Global Functions

ON THIS PAGE

- ifGlobal
  - CreateObject(name As String) As Object
  - RestartScript() As Void
  - RestartApplication() As Void
  - Sleep(milliseconds As Integer)
  - asc(letter As String) As Integer
  - chr(character As Integer) As String
  - len(target_string As String) As Integer
  - str(value As Double) As String
  - strI(value As Integer) As String
  - val(target_string As String) As Double
  - abs(x As Double) As Double
  - atn(x As Double) As Double
  - csng(x As Integer) As Float
  - cdbl(x As Integer) As Double
  - cint(x As Double) As Integer
  - cos(x As Double) As Double
  - exp(x As Double) As Double
  - fix(x As Double) As Integer
  - int(x As Double) As Integer
  - log(x As Double) As Double
  - sgn(x As Double) As Integer
  - sgnI(x As Integer) As Integer
  - sin(x As Double) As Double
  - tan(x As Double) As Double
  - sqr(x As Double) As Double
  - Left(target_string As String, n As Integer) As String
  - Right(target_string As String, n As Integer) As String
  - StringI(n As Integer, character As Integer) As String
  - String(n As Integer, character As String) As String
  - Mid(target_string As String, start_position As Integer, length As Integer) As String
  - Instr(start_position As Integer, search_text As String, substring As String) As Integer
  - GetInterface(object As Object, ifname As String) As Interface
  - Wait(timeout As Integer, port As Object) As Object
  - ReadAsciiFile(file_path As String) As String
  - WriteAsciiFile(file_path As String, buffer As String) As Boolean
  - ListDir(path As String) As Object
  - MatchFiles(path As String, pattern_in As String) As Object
  - LCase(target_string As String) As String
  - UCase(target_string As String) As String
  - DeleteFile(file_path As String) As Boolean
  - DeleteDirectory(diretory As String) As Boolean
  - CreateDirectory(directory As String) As Boolean
  - RebootSystem() As Void
  - ShutdownSystem() As Void
  - UpTime(dummy As Integer) As Double
  - FormatDrive(drive As String, fs_type As String) As Boolean
  - EjectDrive(drive As String) As Boolean
  - CopyFile(source As String, destination As String) As Boolean
  - MoveFile(source As String, destination As String) As Boolean
  - MapFilenameToNative(path As String) As String
  - strtoi(target_string As String) As Integer
  - rnd(a As Dynamic) As Dynamic
  - RunGarbageCollector() As roAssociativeArray
  - GetDefaultDrive() As String
  - SetDefaultDrive(drive As String)
  - EnableZoneSupport(enable As Boolean)
  - EnableAudioMixer(enable As Boolean)
  - Pi() As Double
  - ParseJson(json_string As String) As Object
  - FormatJson(json As roAssociativeArray, flags As Integer) As String

OS8

- OS8
- Version 7.1
- Version 7.0
- Version 6.2
- Version 6.1
- Previous Versions

BrightScript provides a set of standard, module-scope functions that are stored in the global object. If a global function is referenced, the compiler directs the runtime to call the appropriate global object member. When calling a global function, you do not need to use the dot operator to reference the *roGlobal* object.

> **Note**
>
> Global trigonometric functions accept and return values in radians, not degrees.

ifGlobal

CreateObject(name As String) As Object

Creates a BrightScript object corresponding to the specified class name. This method returns invalid if object creation fails. Some objects have parameters in their constructor, which must be passed after the class `name` in a comma-separated list.

```
sw = CreateObject("roGpioControlPort")
serial = CreateObject("roSerialPort", 0, 9600)
```

RestartScript() As Void

Exits the current script. The system then scans for a valid *autorun.brs* file to run.

RestartApplication() As Void

Restarts the BrightSign application.

Sleep(milliseconds As Integer)

Instructs the script to pause for a specified amount of time without wasting CPU cycles. The sleep interval is specified in milliseconds.

asc(letter As String) As Integer

Returns the ASCII code for the first character of the specified string. A null-string argument will cause an error.

chr(character As Integer) As String

Returns a one-character string containing a character reflected by the specified ASCII or control. For example, because quotation marks are normally used as string delimiters, you can pass ASCII code 34 to this function to add quotes to a string.

len(target_string As String) As Integer

Returns the number of characters in a string.

str(value As Double) As String

Converts a specified float value to a string. This method also returns a string equal to the character representation of a value. For example, if A is assigned a value of 58.5, then calling `str(A)` will return "58.5" as a string.

strl(value As Integer) As String

Converts a specified integer value to a string. This method also returns a string equal to the character representation of a value. For example, if A is assigned a value of 58.5, then calling `stri(A)` will return "58" as a string.

val(target_string As String) As Double

Returns a number represented by the characters in the string argument. This is the opposite of the `str()` function. For example, if A is assigned the string "58", and B is assigned the string "5", then calling `val(A+"."+B)` will return the float value 58.5.

abs(x As Double) As Double

Returns the absolute value of the argument `x`.

atn(x As Double) As Double

Returns the arctangent (in radians) of the argument `x` (i.e. `Atn(x)` returns "the angle whose tangent is `x`"). To get the arctangent in degrees, multiply `Atn(x)` by 57.29578.

csng(x As Integer) As Float

Returns a single-precision float representation of the argument `x`.

cdbl(x As Integer) As Double

Returns a double-precision float representation of the argument `x`.

cint(x As Double) As Integer

Returns an integer representation of the argument `x` by rounding to the nearest whole number.

cos(x As Double) As Double

Returns the cosine of the arugment `x`. The argument must be in radians. To obtain the cosine of `x` when `x` is in degrees, use `Cos(x*.01745329)`.

exp(x As Double) As Double

Returns the natural exponential of `x`. This is the inverse of the `log()` function.

fix(x As Double) As Integer

Returns a truncated representation of the argument `x`. All digits to the right of the decimal point are removed so that the resultant value is an integer. For non-negative values of *x*, `fix(x)` is equal to `int(x)`. For negative values of `x`, `fix(x)` is equal to `int(x)+1`.

int(x As Double) As Integer

Returns an integer representation of the argument `x` using the largest whole number that is not greater than the argument. For example, `int(2.2)` returns 2, while `fix(-2.5)` returns -3.

log(x As Double) As Double

Returns the natural logarithm of the argument `i` (i.e. $log_e(x)$). This is the inverse of the `exp()` function. To find the logarithm of a number to a base *b*, use the following formula: $log_b(x) = log_e(x)/log_e(b)$.

sgn(x As Double) As Integer

Returns an integer representing how the float argument *x* is signed: -1 for negative, 0 for zero, and 1 for positive.

sgnl(x As Integer) As Integer

Returns an integer representing how the integer argument *x* is signed: -1 for negative, 0 for zero, and 1 for positive.

sin(x As Double) As Double

Returns the sine of the argument x. The argument must be in radians. To obtain the sine of x when x is in degrees, use `sin(x*.01745329)`.

tan(x As Double) As Double

Returns the tangent of the argument x. The argument must be in radians. To obtain the tangent of *x* when *x* is in degrees, use `tan(x*.01745329)`.

sqr(x As Double) As Double

Returns the square root of the argument x. This function is the same as *x^(1/2)*, but calculates the result faster.

Left(target_string As String, n As Integer) As String

Returns the first n characters of the specified string.

Right(target_string As String, n As Integer) As String

Returns the last n characters of the specified string.

StringI(n As Integer, character As Integer) As String

Returns a string composed of a character symbol repeated n times. The character symbol is passed to the method as an ASCII code integer.

String(n As Integer, character As String) As String

Returns a string composed of a character symbol repeated n times. The character symbol is passed to the method as a string.

Mid(target_string As String, start_position As Integer, length As Integer) As String

Returns a substring of the target string. The first integer passed to the method specifies the starting position of the substring, and the second integer specifies the length of the substring. The start position of a string begins with 1.

Instr(start_position As Integer, search_text As String, substring As String) As Integer

Returns the position of a substring within a string. This function is case sensitive and returns 0 if the specified substring is not found. The start position of a string begins with 1.

> **Tip**
>
> The string object also offers an `Instr()` method (though it uses a zero-based index). See the *roString* documentation for more details.

GetInterface(object As Object, ifname As String) As Interface

Returns a value of the type Interface. All objects have one or more interfaces. In most cases, you can skip interface specification when calling an object component. This will not cause problems as long as the method names within a function are unique.

Wait(timeout As Integer, port As Object) As Object

Instructs the script to wait on an object that has an *ifMessagePort* interface. This method will return the event object that was posted to the message port. If the timeout is specified as zero, `Wait()` will wait indefinitely; otherwise, `Wait()` will return Invalid after the specified number of milliseconds if no messages have been received.

```
p =   CreateObject("roMessagePort")
sw =  CreateObject("roGpioControlPort")
sw.SetPort(p)
msg=wait(0, p)
print type(msg)     ' should be roGpioButton
print msg.GetInt()  ' button number
```

ReadAsciiFile(file_path As String) As String

Reads the specified text file and returns it as a string.

WriteAsciiFile(file_path As String, buffer As String) As Boolean

Creates a text file at the specified file path. The text of the file is passed as the second parameter. This method cannot be used to edit files: A preexisting text file will be overwritten if it has the same name and directory path as the one being created.

> **Note**
>
> The *roCreateFile* object provides more flexibility if you need to create or edit files.

ListDir(path As String) As Object

Returns an *roList* object containing the contents of the specified directory path. File names are converted to all lowercase.

MatchFiles(path As String, pattern_in As String) As Object

Takes a directory to look in (it can be as simple as "." or "/") and a pattern to be matched and then returns an *roList* containing the results. Each listed result contains only the part of the filename that is matched against the pattern, not the full path. The match is only applied in the specified directory; you will get no results if the pattern contains a directory separator. The pattern is a case insensitive wildcard expression. It may contain the following special characters:

- ? – Matches any single character.
- * – Matches zero or more arbitrary characters.
- […] – Matches any single character specified within the brackets. The closing bracket is treated as a member of the character class if it immediately follows the opening bracket (i.e. "[]]" matches a single closed bracket). Within this class, "-" can be used to specify a range unless it is the first or last character (e.g. "[A-Cf-h"] is equivalent to "[ABCfgh]"). A character class may be negated by specifying "^" as the first character. To match a literal of this character, place it elsewhere in the class.

> **Note**
>
> The special characters "?", "*", and "[" lose their function if preceded by a single "\", and a single "\" can be matched using "\\".

LCase(target_string As String) As String

Converts the specified string to all lowercase.

UCase(target_string As String) As String

Converts the specified string to all uppercase.

DeleteFile(file_path As String) As Boolean

Deletes the file at the specified file path. This method returns False if the delete operation fails or if the file does not exist.

DeleteDirectory(diretory As String) As Boolean

Deletes the specified directory. This method will recursively delete any files and directories that are necessary for removing the specified directory. This method returns False if it fails to delete the directory, but it may still delete some of the nested files or directories.

CreateDirectory(directory As String) As Boolean

Creates the specified directory. Only one directory can be created at a time. This method returns True upon success and False upon failure.

RebootSystem() As Void

Instructs the player to perform a soft reboot.

ShutdownSystem() As Void

UpTime(dummy As Integer) As Double

Returns the uptime of the system (in seconds) since the last reboot.

FormatDrive(drive As String, fs_type As String) As Boolean

Formats the specified drive using one of the file systems listed below. This function returns True upon success and False upon failure:

- `vfat` (FAT32, DOS/Windows file system): Readable and writable by Windows, Linux, and MacOS.
- `exfat` (DOS/Windows file system): Supported on Series 3 (XTx43, XDx33, HDx23, LS423, HO523) and Series 4 (XTx44, XDx34, HDx24, LS424) players with firmware versions 6.2.94 and later. Secure boot is not supported with the exFAT filesystem on firmware version 6.2.94.
- `ext2` (Linux file system): Writable by Linux and readable by Windows and MacOS with additional software.
- `ext3` (Linux file system): Writable by Linux and readable by Windows and MacOS with additional software.
- `ext4` (Linux file system): Writable by Linux and readable by Windows and MacOS with additional software. This is the recommended file system for SSD devices and USB hard drives.

EjectDrive(drive As String) As Boolean

Ejects the specified drive (e.g. "SD:") and returns True if successful. If the script is currently accessing files from the specified drive, the ejection process will fail.

CopyFile(source As String, destination As String) As Boolean

Copies the file at the specified source file-path to the specified destination directory. The function returns True if successful and False in the event of failure.

MoveFile(source As String, destination As String) As Boolean

Moves the specified source file to the specified destination directory. The function returns True if successful and False in the event of failure.

> **Note**
>
> Both path names must be on the same drive.

MapFilenameToNative(path As String) As String

Converts the specified BrightScript-style path to the corresponding native path and returns it as a string (e.g. the path "SD:/mydir" will be returned as "/storage/sd/mydir").

strtoi(target_string As String) As Integer

Converts the target string to an integer. Any non-integer characters (including decimal points and spaces), and any numbers to the right of a non-integer character, will not be part of the integer output.

rnd(a As Dynamic) As Dynamic

RunGarbageCollector() As roAssociativeArray

Destroys objects that are currently in a state of circular reference counting. BrightScript normally removes any objects that become unreferenced as part of its automated garbage collection algorithm. However, objects that reference each other will never reach a reference count of zero, and will need to be destroyed manually using this method.

This method is useful when destroying old presentation data structures and generating a new presentation. This method returns an associative array outlining the results of the garbage-collection process.

GetDefaultDrive() As String

Returns the current default drive complete with a trailing slash. When running as *autorun.brs*, the drive containing the autorun is designated as the current default.

SetDefaultDrive(drive As String)

Sets the current default drive, which does not need to include a trailing slash. This method does not fail; however, if the specified default drive does not exist, it will not be possible to retrieve anything.

This method accepts the following values:

- `"USB1:"` – The drive for USB storage devices connected to the player

- "SD:" – The primary SD or microSD drive on the player.
- "SD2:" – The internal microSD drive on the player (4Kx42, XDx32 models only)
- "SSD:" – The internal SSD on the player (XTx44, XTx34, XDx34, XDx33 models only)

EnableZoneSupport(enable As Boolean)

Allows for display of multiple video, HTML, image, and text zones. As of firmware 6.0.x, zone support is enabled by default.

EnableAudioMixer(enable As Boolean)

Pi() As Double

Returns the value of pi as a double-precision floating-point number.

ParseJson(json_string As String) As Object

Parses a string formatted according to the RFC4627 standard and returns an equivalent BrightScript object, which can consist of the following: Booleans, integers, floating point numbers, strings, *roArray* objects, and *roAssociativeArray* objects. The `ParseJson()` method has the following properties:

- Invalid will be returned if the string is not syntactically correct.
- Any *roAssociativeArray* objects that are returned will be case sensitive.
- An error will be returned if an *roArray* or *roAssociativeArray* is nested more than 256 levels deep.

The following script demonstrates how to use `ParseJson()` to process a JSON object containing the titles and URLs of a set of images.

JSON Script

```
{
"photos" : [
        {
              "title" : "View from the hotel",
              "url" : "http://example.com/images/00012.jpg"
        },
        {
              "title" : "Relaxing at the beach",
              "url" : "http://example.com/images/00222.jpg"
        },
        {
              "title" : "Flat tire",
              "url" : "http://example.com/images/00314.jpg"
        }
]
}
```

BrightScript

```
searchRequest = CreateObject("roUrlTransfer")
searchRequest.SetURL("http://api.example.com/services/rest/getPhotos")
response = ParseJson(searchRequest.GetToString())
For Each photo In response.photos
        GetImage(photo.title, photo.url)
End For
```

FormatJson(json As roAssociativeArray, flags As Integer) As String

Converts an associative array to a JSON string (i.e. formatted according to the RFC4627 standard). The following are supported data types: Boolean, Integer, Float, String, *roArray*, and *roAssociativeArray*. If the `flags` parameter is set to 0 or not specified, non-ASCII characters are escaped in the output string as "\uXXXX", where "XXXX" is the hexadecimal representation of the Unicode character value. If the `flags` parameter is set to 1, non-ASCII characters are not escaped.

If arrays or associative arrays are nested more than 256 levels deep, an error will occur. If an error occurs, an empty string will be returned.

> **Important**
>
> By default, using object-literal syntax (e.g. `aa={relativePath:"Foo"}`) to generate an associative array will convert keys to all lower case. To preserve camel case for JSON, use the *roAssociativeArray.AddReplace()* method instead of object literals or call *roAssociativeArray.SetModeCaseSensitive()* before adding entries.