ROMBASIC DEVELOPERS' MANUAL

For the Game *Terrarum* · First Edition

Contents

Chapter 1

APIs and Libraries

1.1   Filesystem

The Filesystem API provides functions for manipulating files and the filesystem.

The path for the argument of functions blocks '..' to be entered, preventing users from access outside of the computer and eliminating the potential of harming the real computer of the innocent players.

1.1.1   Functions

Function	Return	Description
fs.list(**path**: string)	table	Returns list of files in **path**, in lua table.
fs.exists(**path**: string)	bool	Checks if **path** exists on the filesystem.
fs.isDir(**path**: string)	bool	Checks if **path** is a directory.
fs.isFile(**path**: string)	bool	Checks if **path** is a file.
fs.isReadOnly(**path**: string)	bool	Checks if **path** is read only.
fs.getSize(**path**: string)	int	Returns a size of the file/directory, in bytes.
fs.mkdir(**path**: string)	bool	Create a directory to **path**. Returns **true** upon success.
fs.mv(**from**: string, **dest**: string)	bool	Moves the directory to the destination. Subdirectories / files will also be moved. Returns **true** upon success.
fs.cp(**from**: string, **dest**: string)	bool	Copies the directory to the destination. Subdirectories / files will also be copied. Returns **true** upon success.

Function	Return	Description
fs.rm(**path**: string)	bool	Deletes the **path**. If **path** is a directory, all its members will also be deleted. Returns **true** upon success.
fs.concat(**p1**: string, **p2**: string)	string	Concatenates two paths and return new path as string.
fs.open(**path**: string, **mode**: string)	file	Opens file and returns its handle. See section *File Handler* for details.
fs.parent(**path**: string)	string	Returs parent directory to the **path**.
fs.dofile(**path**: string)	nil	Loads the script on **path** and executes it.
fs.fetchText(**path**: string)	string	Opens the file on **path** and returns its contents as a plain text.

1.1.2 File Handler

When it comes to opening a file, there are six modes available—r, w, a, rb, wb, ab, each represents **r**ead, **w**rite, **a**ppend and **b**yte.

Function	Description
file.close()	Closes the file. Any data wrote will be actually wrote to disk when this function is called.
file.flush()	(in write/append mode) Flushes the data to the file, keeps the handle available afterwards
Read mode	
file.readLine()	Reads text from the file line by line. Returns string of line, or *nil* if there is no next line.

Function	Description
file.readAll()	Reads and returns whole text in the file as string.
Read binary mode	
file.read()	Reads single byte in the file as int, or *-1* if end-of-file is reached.
file.readAll()	Reads and returns whole byte in the file as string.
Write/append mode	
file.write(string)	Writes **string** to the file.
file.writeLine(string)	Writes **string** to the file and append line feed.
Write/append binary mode	
file.write(int)	Writes **int** to the file.
file.writeBytes(string)	Writes **string** to the file and append line feed.

1.2 Hexutils

The Hexutils library provides utility to convert byte value to hexadecimal string.

1.2.1 Functions

Function	Return	Description
hexutils.toHexString(**bytes**: string)	string	Converts byte array to the string of its hexadecimal representations.

1.3 Security

The Serurity API provides functions for security purposes, such as hashing and CSPRNG[1].

1.3.1 Functions

Function	Return	Description
security.toSHA1(string)	string	Returns SHA-256 hash of input string in array of bytes (as a string)
security.toSHA256(string)	string	Returns SHA-1 hash of input string in array of bytes
security.toMD5(string)	string	Returns MD-5 hash of input string in array of bytes
security.randomBytes(**len**: int)	string	Returns byte array of random values in desired **len**gth.
security.decodeBase64(string)	string	Decodes Base64 string and returns the result as string.
security.encodeBase64(string)	string	Encodes input string as Base64 format and returns the result as array of bytes.

---

[1]Cryptographically secure psuedo-random number generator

1.4   Shell

Function	Return	Description
shell.run(**path**: string)	nil	Loads the script on **path** and executes it.

1.5   Terminal

The Terminal API provides functions for sending text to the terminals, and drawing text-mode graphics. The API expects connected terminal to use Codepage 437. See section *Codepage* for details.

1.5.1   Functions

Note: cursor coordinates starts from one, not zero.

Function	Return	Description
term.write(string)	nil	Writes string to the current cursor position. Line feed is not appended.
term.print(string)	nil	Writes string to the current cursor position and make a new line.
term.newLine()	nil	Make a new line.
term.moveCursor(**x**: int)	nil	Moves cursor horizontally, starting from 1.
term.width()	int	Returns the width of the terminal. Graphic terminals also can use this.
term.scroll(**n**: int)	nil	Make a new line **n** times.
term.isTeletype()	bool	Returns **true** if the terminal is teletype.
Graphic terminals only		
term.emit(**c**: int, **x**: int, **y**: int)	nil	Emits **c** into (**x**, **y**), control sequence will not be processed and printed as symbols instead. Cursor will not be moved.

Function	Return	Description
term.emitRaw(**bufferChar**: int)	nil	Emits **bufferChar** into into (**x**, **y**). Buffer char means a single character actually stored into the screen buffer, has four bits for back- and foreground colours respectively, and eight bits for a letter.
term.emitString(**s**, **x**: int, **y**: int)	nil	Emits **s** (a string) into (**x**, **y**), printing control sequences as symbols. Cursor will not be moved.
term.resetColour() term.resetColor()	nil	Resets any colour changes to the defaults.
term.clear()	nil	Clears whole screen buffer and move cursor to (1, 1)
term.clearLine()	nil	Clears current line on the screen buffer, does not moves cursor.
term.setCursor(**x**: int, **y**: int)	nil	Moves cursor to (**x**, **y**)
term.getCursor()	int, int	Returns current coordinates of the cursor.
term.getX()	int	Returns X coordinate of the cursor.
term.getY()	int	Returns Y coordinate of the cursor.
term.setX(int)	nil	Sets X coordinate of the cursor.
term.setY(int)	nil	Sets Y coordinate of the cursor.
term.blink(bool)	nil	Sets cursor blinking. **true** makes the cursor blink.
term.size()	int, int	Returns width and height of the terminal.

Function	Return	Description
term.height()	int	Returns height of the terminal.
term.isCol()	bool	Returns if the terminal supports colour.
term.setForeCol(**col**: int)	nil	Sets foreground colour to **col**
term.setBackCol(**col**: int)	nil	Sets background colour to **col**.
term.foreCol()	int	Returns current foreground colour.
term.backCol()	int	Returns current background colour.

1.5.2   Standard Colours

0	Black	1	White	2	Dim grey	3	Bright grey
4	Yellow	5	Orange	6	Red	7	Magenta
8	Purple	9	Blue	10	Cyan	11	Lime
12	Green	13	Dark green	14	Brown	15	Tan

Non-colour terminals support colour index of 0–3.

1.5.3 Codepage

Character 0x9D (currency symbol) and 0xFA (middle dot) can be accessed with following Lua constants: *MONEYSYM* and *MIDDOT*.

1.5.4 Accepted Control Sequences

No.	Description	No.	Description
7	BEL. Emits short beep.	8	BS. Moves cursor to left 1 character.
9	TAB. Inserts appropriate horizontal space. Tab size is variable.	10	LF. Prints a new line.
12	FF. Clears everything in screen buffer and moves cursor to (1, 1)	13	CR. Moves x coordinate of cursor to 1.
16	DLE. Sets foreground colour to the default STDERR colour.	127	DEL. Backspace and deletes one character.

No.	Description	No.	Description
17	DC1. Sets foreground colour to 0. (black)	18	DC2. Sets foreground colour to 1. (white)
19	DC3. Sets foreground colour to 2. (dim grey)	20	DC4. Sets foreground colour to 3. (bright grey)

1.6 Lua Globals

ROMBASIC adds global functions and constants for operability.

1.6.1 Functions

Function	Return	Description
_G.runScript(**fun**: str, **env**: str)	nil	Runs Lua script **fun** with the environment tag **env**.
_G.getMem()	int	Returns the current memory usage in bytes.
_G.getTotalMem()	int	Returns the total size of the memory installed in the computer, in bytes.
_G.getFreeMem()	int	Returns the amount of free memory on the computer.

1.6.2 Constants

Name	Type	Description
_G.MONEYSYM	string	Currency symbol used in the world. Code 0x9D
_G.MIDDOT	string	Middle dot used in typography. Code 0xFA (note: 0xF9 is a Dot Product used in Mathematics)
_G.DC1	string	Ascii control sequence DC1. Used to change foreground colour to black.
_G.DC2	string	Ascii control sequence DC2. Used to change foreground colour to white.
_G.DC3	string	Ascii control sequence DC3. Used to change foreground colour to dim grey.

Name	Type	Description
_G.DC4	string	Ascii control sequence DC4. Used to change foreground colour to bright grey.
_G.DLE	string	Ascii control sequence DLE. Used to change foreground colour to terminal's default error text colour.
_COMPUTER.prompt	string	Default text for prompt input indicator.
_COMPUTER.verbose	bool	Sets whether print debug information to the console.
_COMPUTER.loadedCLayer	table	List of names of compatibility layers has been loaded.
_COMPUTER.bootloader	string	Path to the boot file. Should point to the EFI (/boot/efi).
_COMPUTER.OEM	string	Manufacturer of the computer. If you *are* a manufacturer, you may want to fill in this variable with your own company's name.

Chapter 2

Compatibility Layers
ComputerCraft

2.1   Bit

The Bit API is for manipulating numbers using bitwise binary operations. The ROM-BASIC already comes with Lua's bit32 library so make sure to use that for your casual usage.

2.1.1   Functions

Function	Notes
bit.blshift(n, bits)	Alias of bit32.lshift(n, bits)
bit.brshift(n, bits)	Alias of bit32.arshift(n, bits)
bit.blogic_rshift(n, bits)	Alias of bit32.brshift(n, bits)
bit.bxor(m, n)	Alias of bit32.bxor(m, n)
bit.bor(m, n)	Alias of bit32.bor(m, n)
bit.band(m, n)	Alias of bit32.band(m, n)
bit.bnot(n)	Alias of bit32.bnot(n)

2.2 Colors

The Colors API allows you to manipulate sets of colors. This is useful in colors on Advanced Computers and Advanced Monitors. British spellings are also supported.

2.2.1 Constants

When the colours are used in ComputerCraft's Term API, nearest console colours will be used. Below is the table of colours coded with their substitutions.

colors.white	colors.orange	colors.magenta	colors.lightBlue
colors.yellow	colors.lime	colors.pink	colors.gray
colors.lightGray	colors.cyan	colors.purple	colors.blue
colors.brown	colors.green	colors.red	colors.black

Note that pink is understood as tan when it is used, lightBlue and cyan are merged to cyan.

2.2.2 Functions

All three functions are not supported, as there is no bundled cable thus there is no use of them.

2.3 Term

2.4 Filesystem

Chapter 3

Compatibility Layers
OpenComputers

Chapter 4

Peripherals

4.1   Line Printer

The line printer is a printer that operates on line basis. It only prints text in line-by-line, hence the name, on almost endlessly long roll of papers; it has no notion of page, it just prints. If you want some pages to keep, you must tear them out yourself.

Line printers do not work indefinitely; ignoring the obvious depletion of ink, belt for loading paper will be out of service on about 50 000 lines of printing, give or take a few, or paper will jam if the printer had struck with the unluckiness.

4.1.1   Functions

Function	Return	Description
lp.print(string)	nil	Prints a line of string.
lp.scroll(**n**: int)	nil	Scrolls the paper by **n** lines.
lp.status()	int	Returns a status of the line printer.
lp.reset()	nil	Resets the line printer.

Chapter 5

References

Some of the texts are taken from following sources:

- Lua Manual version 5.2, Lua.org, POC-Rio
- ComputerCraft, dan200
- OpenComputers, MightyPirates