SPRINT LIBRARY

The *mikroC PRO for PIC* provides the standard ANSI C Sprintf function for easy data formatting.

Note: In addition to ANSI C standard, the Sprint Library also includes two limited versions of the `sprintf` function (`sprinti` and `sprintl`). These functions take less ROM and RAM and may be more convenient for use in some cases.

Functions

- sprintf
- sprintl
- sprinti

sprintf

Prototype	`sprintf(char *wh, const char *f,...);`
Returns	The function returns the number of characters actually written to destination string.
Description	`sprintf` is used to format data and print them into destination string. Parameters: - `wh:` destination string - `f`: format string The `f` argument is a format string and may be composed of characters, escape sequences, and format specifications. Ordinary characters and escape sequences are copied to the destination string in the order in which they are interpreted. Format specifications always begin with a percent sign (`%`) and require additional arguments to be included in the function call. The format string is read from left to right. The first format specification encountered refers to the first argument after `f` and then converts and outputs it using the format specification. The second format specification accesses the second argument after `f`, and so on. If there are more arguments than format specifications, then these extra arguments are ignored. Results are unpredictable if there are not enough arguments for the format specifications. The format specifications have the following format: `% [ flags] [ width] [ .precision] [{ l | L }] conversion_type`

Description	Each field in the format specification can be a single character or a number which specifies a particular format option. The `conversion_type` field is where a single character specifies that the argument is interpreted as a character, string, number, or pointer, as shown in the following table:

`conversion_type`	**Argument Type**	**Output Format**
d	`int`	Signed decimal number
u	`unsigned int`	Unsigned decimal number
o	`unsigned int`	Unsigned octal number
x	`unsigned int`	Unsigned hexadecimal number using 0123456789abcdef
X	`unsigned int`	Unsigned hexadecimal number using 0123456789ABCEDF
f	`double`	Floating-point number using the format [-]dddd.dddd
e	`double`	Floating-point number using the format [-]d.ddde[-]dd
E	`double`	Floating-point number using the format [-]d.ddddE[-]dd
g	`double`	Floating-point number using either e or f format, whichever is more compact for the specified value and precision
c	`int`	int is converted to an `unsigned char`, and the resulting character is written
s	`char *`	String with a terminating null character
p	`void *`	Pointer value, the X format is used
%	`<none>`	A % is written. No argument is converted. The complete conversion specification shall be %%.

Description	The `flags` field is where a single character is used to justify the output and to print +/- signs and blanks, decimal points, and octal and hexadecimal prefixes, as shown in the following table.

`flags`	**Meaning**
`+`	Left justify the output in the specified field width.
`-`	Prefix the output value with + or - sign if the output is a signed type.
`space (' ')`	Prefix the output value with a blank if it is a signed positive value. Otherwise, no blank is prefixed.
`#`	Prefix a non-zero output value with `0`, `0x`, or `0X` when used with `o`, `x`, and `X` field types, respectively. When used with the `e`, `E`, `f`, `g`, and `G` field types, the `#` flag forces the output value to include a decimal point. In any other case the `#` flag is ignored.
`*`	Ignore format specifier.

The `width` field is a non-negative number that specifies a minimum number of printed characters. If a number of characters in the output value is less than width, blanks are added on the left or right (when the - flag is specified) in order to pad to the minimum width. If the width is prefixed with 0, then zeros are padded instead of blanks. The `width` field never truncates a field. If the length of the output value exceeds the specified width, all characters are output.

The `precision` field is a non-negative number that specifies the number of characters to print, number of significant digits, or number of decimal places. The precision field can cause truncation or rounding of the output value in the case of a floating-point number as specified in the following table.

`flags`	**MeaningMeaning of the `precision` field**
`d, u, o, x, X`	The precision field is where you specify the minimum number of digits that will be included in the output value. Digits are not truncated if the number of digits in an argument exceeds that defined in the precision field. If the number of digits in the argument is less than the precision field, the output value is padded on the left with zeros.
`f`	The precision field is where you specify the number of digits to the right of the decimal point. The last digit is rounded.
`e, E`	The precision field is where you specify the number of digits to the right of the decimal point. The last digit is rounded.
`g`	The precision field is where you specify the maximum number of significant digits in the output value.
`c, C`	The precision field has no effect on these field types.
`s`	The precision field is where you specify the maximum number of characters in the output value. Excess characters are not output.

Description	The optional characters `l` or `L` may immediately precede `conversion_type` to respectively specify long versions of the integer types `d, i, u, o, x,` and `X`. You must ensure that the argument type matches that of the format specification. You can use type casts to ensure that the proper type is passed to `sprintf`.

sprintl

Prototype	`sprintl(char *wh, const char *f,...);`
Returns	The function returns the number of characters actually written to destination string.
Description	The same as sprintf, except it doesn't support float-type numbers.

sprinti

Prototype	`sprinti(char *wh, const char *f,...);`
Returns	The function returns the number of characters actually written to destination string.
Description	The same as sprintf, except it doesn't support long integers and float-type numbers.