Qbasic String Functions

One of QBasic's strengths is in the manipulation of strings, offering over 20 different built-in string functions.

• **Case-Conversion**	`ucase$, lcase$`
• **String Variable Properties..**	`val len`
• **Single Character Function**	`asc, chr$`
• **Truncate Strings**	`left$, right$`
• **Remove Spaces**	`ltrim$, rtrim$`
• **Find/Replace**	`mid$, instr`
• **Justification**	`lset, rset`
• **Return Repetitive Strings**	`space$, string$`
• **Convert Numbers to String**	`cvi, cvl, cvs, cvd`
• **Convert Numbers to String**	`str$, mkd$, mki$, mks$, mkl$`
• **Assignment**	`let, swap, clear`

String Function Reference
Here's a quick reference of the available string functions, in alphabetical order. The functions are very simple to use.

- **asc** - returns ASCII code for 1st character in a string
- `    result$ = asc("hello")      # returns ASCII of "h", which is 104`

- **chr$** - returns characters corresponding to specified ASCII code
- `    result$ = chr$ (104)        # returns "h"`

- **clear** - closes all files, clears all common variables, reset values of all strings/numbers.
- `    clear                     # no arguments required`

- **cvi** - convert integer to string – returns an integer
- `    result$ = cvi(var%)      # result$ = string value of var$`
- `    result$ = cvi(24)        # result$ = "24"`

- **cvl** - convert long integer to string – returns a long integer
- `result$ = cvl(var&)      # result$ = string value of var&`
- `result$ = cvl(71224)        # result$ = "71224"`

- **cvs** - convert single to string – returns single-precision number
- `result$ = cvs(var!)      # result$ = string value of var!`
- `result$ = cvs(1.224)        # result$ = "1.224"`

- **cvd** - convert double to string – returns a double-precision number
- `result$ = cvd(var#)      # result$ = string value of var#`
- `result$ = cvd(24.78882)        # result$ = "24.78882"`

- **instr** - returns position of one string within another string
- `result = instr (StartPos%, StringToSearch$, StringToFind$)`
- `result = instr (1, "abcdefg", "de")  # returns 4`

Note: First character is position 1. StartPos% is optional (default is position 1).

- **lcase$** - converts string to all lowercase letters
- `result$ = lcase$ ("HELP Me")      # returns "help me"`

- **left$** - returns specified leftmost number of characters
- `result$ = left$("Hello",2)        # returns "He"`

- **len** - desc
- `result = len ("hello")      # returns 5`
- `x$ = "dog"`
- `result = len(x$)           # returns 3`

- **let** - assigns value to a variable
- `let result = 5`

- **lset** - left justifies a smaller string within a larger string of spaces
- `buffer$ = "          "        # ten spaces`
- `lset buffer$ = "dog"          # buffer$ contains "dog       "`

Note: The original content of buffer$ is replaced with spaces before placing the smaller string.

- **ltrim$** - removes leading spaces from a string
- `result$ = ltrim$("  mydog")        # returns "mydog"`

- **mid$** - returns or replaces part of a string
- `    result$ = mid$(StringToSearch$, StartPos%, Length%)`
- `    result$ = mid$("abc123def", 1, 3)       # returns "abc"`
- `    result$ = mid$("abc123def", 3, 3)       # returns "c12"`
-
- `    mid$(EditString$, StartPos%, Length%) = ReplacementString$`
- `    var$ = "my dog is nice"`
- `    mid$(var$, 4, 3) = "cat"      # var$ = "my cat is nice"`

Length is optional in both cases. If omitted, mid$ returns or replaces all characters to the right of the start position.

- **mkd$** - convert double to a string – returns a 8-byte string
- `    result$ = mkd$(var#)        # result is string version of var#`
- `    result$ = mkd$(1.234)       # result is "1.234"`

- **mki$** - convert integer to a string – returns a 2-byte string
- `    result$ = mki$(var%)        # result is string version of var%`
- `    result$ = mki$(12)          # result is "12"`

- **mkl$** - convert long integer to a string – returns a 4-byte string
- `    result$ = mkl$(var&)        # result is string version of var&`
- `    result$ = mkl$&(7423234)    # result is "7423234"`

- **mks$** - convert single to a string – returns a 4-byte string
- `    result$ = mks$(var!)        # result is string version of var!`
- `    result$ = mks$(1.234)       # result is "1.234"`

- **right$** - returns specified rightmost number of characters
- `    result$ = right$("Hello",2)        # returns "lo"`

- **rset** - right justifies a smaller string within a larger string of spaces
- `    buffer$ = "          "     # ten spaces`
- `    rset buffer$ = "dog"       # buffer$ contains "       dog"`

Note: The original content of buffer$ is replaced with spaces before placing the smaller string.

- **rtrim$** - removes trailing spaces from a string
- `    result$ = rtrim$("mycat  ")        # returns "mycat"`

- **space$** - returns a string of spaces of a specified length
- `    result$ = space$(5)       # returns "     ", which is 5 spaces`

- **str$** - returns a string representation of a number
- `    result$ = str$(12)           # returns "12"`
- `    result$ = str$(1.342)        # returns "1.342)`

- **string$** - returns string of a repetitive character
- `    result$ = string$(Repititions%, CharacterToRepeat$)`
- `    result$ = string$(5,"a")    # returns "aaaaa"`
- `    result$ = string$(5,66)     # returns "BBBBB" - ASCII 66 is "B"`

  Note: If CharacterToRepeat is more than one character, only the first character is repeated.

- **swap** - exchange values of two variables
- `    a$ = "a" : b$ = "b"   # set test values`
- `    swap a$, b$           # a$ = "b" and b$ = "a" (values swapped)`

- **ucase$** - converts string to all uppercase letters
- `    result$ = lcase$ ("Help Me")      # returns "HELP ME"`

- **val** - converts a string to a number
- `    result = val("12.3")          # returns 12.3`
- `    result = val("abc")           # returns 0`
- `    result = val("12ab")          # returns 12`

- QBasic uses the **+** operator to combine strings.
- `+      string concatenation        # "a" + "b" returns "ab"`