_Int32U()
Str_FmtNbr_Int32S()
Str_FmtNbr_32()
```

Version 1.33

V1.33-001

Modified `Mem_PoolBlkGet()` to invalidate memory requests of 0 size.

V1.33-002

Modified `Str_Copy_N()` to allow copies of 0 size. See also 'Improvements V1.33-002'.

Version 1.32

N/A

Version 1.31

N/A

Version 1.30

V1.30-001

Replaced assembly-optimized configuration from generic `uC_CFG_OPTIMIZE_ASM_EN` to library-specific `LIB_MEM_CFG_OPTIMIZE_ASM_EN`. See also 'New Features V1.30-002'.

Version 1.29

N/A

Version 1.28

N/A

Version 1.27

V1.27-001
Renamed the following `lib_ascii.h` macros and functions:

```
ASCII_IsAlnum()      renamed to ASCII_IsAlphaNum()
ASCII_IS_ALNUM()     renamed to ASCII_IS_ALPHA_NUM()
```

V1.27-002
Modified `Str_FmtNbr_???()` leading character parameter from a Boolean ('`lead_zeros`') that specified whether leading zeros were prepended to the formatted number string when necessary, to the desired ASCII character ('`lead_char`') to prepend to the formatted number string:

```
CPU_CHAR  *Str_FmtNbr_Int32U(CPU_INT32U    nbr,
                             CPU_INT08U    nbr_dig,
                             CPU_INT08U    nbr_base,
                             CPU_CHAR      lead_char,
                             CPU_BOOLEAN   lower_case,
                             CPU_BOOLEAN   nul,
                             CPU_CHAR     *pstr);


CPU_CHAR  *Str_FmtNbr_Int32S(CPU_INT32S    nbr,
                             CPU_INT08U    nbr_dig,
                             CPU_INT08U    nbr_base,
                             CPU_CHAR      lead_char,
                             CPU_BOOLEAN   lower_case,
                             CPU_BOOLEAN   nul,
                             CPU_CHAR     *pstr);


CPU_CHAR  *Str_FmtNbr_32    (CPU_FP32      nbr,
                             CPU_INT08U    nbr_dig,
                             CPU_INT08U    nbr_dp,
                             CPU_CHAR      lead_char,
                             CPU_BOOLEAN   nul,
                             CPU_CHAR     *pstr);
```

Version 1.26

V1.26-001
Changed memory pool configuration to memory allocation configuration — `LIB_MEM_CFG_POOL_EN` to `LIB_MEM_CFG_ALLOC_EN`.

V1.26-002
Changed the following `lib_mem.h` error codes:

`LIB_MEM_ERR_INVALID_ADDR` changed to `LIB_MEM_ERR_INVALID_BLK_ADDR`

V1.26-003
Changed the following `lib_def.h` macro constants:

```
DEF_INACTIVE    redefined to 0
DEF_ACTIVE      redefined to 1
```

Version 1.25

V1.25-001
The following macros in `lib_str.h` have been deprecated and replaced with new macros and functions in `lib_ascii.h`:

```
Str_IsAlpha()    replaced with ASCII_IsAlpha() / _IS_ALPHA()
Str_IsDigit()    replaced with ASCII_IsDig()   / _IS_DIG()
Str_IsSpace()    replaced with ASCII_IsSpace() / _IS_SPACE()
Str_IsPrint()    replaced with ASCII_IsPrint() / _IS_PRINT()
Str_IsUpper()    replaced with ASCII_IsUpper() / _IS_UPPER()
Str_IsLower()    replaced with ASCII_IsLower() / _IS_LOWER()

Str_ToUpper()    replaced with ASCII_ToUpper() / _TO_UPPER()
Str_ToLower()    replaced with ASCII_ToLower() / _TO_LOWER()
```

See also 'New Features V1.25-002'.

Version 1.24

N/A

Version 1.23

N/A

Version 1.22

N/A

Version 1.21

N/A

Version 1.20

V1.20-001
The following macro names in `lib_str.h` have been changed to comply with standard naming conventions:

```
Is_Alpha()      changed to Str_IsAlpha()
Is_Digit()      changed to Str_IsDigit()
Is_Space()      changed to Str_IsSpace()
Is_Print()      changed to Str_IsPrint()
Is_Upper()      changed to Str_IsUpper()
Is_Lower()      changed to Str_IsLower()

To_Upper()      changed to Str_ToUpper()
To_Lower()      changed to Str_ToLower()

Str_To_Long()        changed to Str_ToLong()
Str_Format_Print()   changed to Str_FmtPrint()
Str_Format_Scan()    changed to Str_FmtScan()
```

Version 1.19

V1.19-001
Macros `Str_Format_Print()` and `Str_Format_Scan()` in `lib_str.h` have been corrected to be compatible with some compilers.

Version 1.18

V1.18-001
`DEF_BIT_MASK()` macro and `DEF_BIT_FIELD()` macro switched names.

V1.18-002
Renamed `Str_Char_R()` to `Str_Char_Last()`.

Corrections

Version 1.35.00

N/A

Version 1.34

N/A

Version 1.33

V1.33-001

`Str_Char_N()` incorrectly returned a pointer to the search character even if its first occurrence was (`len_max + 1`) characters into the search string. Fixed by always returning a pointer to `NULL` string if the search character is not found in the search string within the first '`len_max`' characters.

Version 1.32

N/A

Version 1.31

V1.31-001

Refactored the following functions to fully comply with their standard library equivalents (see also 'Improvements V1.31-003'):

V1.31-001a

`Str_Copy_N()` incorrectly always appended a terminating `NULL` character to the destination string, regardless of the specified maximum number of characters to copy. Fixed by only copying the source string's terminating `NULL` character if available within the specified maximum number of characters to copy.

V1.31-001b

`Str_Str_N()` incorrectly returned a pointer to the string's terminating `NULL` character if the search string was a zero-length `NULL` string. Fixed by returning a pointer to the string if the search string is a zero-length `NULL` string.

Version 1.30

N/A

Version 1.29

N/A

Version 1.28

N/A

Version 1.27

V1.27-001

`Str_ParseNbr_Int32()` failed to always set negative sign ('neg') during validation. Fixed by always setting 'neg' for all conditions.

Version 1.26

V1.26-001

`Mem_PoolCreate()` incorrectly calculated the number of additional octets required to successfully allocate all requested memory (returned by '`p_octets_reqd`') for certain fault conditions. Fixed by calculating and returning the actual additional octets required to successfully allocate all requested memory for all error/fault conditions.

Version 1.25

N/A

Version 1.24

N/A

Version 1.23

V1.23-001

ARM assembly port files were not completely compatible for both ARM and Thumb modes (see 'Improvements V1.20-001'). Corrected by using only ARM and Thumb mode instructions.

Version 1.22

N/A

Version 1.21

N/A

Version 1.20

N/A

Version 1.19

N/A

Version 1.18

V1.18-001

`Str_Str()` incorrectly assigned unsigned string lengths to signed variables. Corrected by assigning string lengths to unsigned variables.

V1.18-002

`lib_mem_a.asm` did not correctly terminate the memory copy during the `Pre_Copy_1` label if no more data octets to copy. Corrected by terminating the memory copy if no more data octets.

Known Problems

Version 1.35.00

Version 1.34

Version 1.33

Version 1.32

Version 1.31

Version 1.30

Version 1.29

Version 1.28

Version 1.27

Version 1.26

Version 1.25

Version 1.24

Version 1.23

V1.18-001b (Unresolved)

Version 1.22

Version 1.21

Version 1.20

Version 1.19

V1.18-001a (Unresolved)
V1.18-001b (Unresolved)

Version 1.18

V1.18-001a
`lib_mem.h` includes some standard library files and functions. All references to standard library files and functions should be removed once all custom library functions are implemented.

V1.18-001b
`lib_str.h` includes some standard library files and functions. All references to standard library files and functions should be removed once all custom library functions are implemented.

Limitations

001

Does not support variable argument library functions

Contacts

Micriµm
1290 Weston Road, Suite 306
Weston, FL 33326
USA

Phone: +1 954 217 2036
Fax: +1 954 217 2037
E-mail: Licensing@Micrium.com
Web: www.Micrium.com