lsnes Lua functions reference

May 8, 2015

1 Table of contents

Contents

2 Conventions

2.1 Coordinates:

- Coordinates increase to right and down.
- The origin is at top left of game area or buffer.

2.2 Drawing and contexts

- Methods that draw something (unless stated otherwise) require a valid rendering context. This context can come in three ways:
  1. The default rendering context of paint callback (the screen).
  2. The default rendering context of video callback (the video).
  3. Explicitly set rendering context (RENDERCTX:set).
- The rendering context is always reset when callback ends.

2.3 Colors

(Direct) colors can be specified either as numbers or strings.

- -1 is fully transparent.
- Non-negative numbers less than 2^{32} are partially opaque colors ($a * 2^{24} + r * 2^{16} + g * 2^8 + b$)
  - a is transparency $0 - 255$, 0 is fully opaque, 256 would be fully transparent.
    - Thus, numbers in range $0 - 16777215$ stand for fully opaque colors.
  - r, g and b are intensities of base colors on scale $0 - 255$.
- Color can also be specified by name as string: The following color names are known: aliceblue antiquewhite antiquewhite1 antiquewhite2 antiquewhite3 antiquewhite4 aqua aquamarine aquamarine1 aquamarine2 aquamarine3 aquamarine4 azure azure1 azure2 azure3 azure4 beige bisque bisque1 bisque2 bisque3 bisque4 black blanchedalmond blue blue1 blue2 blue3 blue4 blueviolet brown brown1 brown2 brown3 brown4 burlywood burlywood1 burlywood2 burlywood3 burlywood4 cadet cadetblue cadetblue1 cadetblue2 cadetblue3 cadetblue4 chartreuse chartreuse1 chartreuse2 chartreuse3 chartreuse4 chocolate chocolate1 chocolate2 chocolate3 chocolate4 coral coral1 coral2 coral3 coral4 cornflowerblue cornsilk cornsilk1 cornsilk2 cornsilk3 cornsilk4 crimson cyan cyan1 cyan2 cyan3 cyan4 darkblue darkcyan darkgoldenrod darkgoldenrod1 darkgoldenrod2 darkgoldenrod3 darkgoldenrod4 darkgray darkgreen darkgrey darkkhaki darkmagenta darkolivegreen darkolivegreen1 darkolivegreen2 darkolivegreen3 darkolivegreen4 darkorange darkorange1 darkorange2 darkorange3 darkorange4 darkorchid darkorchid1 darkorchid2 darkorchid3 darkorchid4 darkred darksalmon darkseagreen darkseagreen1 darkseagreen2 darkseagreen3 darkseagreen4 darkslateblue darkslategray darkslategray1 darkslategray2 darkslategray3 darkslategray4 darkslategrey darkturquoise darkviolet deeppink deeppink1 deeppink2 deeppink3 deeppink4 deepskyblue deepskyblue1 deepskyblue2 deepskyblue3 deepskyblue4 dimgray dimgrey dodgerblue dodgerblue1 dodgerblue2 dodgerblue3 dodgerblue4 firebrick firebrick1 firebrick2 firebrick3 firebrick4 floralwhite forestgreen fractal fuchsia gainsboro ghostwhite gold gold1 gold2 gold3 gold4 goldenrod goldenrod1 goldenrod2 goldenrod3 goldenrod4 gray gray0 gray1 gray10 gray100 gray11 gray12 gray13 gray14 gray15 gray16 gray17 gray18 gray19 gray2 gray20 gray21 gray22 gray23 gray24 gray25 gray26 gray27 gray28 gray29 gray3 gray30 gray31 gray32 gray33 gray34 gray35 gray36 gray37 gray38 gray39 gray4 gray40 gray41 gray42 gray43 gray44 gray45 gray46 gray47 gray48 gray49 gray5 gray50 gray51 gray52 gray53 gray54 gray55 gray56 gray57 gray58 gray59 gray6 gray60 gray61 gray62 gray63 gray64 gray65 gray66 gray67 gray68 gray69 gray7 gray70 gray71 gray72 gray73 gray74 gray75 gray76 gray77 gray78 gray79 gray8 gray80 gray81 gray82 gray83 gray84 gray85 gray86 gray87 gray88 gray89 gray9 gray90 gray91 gray92 gray93 gray94 gray95 gray96 gray97 gray98 gray99 green green1 green2 green3 green4 greenyellow grey grey0 grey1 grey10 grey100 grey11 grey12 grey13 grey14 grey15 grey16 grey17 grey18 grey19 grey2 grey20 grey21 grey22 grey23 grey24 grey25 grey26 grey27 grey28 grey29 grey3 grey30 grey31 grey32 grey33 grey34 grey35 grey36 grey37 grey38 grey39 grey4 grey40 grey41 grey42 grey43 grey44 grey45 grey46 grey47 grey48 grey49 grey5 grey50 grey51 grey52 grey53 grey54 grey55 grey56 grey57 grey58 grey59 grey6 grey60 grey61 grey62 grey63 grey64 grey65 grey66 grey67 grey68 grey69 grey7 grey70 grey71 grey72 grey73 grey74 grey75 grey76 grey77 grey78 grey79 grey8 grey80 grey81 grey82 grey83 grey84 grey85 grey86 grey87 grey88 grey89 grey9 grey90 grey91 grey92 grey93 grey94 grey95 grey96 grey97 grey98 grey99 honeydew honeydew1 honeydew2 honeydew3 honeydew4 hotpink hotpink1 hotpink2 hotpink3 hotpink4 indianred indianred1 indianred2

indianred3 indianred4 indigo ivory ivory1 ivory2 ivory3 ivory4 khaki khaki1 khaki2 khaki3 khaki4 lavender lavenderblush lavenderblush1 lavenderblush2 lavenderblush3 lavenderblush4 lawngreen lemonchiffon lemonchiffon1 lemonchiffon2 lemonchiffon3 lemonchiffon4 lightblue lightblue1 lightblue2 lightblue3 lightblue4 lightcoral lightcyan lightcyan1 lightcyan2 lightcyan3 lightcyan4 lightgoldenrod lightgoldenrod1 lightgoldenrod2 lightgoldenrod3 lightgoldenrod4 lightgoldenrodyellow lightgray lightgreen lightgrey lightpink lightpink1 lightpink2 lightpink3 lightpink4 lightsalmon lightsalmon1 lightsalmon2 lightsalmon3 lightsalmon4 lightseagreen lightskyblue lightskyblue1 lightskyblue2 lightskyblue3 lightskyblue4 lightslateblue lightslategray lightslategrey lightsteelblue lightsteelblue1 lightsteelblue2 lightsteelblue3 lightsteelblue4 lightyellow lightyellow1 lightyellow2 lightyellow3 lightyellow4 lime limegreen linen magenta magenta1 magenta2 magenta3 magenta4 maroon maroon1 maroon2 maroon3 maroon4 mediumaquamarine mediumblue mediumforestgreen mediumgoldenrod mediumorchid mediumorchid1 mediumorchid2 mediumorchid3 mediumorchid4 mediumpurple mediumpurple1 mediumpurple2 mediumpurple3 mediumpurple4 mediumseagreen mediumslateblue mediumspringgreen mediumturquoise mediumvioletred midnightblue mintcream mistyrose mistyrose1 mistyrose2 mistyrose3 mistyrose4 moccasin navajowhite navajowhite1 navajowhite2 navajowhite3 navajowhite4 navy navyblue oldlace olive olivedrab olivedrab1 olivedrab2 olivedrab3 olivedrab4 orange orange1 orange2 orange3 orange4 orangered orangered1 orangered2 orangered3 orangered4 orchid orchid1 orchid2 orchid3 orchid4 palegoldenrod palegreen palegreen1 palegreen2 palegreen3 palegreen4 paleturquoise paleturquoise1 paleturquoise2 paleturquoise3 paleturquoise4 palevioletred palevioletred1 palevioletred2 palevioletred3 palevioletred4 papayawhip peachpuff peachpuff1 peachpuff2 peachpuff3 peachpuff4 peru pink pink1 pink2 pink3 pink4 plum plum1 plum2 plum3 plum4 powderblue purple purple1 purple2 purple3 purple4 red red1 red2 red3 red4 rosybrown rosybrown1 rosybrown2 rosybrown3 rosybrown4 royalblue royalblue1 royalblue2 royalblue3 royalblue4 saddlebrown salmon salmon1 salmon2 salmon3 salmon4 sandybrown seagreen seagreen1 seagreen2 seagreen3 seagreen4 seashell seashell1 seashell2 seashell3 seashell4 sienna sienna1 sienna2 sienna3 sienna4 silver skyblue skyblue1 skyblue2 skyblue3 skyblue4 slateblue slateblue1 slateblue2 slateblue3 slateblue4 slategray slategray1 slategray2 slategray3 slategray4 slategrey snow snow1 snow2 snow3 snow4 springgreen springgreen1 springgreen2 springgreen3 springgreen4 steelblue steelblue1 steelblue2 steelblue3 steelblue4 tan tan1 tan2 tan3 tan4 teal thistle thistle1 thistle2 thistle3 thistle4 tomato tomato1 tomato2 tomato3 tomato4 transparent turquoise turquoise1 turquoise2 turquoise3 turquoise4 violet violetred violetred1 violetred2 violetred3 violetred4 wheat wheat1 wheat2 wheat3 wheat4 white whitesmoke yellow yellow1 yellow2 yellow3 yellow4 yellowgreen

- The HSL base color names: hsl-<hue><saturation><lightness>.
  - Hue can be one of: r (red), ry (red-yellow), o (orange, same as red-yellow), y (yellow), yg (yellow-green), g (green), gc (green-cyan), c (cyan), cb (cyan-blue), b (blue), bm (blue-magenta), m (magenta), mr (magenta-red).
  - Saturation is 0-8, where 0 is greyscale and 8 is fully saturated.
  - Lightness is 0-8, where 0 is black and 8 is white.
- The color names can have a modifier after space (multiple modifiers are allowed, separated by spaces): opaque10, opaque20, opaque25, opaque30, opaque40, opaque50, opaque60, opaque70, opaque75, opaque80, opaque90, opaque, hue{+,-}{1-23}, {saturation,lightness}{+,-}{1-16}.
  - opaqueness is measured as percentage.
  - Hue adjustments are on 24-step scale.
  - Saturation and lightness adjustments are on 16-step scale.

3 Special tokens

These tokens are special, and are expanded while the script is being loaded

3.1 @@LUA_SCRIPT_FILENAME@@

Expanded to string token containing path and filename of this Lua script. Handy for referencing other lua scripts or resources that are relative to this Lua script.

In practicular, this is suitable to be passed as base argument of various functions like loadfile, dofile, resolve_filename, gui.bitmap_load, gui.bitmap_load_png and gui.bitmap_load_pal.

4 Classes

4.1 MMAP_STRUCT: Quasi-table mapping emulated memory

Objects act like tables, but the values reflect emulated memory.

4.1.1 static function new: Create a new instance

- Syntax: mmap memory.mmap.new()
- Syntax: mmap classes.MMAP_STRUCT.new()

Return value:

- mmap: MMAP_STRUCT: The created mmap structure.

Create a new object (with no mappings) and return it.

4.1.2 operator(): Bind key in mmap structure

- Syntax: obj(key, {marea, address|addrobj}, type)

Parameters:

- obj: MMAP_STRUCT: The structure to manipulate.
- key: String: The name of the key in array to map.
- marea: String: The memory area the mapped address is in (default: global memory space).
- address: Number: The offset of memory addreess, relative to specified memory area or global memory space.
- addrobj: ADDRESS: The memory address.
- type: String: The type of data to map.
  - One of: byte, sbyte, word, sword, hword, shword, dword, sdword, qword, sqword, float or double.

Bind key <key> in mmap structure <obj> to address <address> (relative to <marea>). The memory address is treated as type <type>.

- Example: foomap("bar", "WRAM", 0x2A, "sword")

This binds signed word at address WRAM+0x2A into key "bar".

4.2 ZIPWRITER: Write .zip files

This class does writing of .zip files.

4.2.1 Static function new: Create a new zipfile

- Syntax: zip zip.writer.new(filename, [compression])
- Deprecated: zip zip.create(filename, [compression])

Parameters:

- filename: string: The name of the file to write.
- compression: number: Compression level (0-9). Default is 9.

Return value:

- zip: ZIPWRITER: The newly created ZIP writer.

Create a new ZIPWRITER object and return it.

4.2.2 Method commit: Finish creating ZIP file.

- Syntax: zipfile:commit()

Parameters:

- zipfile: ZIPFILE: The ZIP file object.

Finish writing the ZIP file and actually create it on disk.

- If a member is currently open, it is implicitly closed.
- Invoking this on already committed or rolled back zipfile causes an error.

4.2.3 Method rollback: Cancel writing the ZIP file.

- Syntax: zipfile:rollback()

Parameters:

- zipfile: ZIPFILE: The ZIP file object.

Cancel writing the whole ZIP file. The file on disk will not be modified.

- If a member is currently open, it is implicitly closed.
- Invoking this on already committed or rolled back zipfile causes an error.

4.2.4 Method create_file: Start writing a new member

- Syntax: zipfile:create_file(filename)

Parameters:

- zipfile: ZIPFILE: The ZIP file object.
- string filename: Name of the new member to create

Start writing a new member <filename> in ZIP file.

- If a member is currently open, it is implicitly closed.
- Invoking this on already committed or rolled back zipfile causes an error.

4.2.5 Method close_file: Close member

- Syntax: zipfile:close_file()

Parameters:

- zipfile: ZIPFILE: The ZIP file object.

Close the currently open member in zipfile.

- Invoking this on already committed or rolled back zipfile causes an error.
- Invoking this without an open member causes an error.

4.2.6 Method write: Write data

- Syntax: zipfile:write(data)

Parameters:

- zipfile: ZIPFILE: The ZIP file object.
- data: string: The data to write.

Write <data> in binary mode (as-is) to currently open member.

- Invoking this without a member being open causes an error.

4.3 TILEMAP: Tiled collection of bitmaps.

A tilemap is tiled table, each cell holding a bitmap (indexed or direct).

4.3.1 Static function new: Create a tilemap

- Syntax: tilemap gui.tiled_bitmap.new(w, h, bw, bh);
- Syntax: tilemap classes.TILEMAP.new(w, h, bw, bh);
- Deprecated: tilemap gui.tilemap(w, h, bw, bh);

Parameters:

- w: number: Width of the tilemap in tiles
- h: number: Height of the tilemap in tiles
- bw: number: Width of each tile in pixels.
- bh: number: Height of each tile in pixels.

Return value:

- tilemap: TILEMAP: the newly created tilemap

Create a new tilemap of <w>*<h> tiles, each of <bw>*<bh> pixels and return it.

4.3.2 TILEMAP:getsize: Query tilemap size

- Syntax: width, height tmap:getsize()

Parameters:

- tmap: TILEMAP: The tilemap to query.

Return value:

- width: number : The width of the tilemap in tiles.
- height: number: The height of the tilemap in tiles.

Get size of tilemap in tiles.

4.3.3 TILEMAP:getcsize: Query tilemap cell size

- Syntax: width, height tmap:getcsize()

Parameters:

- tmap: TILEMAP: The tilemap to query.

Return value:

- width: number: The width of tilemap tile in pixels.
- height: number: The height of tilemap tile in pixels.

Get size of each tilemap tile in pixels.

4.3.4 TILEMAP:get: Query tilemap tile

- Syntax: bitmap, palette tmap:get(x, y)

Parameters:

- tmap: TILEMAP: The tilemap to query.
- x: number: The x-coordinate of tile to query.
- y: number: The y-coordinate of tile to query.

Return value:

- bitmap: The associated bitmap (BITMAP or DBITMAP), or nil if none.
- palette: The palette (PALETTE) associated with indexed-color bitmap, if any. Otherwise nil.

Return the contents of tile at <x>,<y>.

4.3.5 TILEMAP:set: Set tilemap cell

- Syntax: tmap:set(x, y)
- Syntax: tmap:set(x, y, bitmap)
- Syntax: tmap:set(x, y, bitmap, palette)

Parameters:

- tmap: TILEMAP: The tilemap to manipulate.
- number x: The x-coordinate of tile to set.
- number y: The y-coordinate of tile to set.
- bitmap: BITMAP/DBITMAP: The bitmap to set to tile.
- palette: PALETTE: The associated palette for bitmap.

Set the contents of tile <x>,<y>. If no <bitmap> is given, clears the tile. Otherwise sets the tile to contain bitmap <bitmap> with associated palette <palette>.

- Specifying index bitmap <bitmap> without palette <palette> causes an error.

4.3.6 TILEMAP:scroll: Scroll tilemap

- Syntax: tmap:scroll(ox, oy);
- Syntax: tmap:scroll(ox, oy, x, y, w, h);
- Syntax: tmap:scroll(ox, oy, x, y, w, h, circ_x, circ_y);

Parameters:

- tmap: TILEMAP: The tilemap to manipulate.
- ox: number: The number of tiles to scroll horizontally (positive is to right).
- oy: number: The number of tiles to scroll vertically (positive is to down).
- x: number: The left edge of scroll window.
- y: number: The top edge of scroll window.
- w: number: The width of scroll window.
- h: number: The height of scroll window.
- circ_x: boolean: If true, treat the window as circular in horizontal direction.
- circ_y: boolean: If true, treat the window as circular in vertical direction.

Scroll the specified tilemap <tmap> by <ox>,<oy>. If a window <x>,<y>,<w>,<h> is specified, only that part is scrolled. <circ_x> and <circ_y> control if scroll is circular or not.

- Specifying out-of-range window causes an error.

4.3.7 TILEMAP:draw: Draw tilemap

- Name: tmap:draw(x, y)
- Name: tmap:draw(x, y, x0, y0, w, h)

Parameters:

- tmap: TILEMAP: The tilemap to draw.
- x: The x coordinate on screen to draw to.
- y: The y coordinate on screen to draw to.

- x0: number: The x coordinate on tilemap (in pixels) to start drawing from.
- y0: number: The y coordinate on tilemap (in pixels) to start drawing from.
- w: number: The width to draw (in pixels).
- h: number: The height to draw (in pixels).

Draw tilemap <tmap> to screen at <x>,<y>. If a window <x0>,<y0>,<w>,<h> is specified, only that part is drawn.

- This method requires a rendering context to work.

4.3.8 TILEMAP:draw_outside: Draw tilemap outside game area

- Name: tmap:draw_outside(x, y)
- Name: tmap:draw_outside(x, y, x0, y0, w, h)

Like TILEMAP:draw, but draws only outside game area.

4.4 RENDERCTX: Off-screen rendering context

Object acting as off-screen rendering context.

4.4.1 Static function new: Create a render queue

- Syntax: renderctx gui.renderctx.new(width, height);
- Syntax: renderctx classes.RENDERCTX.new(width, height);
- Deprecated: renderctx gui.renderq_new(width, height);

Parameters:

- width: number: The notional width of the game area.
- height: number: The notional height of the game area.

Returns:

- rendectx: RENDERCTX: The newly created render context.

Create a render context with reported size <width>*<height> and return it.

4.4.2 Static function setnull: Reset to default queue

- Syntax: gui.renderctx.setnull()
- Syntax: classes.RENDERCTX:setnull()
- Deprecated: gui.renderq_set()

Reset the used render context back to default for the executing callback:

- The default for paint callback is the screen
- The default for video callback is the video frame
- The default otherwise is nothing.

4.4.3 Method clear: Clear a render queue

- Syntax: renderctx:clear()
- Deprecated: gui.renderq_clear(renderctx)

Parameters:

- renderctx: RENDERCTX: The render queue to clear.

Clear all drawing from the context.

4.4.4 Method set: Change active render context

- Syntax: renderctx:set()
- Deprecated: gui.renderq_set(renderctx)

Parameters:

- renderctx: RENDERCTX: The render queue to use.

Switch the current rendering context <renderctx>.

4.4.5 Method run: Draw all objects in context to another

- Syntax: renderctx:run()
- Deprecated: gui.renderq_run(renderctx)

Parameters:

- renderctx: RENDERCTX: The render context to overlay.

Overlay the specified render context <context> upon the active rendering context.

- Trying to overlay rendering context upon itself is a bad idea.

4.4.6 Method render: Render a contex to bitmap

- Syntax: bitmap renderctx:render()

Parameters:

- renderctx: RENDERCTX: The context to render.

Returns:

- bitmap: DBITMAP: The rendered bitmap.

Render the specified context <renderctx> to a new bitmap.

- The size of bitmap will be nominal game area size, plus any set gaps.
- This method does not require active rendering context.

4.4.7 Method synchronous_repaint: Paint screen now

- Syntax: renderctx:synchronous_repaint()
- Deprecated: gui.synchronous_repaint(renderctx)

Parameters:

- renderctx: RENDERCTX: The context to paint.

Immediately redraw the screen with game overlayed by drawings from context <renderctx>.

- This does not require active rendering context.
- Will not cause paint callback to be invoked.

4.5 PALETTE: Color palette for indexed image

4.5.1 Static function new: Create a new palette

- Syntax: palette gui.palette.new()
- Syntax: palette classes.PALETTE.new()
- Deprecated: palette gui.palette_new()

Returns:

- palette: PALETTE: The created palette.

Create a new palette (with all colors transparent) and return it.

4.5.2 Static function load: Load a palette

- Syntax: palette gui.palette.load(file, [base])
- Syntax: palette classes.PALETTE.load(file, [base])
- Deprecated: palette gui.palette_load(file, [base])

Parameters:

- string file: The file to load.
- string base (optional): The base file to resolve file relative to.

Returns:

- palette: PALETTE: The loaded palette.

Load a palette from file <file> (resolved relative to <base>).

- The file format is a series of lines, each with following format:
  - Blank or just whitespace: Ignored
  - First non-whitespace is '#': Ignored
  - <r> <g> <b>: Fully opaque color with specified RGB values (0-255)
  - <r> <g> <b> <a>: Color with specified RGB values (0-255) and specified alpha (0-256, 0 being fully transparent and 256 fully opaque).
  - transparent: Fully transparent color

4.5.3 Static function load_str: Load a palette from string

- Syntax: palette gui.palette.load(data)
- Syntax: palette classes.PALETTE.load(data)
- Deprecated: palette gui.palette_load(data)

Parameters:

- string data: The palette data.

Returns:

- palette: PALETTE: The loaded palette.

Like PALETTE:load, but instead of reading palette from file, reads it from a string.

4.5.4 Method set: Set palette entry

- Syntax: palette:set(index, color)
- Deprecated: gui.palette_set(palette, ...)

Parameters:

- palette: PALETTE: The palette to manipulate
- index: number: The index of color to set (0-65535).
- color: number/string: The color value to set.

Set palette <palette> index <index> to color <color>.

4.5.5 Method get: Get palette entry

- Syntax: number palette:get(index)

Parameters:

- palette: PALETTE: The palette to query
- index: number: The index of color to get (0-65535).

Returns:

- The palette entry as integer.

Get palette entry <index> of palette <palette> and return it.

4.5.6 Method hash: Hash a palette

- Syntax: hash palette:hash()
- Deprecated: hash gui.palette_hash(palette)

Parameters:

- palette: The palette to hash.

Return value:

- hash: string: 64-hex digit hash.

Obtain crypto-grade hash of palette data of <palette>.

- All colors after the last non-transparent one are ignored.

4.5.7 Method adjust_transparency: Adjust transparency

- Syntax: palette:adjust_transparency(newvalue)
- Deprecated: gui.adjust_transparency(palette, ...)

Parameters:

- palette: PALETTE: The palette to adjust.
- Number adj: The factor to multiply opaqueness with times 256.

Multiply opaqueness of all colors in palette <palette> by factor of <adj>/256.

4.6 BITMAP: Indexed-color bitmap

4.6.1 Static function new: Create a new bitmap

- Syntax: bitmap gui.bitmap.new(w, h, [fillcolor])
- Syntax: bitmap classes.BITMAP.new(w, h, [fillcolor])
- Deprecated: bitmap gui.bitmap_new(w, h, false, [fillcolor])

Parameters:

- w: number: The width of bitmap to create in pixels.
- h: number: The height of bitmap to create in pixels.
- false: boolean: Constant boolean false.
- fillcolor: number: The initial fill index. Default 0 if not specified.

Create a new bitmap of size <w>*<h>. Fill the bitmap with color index <fillcolor>.

4.6.2 Method draw: Draw a bitmap

- Syntax: bitmap:draw(x, y, palette)
- Deprecated: gui.bitmap_draw(x, y, bitmap, palette)

Parameters:

- bitmap: BITMAP: The bitmap to draw
- x: number: The x-coordinate on screen.
- y: number: The y-coordinate on screen.
- palette: The palette to use for drawing.

Draw bitmap <bitmap> on screen at <x>,<y> with palette <palette>.

4.6.3 Method draw_outside: Draw a bitmap outside game area

- Syntax: bitmap:draw_outside(x, y, palette)

Like bitmap:draw, but does not draw on game area.

4.6.4 Method draw_clip: Draw a bitmap, with clipping

- Syntax: bitmap:draw_clip(x, y, palette, x0, y0, width, height)

Parameters:

- bitmap: BITMAP: The bitmap to draw
- x: number: The x-coordinate on screen.
- y: number: The y-coordinate on screen.
- palette: The palette to use for drawing.
- x0: The smallest bitmap x coordinate to draw.
- y0: The smallest bitmap y coordinate to draw.
- width: Width of region to draw
- height: Height of region to draw.

Like bitmap:draw, but clip the bitmap area drawn.

4.6.5 Method draw_clip_outside: Draw a bitmap outside game are, with clipping

- Syntax: bitmap:draw_clip_outside(x, y, palette, x0, y0, width, height)

Like bitmap:draw_clip, but only draw outside game area.

4.6.6 Method pset: Set pixel in bitmap

- Syntax: bitmap:pset(x, y, color)
- Deprecaed: gui.bitmap_pset(bitmap, ...)

Parameters:

- bitmap: BITMAP: The bitmap to manipulate.
- x: number: x-coordinate of pixel to set.
- y: number: y-coordinate of pixel to set.
- color: number: The color index to set.

Sets specified pixel <x>,<y> in bitmap <bitmap> to color index <color>.

4.6.7 Method pget: Get pixel in bitmap

- Syntax: color bitmap:pget(x,y)
- Deprecated: color gui.bitmap_pget(bitmap, ...)

Parameters:

- bitmap: BITMAP: The bitmap to query.
- x: number: x-coordinate of pixel to get.
- y: number: y-coordinate of pixel to get.

Returns:

- color: number: The color index in specified pixel.

Gets color index of specified pixel <x>,<y> in bitmap <bitmap>.

4.6.8 Method size: Get size of bitmap

- Syntax: width, height bitmap:size()
- Syntax: width, height gui.bitmap_size(bitmap)

Parameters:

- bitmap: BITMAP: The bitmap to query.

Returns:

- width: number: The width of the bitmap.
- height: number: The height of the bitmap.

Get size of bitmap <bitmap>.

4.6.9 Method blit: Blit a bitmap into another

- Syntax: dest:blit(dx, dy, src, sx, sy, w, h, [ck])
- Deprecated: gui.bitmap_blit(dest, ...)

Parameters:

- dest: BITMAP: The target bitmap to blit to.
- dx: number: The x-coordinate in target.
- dy: number: The y-coordinate in target.
- src: BITMAP: The source bitmap.
- sx: number: The x-coordinate in source.
- sy: number: The y-coordinate in source.
- w: number: The width of area to blit.
- h: number: The height of area to blit.
- ck: number: The color key. Pixels with this index are not copied.
  - If none is specified, all pixels are copied.

Copy region of size <w>*<h> at <sx>,<sy> in bitmap <src> into <dest> at <dx>,<dy>. If a color key <ck> is specified, pixels of that color are not copied.

4.6.10 Method blit_scaled: Blit a bitmap into another with scaling

- Syntax: dest:blit_scaled(dx, dy, src, sx, sy, w, h, hscl, [vscl], [ck])
- Deprecated: gui.bitmap_blit_scaled(dest, ...)

Parameters:

- dest: BITMAP: The target bitmap to blit to.
- dx: number: The x-coordinate in target.
- dy: number: The y-coordinate in target.
- src: BITMAP: The source bitmap.
- sx: number: The x-coordinate in source.
- sy: number: The y-coordinate in source.
- w: number: The width of area to blit.
- h: number: The height of area to blit.
- hscl: number: Horizontal scale factor (integer).
- vscl: number: Vertical scale factor (integer). Defaults to the same as <hscl>.
- ck: number: The color key. Pixels with this index are not copied.
  - If none is specified, all pixels are copied.

Like BITMAP:blit, but also scales the copied part of bitmap (using nearest neighbor) by factor of <hscl>*<vscl>.

4.6.11 Method blit_porterduff: Blit a bitmap into another with Porter-Duff composition

- Syntax: dest:blit_porterduff(dx, dy, src, sx, sy, w, h, operator)
- Deprecated: gui.bitmap_blit_porterduff(dest, ...)

Parameters:

- dest: BITMAP: The target bitmap to blit to.
- dx: number: The x-coordinate in target.
- dy: number: The y-coordinate in target.
- src: BITMAP: The source bitmap.
- sx: number: The x-coordinate in source.
- sy: number: The y-coordinate in source.
- w: number: The width of area to blit.
- h: number: The height of area to blit.
- operator: string: The operator to use.
  - Can be one of: Src, Atop, Over, In, Out, Dest, DestAtop, DestOver, DestIn, DestOut, Clear or Xor.

Like BITMAP:blit, but instead of just overwriting, applies specified Porter-Duff operator.

- Color index 0 in source and target is treated as background.

4.6.12 Method blit_scaled_porterduff: Blit a bitmap into another with scaling and Porter-Duff composition

- Syntax: dest:blit_scaled_porterduff(dx, dy, src, sx, sy, w, h, hscl, [vscl], operator)
- Deprecated: gui.bitmap_blit_scaled_porterduff(dest, ...)

Parameters:

- dest: BITMAP: The target bitmap to blit to.
- dx: number: The x-coordinate in target.
- dy: number: The y-coordinate in target.
- src: BITMAP: The source bitmap.
- sx: number: The x-coordinate in source.
- sy: number: The y-coordinate in source.
- w: number: The width of area to blit.
- h: number: The height of area to blit.
- hscl: number: Horizontal scale factor (integer).
- vscl: number: Vertical scale factor (integer). Defaults to the same as <hscl>.
- operator: string: The operator to use.
  - Can be one of: Src, Atop, Over, In, Out, Dest, DestAtop, DestOver, DestIn, DestOut, Clear or Xor.

Like BITMAP:blit_porterduff, but also scales the source by <hscl>*<vscl> like BITMAP:blit_scaled.

4.6.13 Method blit_priority: Blit a bitmap into another with color priority

- Syntax: dest:blit_priority(dx, dy, src, sx, sy, w, h)
- Deprecated: gui.bitmap_blit_priority(dest, ...)

Parameters:

- dest: BITMAP: The target bitmap to blit to.
- dx: number: The x-coordinate in target.
- dy: number: The y-coordinate in target.
- src: BITMAP: The source bitmap.
- sx: number: The x-coordinate in source.
- sy: number: The y-coordinate in source.
- w: number: The width of area to blit.
- h: number: The height of area to blit.

Like BITMAP:blit, but only copy pixels where source color index is greater than destination color index.

4.6.14 Method blit_scaled_priority: Blit a bitmap into another with color priority and scaling

- Syntax: dest:blit_scaled_priority(dx, dy, src, sx, sy, w, h, hscl, [vscl])
- Deprecated: gui.bitmap_blit_scaled_priority(dest, ...)

Parameters:

- dest: BITMAP: The target bitmap to blit to.
- dx: number: The x-coordinate in target.
- dy: number: The y-coordinate in target.
- src: BITMAP: The source bitmap.
- sx: number: The x-coordinate in source.
- sy: number: The y-coordinate in source.
- w: number: The width of area to blit.
- h: number: The height of area to blit.
- hscl: number: Horizontal scale factor (integer).
- vscl: number: Vertical scale factor (integer). Defaults to the same as <hscl>.

Like BITMAP:blit_priority, but apply scaling by <hscl>*<vscl> like BITMAP:blit_scaled.

4.6.15 Method save_png: Save a bitmap to PNG

- Syntax: bitmap:save_png(filename, [base], palette)
- Syntax: data bitmap:save_png(palette)
- Deprecated: ... gui.bitmap_save_png(bitmap, ...)

Parameters:

- bitmap: BITMAP: The bitmap to save.
- filename: string: The filename to save to.
- base: string: The base filename is resolved relative to.
- palette: PALETTE: The palette to use.

Return value:

- data: string: BASE64 encoded PNG data.

Save bitmap <bitmap>, with palette <pal> into PNG file <filename> (relative to <base>) or return BASE64 encoding of it.

4.6.16 Method hash: Hash a bitmap

- Syntax: hash bitmap:hash()
- Deprecated: hash bitmap:hash(bitmap)

Parameters:

- bitmap: BITMAP: The bitmap to hash.

Return value:

- hash: string: 64-hex digit hash

Hashes bitmap <bitmap> and returns crypto-strong hash.

- Color order in bitmap is significant.

4.7 DBITMAP: Direct-color bitmap

4.7.1 Static function: new: Create a new bitmap

- Syntax: bitmap gui.dbitmap.new(w, h, [fillcolor])
- Syntax: bitmap classes.DBITMAP.new(w, h, [fillcolor])
- Deprecated: bitmap gui.bitmap_new (w, h, true, [fillcolor]).

Parameters:

- w: number: The width of new bitmap.
- h: number: The height of new bitmap.
- true: boolean: Fixed boolean true
- fillcolor: The color to fill the bitmap with (default transparent).

Return value:

- bitmap: DBITMAP: The new bitmap.

Create a new direct-color bitmap of size <w>*<h>, initially filled with <fillcolor>.

4.7.2 Method draw: Draw a bitmap

- Syntax: bitmap.draw(x, y)
- Deprecated: gui.bitmap_draw(x, y, bitmap)

Parameters:

- bitmap: DBITMAP: The bitmap to draw.
- x: number: X-coordinate on screen.
- y: number: Y-coordinate on screen.

Draw bitmap <bitmap> on screen at <x>,<y>.

4.7.3 Method draw_outside: Draw a bitmap outside game area

- Syntax: dbitmap:draw_outside(x, y, palette)

Like dbitmap:draw, but does not draw on game area.

4.7.4 Method draw_clip: Draw a bitmap, with clipping

- Syntax: dbitmap:draw(x, y, palette, x0, y0, width, height)

Parameters:

- bitmap: DBITMAP: The bitmap to draw
- x: number: The x-coordinate on screen.
- y: number: The y-coordinate on screen.
- x0: The smallest bitmap x coordinate to draw.
- y0: The smallest bitmap y coordinate to draw.
- width: Width of region to draw
- height: Height of region to draw.

Like dbitmap:draw, but clip the bitmap area drawn.

4.7.5 Method draw_clip_outside: Draw a bitmap outside game are, with clipping

- Syntax: dbitmap:draw_clip_outside(x, y, palette, x0, y0, width, height)

Like dbitmap:draw_clip, but only draw outside game area.

4.7.6 Method pset: Set pixel in bitmap

- Syntax: bitmap:pset(x, y, color)
- Deprecaed: gui.bitmap_pset(bitmap, ...)

Parameters:

- bitmap: DBITMAP: The bitmap to manipulate.
- x: number: x-coordinate of pixel to set.
- y: number: y-coordinate of pixel to set.
- color: number/string: The color to set.

Sets specified pixel <x>,<y> in bitmap <bitmap> to color <color>.

4.7.7 Method pget: Get pixel in bitmap

- Syntax: color bitmap:pget(x,y)
- Deprecated: color gui.bitmap_pget(bitmap, ...)

Parameters:

- bitmap: DBITMAP: The bitmap to query.
- x: number: x-coordinate of pixel to get.
- y: number: y-coordinate of pixel to get.

Returns:

- color: number: The color of specified pixel.

Gets color index of specified pixel <x>,<y> in bitmap <bitmap>.

4.7.8 Method size: Get size of bitmap

- Syntax: width, height bitmap:size()
- Syntax: width, height gui.bitmap_size(bitmap)

Parameters:

- bitmap: DBITMAP: The bitmap to query.

Returns:

- width: number: The width of the bitmap.
- height: number: The height of the bitmap.

Get size of bitmap <bitmap>.

4.7.9 Method blit: Blit a bitmap into another

- Syntax: dest:blit(dx, dy, src, sx, sy, w, h, [ck])
- Syntax: dest:blit(dx, dy, src, srcpal, sx, sy, w, h, [ck])
- Deprecated: gui.bitmap_blit(dest, ...)

Parameters:

- dest: DBITMAP: The target bitmap to blit to.
- dx: number: The x-coordinate in target.
- dy: number: The y-coordinate in target.
- src: BITMAP/DBITMAP: The source bitmap.
- srcpal: PALETTE: If <src> is indexed, this is the palette for source.
- sx: number: The x-coordinate in source.
- sy: number: The y-coordinate in source.
- w: number: The width of area to blit.
- h: number: The height of area to blit.
- ck: number: The color key. Pixels with this index are not copied.
  - If none is specified, all pixels are copied.
  - If <src> is paletted, this is color index, otherwise it is a color.

Copy region of size <w>*<h> at <sx>,<sy> in bitmap <src> (with palette <pal> if indexed) into <dest> at <dx>,<dy>. If a color key <ck> is specified, pixels of that color are not copied.

4.7.10 Method blit_scaled: Blit a bitmap into another with scaling

- Syntax: dest:blit_scaled(dx, dy, src, sx, sy, w, h, hscl, [vscl], [ck])
- Syntax: dest:blit_scaled(dx, dy, src, srcpal, sx, sy, w, h, hscl, [vscl], [ck])
- Deprecated: gui.bitmap_blit_scaled(dest, ...)

Parameters:

- dest: DBITMAP: The target bitmap to blit to.
- dx: number: The x-coordinate in target.
- dy: number: The y-coordinate in target.
- src: BITMAP/DBITMAP: The source bitmap.
- srcpal: PALETTE: If <src> is indexed, this is the palette for source.
- sx: number: The x-coordinate in source.
- sy: number: The y-coordinate in source.
- w: number: The width of area to blit.
- h: number: The height of area to blit.
- hscl: number: Horizontal scale factor (integer).
- vscl: number: Vertical scale factor (integer). Defaults to the same as <hscl>.
- ck: number: The color key. Pixels with this index are not copied.
  - If none is specified, all pixels are copied.
  - If <src> is paletted, this is color index, otherwise it is a color.

Like DBITMAP:blit, but also scales the copied part of bitmap (using nearest neighbor) by factor of <hscl>*<vscl>.

4.7.11 Method blit_porterduff: Blit a bitmap into another with Porter-Duff composition

- Syntax: dest:blit_porterduff(dx, dy, src, sx, sy, w, h, operator)
- Syntax: dest:blit_porterduff(dx, dy, src, srcpal, sx, sy, w, h, operator)
- Deprecated: gui.bitmap_blit_porterduff(dest, ...)

Parameters:

- dest: DBITMAP: The target bitmap to blit to.
- dx: number: The x-coordinate in target.
- dy: number: The y-coordinate in target.
- src: BITMAP/DBITMAP: The source bitmap.
- srcpal: PALETTE: If <src> is indexed, this is the palette for source.
- sx: number: The x-coordinate in source.
- sy: number: The y-coordinate in source.
- w: number: The width of area to blit.
- h: number: The height of area to blit.
- operator: string: The operator to use.
  - Can be one of: Src, Atop, Over, In, Out, Dest, DestAtop, DestOver, DestIn, DestOut, Clear or Xor.

Like DBITMAP:blit, but instead of just overwriting, applies specified Porter-Duff operator.

- In target, fully transparent is background. In source, either fully transparent (if direct) or index 0 (if paletted) is treated as background.

4.7.12 Method blit_scaled_porterduff: Blit a bitmap into another with scaling and Porter-Duff composition

- Syntax: dest:blit_scaled_porterduff(dx, dy, src, sx, sy, w, h, hscl, [vscl], operator)
- Syntax: dest:blit_scaled_porterduff(dx, dy, src, srcpal, sx, sy, w, h, hscl, [vscl], operator)
- Deprecated: gui.bitmap_blit_scaled_porterduff(dest, ...)

Parameters:

- dest: DBITMAP: The target bitmap to blit to.
- dx: number: The x-coordinate in target.
- dy: number: The y-coordinate in target.
- src: BITMAP/DBITMAP: The source bitmap.
- srcpal: PALETTE: If <src> is indexed, this is the palette for source.
- sx: number: The x-coordinate in source.
- sy: number: The y-coordinate in source.
- w: number: The width of area to blit.
- h: number: The height of area to blit.
- hscl: number: Horizontal scale factor (integer).
- vscl: number: Vertical scale factor (integer). Defaults to the same as <hscl>.
- operator: string: The operator to use.
  - Can be one of: Src, Atop, Over, In, Out, Dest, DestAtop, DestOver, DestIn, DestOut, Clear or Xor.

Like DBITMAP:blit_porterduff, but also scales the source by <hscl>*<vscl> like DBITMAP:blit_scaled.

4.7.13 Method adjust_transparency: Adjust transparency of bitmap

- Syntax: bitmap:adjust_transparency(newvalue)
- Deprecated: gui.adjust_transparency(bitmap, ...)

Parameters:

- bitmap: DBITMAP: The bitmap to adjust.
- Number adj: The factor to multiply opaqueness with times 256.

Multiply opaqueness of all colors in bitmap <bitmap> by factor of <adj>/256.

4.7.14 Method save_png: Save a bitmap to PNG

- Syntax: bitmap:save_png(filename, [base])
- Syntax: data bitmap:save_png()
- Deprecated: ... gui.bitmap_save_png(bitmap, ...)

Parameters:

- bitmap: DBITMAP: The bitmap to save.
- filename: string: The filename to save to.
- base: string: The base filename is resolved relative to.

Return value:

- data: string: BASE64 encoded PNG data.

Save bitmap <bitmap> into PNG file <filename> (relative to <base>) or return BASE64 encoding of it.

4.7.15 Method hash: Hash a bitmap

- Syntax: hash bitmap:hash()
- Deprecated: hash bitmap:hash(bitmap)

Parameters:

- bitmap: DBITMAP: The bitmap to hash.

Return value:

- hash: string: 64-hex digit hash

Hashes bitmap <bitmap> and returns crypto-strong hash.

4.8 IMAGELOADER: Load an image

4.8.1 Static function load: Load a bitmap from file

- Syntax: bitmap, palette gui.image.load(file, [base])
- Syntax: bitmap, palette classes.IMAGELOADER.load(file, [base])
- Deprecated: ... gui.bitmap_load(...)

Parameters:

- file: string: The file to read.
- base: string: The base to resolve <file> relative to.

Returns:

- bitmap: BITMAP/DBITMAP: The new bitmap
- palette: PALETTE: If bitmap is paletted, this is the palette, otherwise nil.

Load a new bitmap and return it (plus associated palette if any).

4.8.2 Static function load_str: Load a bitmap from string

- Syntax: bitmap, palette gui.image.load_str(data)
- Syntax: bitmap, palette classes.IMAGELOADER.load_str(data)
- Deprecated: ... gui.bitmap_load_str(...)

Parameters:

- data: string: The image data

Returns:

- bitmap: BITMAP/DBITMAP: The new bitmap
- palette: PALETTE: If bitmap is paletted, this is the palette, otherwise nil.

Like IMAGELOADER:load, but read the bitmap from string <data> instead of file.

4.8.3 Static function load_png: Load a bitmap from PNG file

- Syntax: bitmap, palette gui.image.load_png(file, [base])
- Syntax: bitmap, palette classes.IMAGELOADER.load_png(file, [base])
- Deprecated: ... gui.bitmap_load_png(...)

Parameters:

- file: string: The file to read.
- base: string: The base to resolve <file> relative to.

Returns:

- bitmap: BITMAP/DBITMAP: The new bitmap
- palette: PALETTE: If bitmap is paletted, this is the palette, otherwise nil.

Like IMAGELOADER:load, but load a PNG file instead.

4.8.4 Static function load_png_str: Load a PNG bitmap from string

- Syntax: bitmap, palette gui.image.load_png_str(data)
- Syntax: bitmap, palette classes.IMAGELOADER.load_png_str(data)
- Deprecated: ... gui.bitmap_load_png_str(...)

Parameters:

- data: string: The image data, base64 encoded.

Returns:

- bitmap: BITMAP/DBITMAP: The new bitmap
- palette: PALETTE: If bitmap is paletted, this is the palette, otherwise nil.

Like IMAGELOADER:load_png, but read the bitmap from BASE64 encoded string <data> instead of file.

4.9 CUSTOMFONT: Arbitrary-sized bitmap font

4.9.1 Static function new: Return a new empty font

- Syntax: font gui.font.new()
- Syntax: font classes.CUSTOMFONT.new()
- Deprecated: font gui.font_new()

Return value:

- font: CUSTOMFONT: New font.

Create a new font with no characters and return it.

4.9.2 Static function: load: Load a font file

- Syntax: font gui.font.load(file, [base])
- Syntax: font gui.font.load()
- Syntax: font classes.CUSTOMFONT.load(file, [base])
- Syntax: font classes.CUSTOMFONT.load()
- Deprecated: font gui.loadfont(...)

Parameters:

- file: string: The file to read the font from
- base: string: The file to resolve <file> relative to.

Return value:

- font: CUSTOMFONT: New font.

Load font from file <file> (relative to <base>). If no filename is given, system default font is loaded.

4.9.3 operator(): Render text to screen

- Syntax: font(x, y, text, [fgc], [bgc], [hlc])

Parmeters:

- font: CUSTOMFONT: The font to use.
- x: number: The x-position on screen to draw to.
- y: number: The y-position on screen to draw to.
- text: string: The text to draw.
- fgc: number/string: Foreground color (default white).
- bgc: number/string: Background color (default transparent).
- hlc: number/string: Outline color (default transparent).

Draws a string <text> with specified font <font> and colors <fgc>,<bgc>,<hlc> onto screen.

- If <hlc> is transparent, no outline is drawn.

4.9.4 Method edit: Alter glyph in font

- Syntax: font:edit(character, glyph)

Parameters:

- font: CUSTOMFONT: The font to edit.
- character: string: The character to edit (UTF-8 encoded).
- glyph: BITMAP: The bitmap to use.

Replace character <character> in font <font> by <glyph>.

- Color index 0 is background, everything else is foreground.
- <character> may be empty string, meaning the replacement character used for bad characters.
- <character> may be multi-codepoint string, meaning character used for that ligature.

4.10 ICONV: Character set conversions

4.10.1 Static function new: Create new character set converter

- Syntax: iconv iconv.new(from, to);
- Syntax: iconv classes.ICONV.new(from, to);
- Deprecated: iconv iconv_new(...);

Parameters:

- from: string: The source character set.
- to: string: The target character set.

Returns:

- iconv: ICONV: The converter.

Create a character set converter, converting from <from> to <to> and return it.

4.10.2 Operator(): Convert string fragment from character set to another

- Syntax: success, result, unconverted, error iconv(input)

Parameters:

- iconv: ICONV: The context to use.
- input: string: The input to convert.

Return value:

- success: boolean: True if conversion was successful, false if not.
- result: string: The string (partially) converted.
- unconvered: number: Number of bytes that were not converted (only if <success> is false).
- error: string: Error that caused conversion to stop (only if <success> is false).
  - INVALID: The input string is invalid.
  - INCOMPLETE: The input string cuts off in middle of character.
  - INTERNALERR: Internal error.

Convert a string <input> using character set converter <iconv> and return the result.

4.11 FILEREADER: Read a file as a stream

4.11.1 Static function open: Open a stream

- Syntax: handle filereader.open(file, [base])
- Syntax: handle classes.FILEREADER.open(file, [base])
- Deprecated: handle open_file(file, [base])

Parameters:

- file: string: The filename to read.
- base: string: The base <file> is resolved against.

Returns:

- handle: FILEREADER: The new file reader.

Open file <file> (relative to <base>) and return a handle to it.

4.11.2 operator(): Read line/bytes from stream

- Syntax: result handle()
- Syntax: result handle(bytes)

Parameters:

- handle: FILEREADER: The handle to read from.
- bytes: Number of bytes to read (default is next line).

Returns:

- result: string: The read data, or nil on end-of-file.

Reads next line or <bytes> bytes from specified file handle <handle>.

- If reading specified number of bytes, the bytes are read in binary mode (as-is).
- If reading next line, the line is read in text mode (any line ending is skpped).

4.11.3 Method lines: Iterator to read all lines

- Syntax: for line in handle:lines() do ... end

Parameters:

- handle: FILEREADER: The handle to read.

Returns:

- A lua iterator with one variable.

Return a Lua iterator that iterates all the lines in <handle>.

4.12 COMPARE_OBJ: Watch memory area for changes

Objects of this class allow fast checking for modifications to given memory block.

4.12.1 Static function new: Create a checker

- Syntax: handle classes.COMPARE_OBJ.new({marea, offset|addrobj}, size, [rows, stride])
- Syntax: handle memory.compare_new({marea, offset|addrobj}, size, rows, stride)

Parameters:

- marea: string: The memory area to interpret <offset> against.
- offset: number: The initial offset in memory area.
- addrobj: ADDRESS: The memory address.
- size: number: The number of bytes in each row.
- rows: number: The number of rows. Default is 1.
- stride: number: The number of bytes offset increments from one row to next.

Returns:

- A handle to object.

Return an object watching specified memory area.

- Note: For fastest operation, limit checks to inside one memory area (that has to be mappable, individual RAM areas often are).

4.12.2 operator(): Check area for modifications

- Syntax: boolean handle()

Returns:

- True if memory block has been modified since last call (or object creation if first call), false if not.

Check if the block has been modified.

4.13 ADDRESS: Memory address

Objects of this class contain a memory address.

4.13.1 Static function new: Create new memory address

- Syntax: ADDRESS classes.ADDRESS.new(string marea, number offset)
- Syntax: ADDRESS memory.address.new(string marea, number offset)
- Syntax: ADDRESS memory.mkaddr(string marea, number offset)

Parameters:

- marea: The memory area
- offset: The offset in memory area.

Returns:

- The memory area.

Construct a new memory address object.

4.13.2 Method: addr: Get global address

- Syntax: number addr:addr()

Parameters:

- addr: ADRESS: The original address.

Returns:

- The global address corresponding to this address.

4.13.3 Method: vma: Get memory area

- Syntax: string addr:vma()

Parameters:

- addr: ADRESS: The original address.

Returns:

- The memory area corresponding to this address.

4.13.4 Method: offset: Get memory area offset

- Syntax: string addr:offset()

Parameters:

- addr: ADRESS: The original address.

Returns:

- The offset in memory area corresponding to this address.

4.13.5 Method: replace: Replace address part

- Syntax: ADDRESS addr:replace(offset[, bits])

Parameters:

- addr: ADRESS: The original address.
- offset: number: The new offset
- bits: number: The number of LSB to replace.

Returns:

- The modified address

Returns a new address, with <bits> (all if missing) least significant bits of <addr> replaced by LSB of <offset>.

4.13.6 Method: add: Add to address

- Syntax: ADDRESS addr:add(offset)
- Syntax: ADDRESS addr:add(number, stride)
- Syntax: ADDRESS addr:add(number, stride, offset)

Parameters:

- addr: ADRESS: The original address.
- offset: number: Offset to add.
- number: number: Number of table strides to add.
- stride: number: The table stride.

Returns:

- The modified address

Adds <offset>, <number>*<stride> or <number>*<stride>+<offset> into specified address <addr>.

4.14 ADDRESS:<op>: Read/Write memory

- Syntax: none addr:<op>(value)
- Syntax: number addr:<op>()

Parameters:

- addr: ADDRESS: The address to read/write.
- value: number: The number to write.

Returns:

- The value read.

Read/Write value from/to given address <addr>. The value written is <value>. <Op> is of form: [i][s]<type>, where:

- <type> is one of 'byte', 'word', 'hword', 'dword', 'qword', 'float', 'double'.
- 'i' signifies that the value is treated as opposite-to-normal endianess,
- 's' signifies that value is treated as signed (not available for floating-point).

5 Global

5.1 print: Print values to console

- Syntax: none print(value... values)

Prints specified values to console. Can print any Lua type at least enough to identify the type and instance.

5.2 tostringx: Format a value to string

- Syntax: string tostringx(value val)

Formats value <val> like print would, and returns the result as a string.

5.3 exec: Execute lsnes commands

- Syntax: none exec(string cmd)

Execute lsnes command <cmd>.

5.4 utime: Get current time

- Syntax: (number,number) utime()

Returns two numbers. First is time since some epoch in seconds, the second is microseconds mod 10^6 since that epoch.

5.5 set_idle_timeout: Run function after timeout when emulator is idle

- Syntax: none set_idle_timeout(number timeout)

Set number of microseconds to block idle for. After this timeout has expired, on_idle() will be called once.

5.6 set_timer_timeout: Run function after timeout.

- Syntax: none set_timer_timeout(number timeout)

Set number of microseconds to block timer for. After this timeout has expired, on_timer() will be called once.

5.7 bus_address: Look up address in system bus.

- Syntax: none bus_address(number bus_addr)

Returns virtual address corresponding to specified address on system bus.

5.8 loopwrapper: Convert loop into callable function

- Syntax: function loopwrapper(function fun, ...)

Calls function <fun> with function and specified arguments. The function passed suspends execution until the function returned is called. Handy for linear flow control among multiple invocations of a hook. Example code:

```
on_paint = loopwrapper(function(wait)
    while true do
        gui.text(0, 0, “Test!”);
        wait();
    end
end);
```

5.9 list_bindings: List keybindings

- Syntax: table list_bindings([string cmd])

Get table of all keybindings, indexed by keyspec (modifiers|mask/key). If <cmd> is specified, the table is limited to that command. Also searches for controller keys.

5.10 get_alias: Get expansion of alias

- Syntax: string get_alias(string aname)

Get expansion of given alias <aname>.

5.11 set_alias: Set expansion of alias

- Syntax: none set_alias(string aname, string value)

Set expansion of given alias.

5.12 create_ibind: Create invese binding

- Syntax: INVERSEBIND create_ibind(string name, string cmd)

Return object representing inverse binding with specified name <name> and specified command <cmd>.

- Note: To create press/release commands, use aliases +foo and -foo .
- Note: Keep the returned object around.

5.13 create_command: Create a command

- Syntax: COMMANDBIND create_command(string name, function a)
- Syntax: COMMANDBIND create_command(string name, function a, function b)

Return object representing a command (pair).

- If only one function is specied, the command is level-sensitive, <a> is callback.
- If <b> is function, the function is edge-sensitive, <a> is positive edge callback and <b> is negative edge callback.
- All callbacks get single argument: The parameters passed.
- Keep the returned object around.

5.14 loadfile: Load Lua script

- Syntax: function loadfile(string filename[, string base])

Load lua script from <filename>, resolved relative to <base> (if empty, current directory).

5.15 dofile: Execute Lua script

- Syntax: function dofile(string filename[, string base])

Execute lua script from <filename>, resolved relative to <base> (if empty, current directory) and return all return values.

5.16 resolve_filename: Resolve name of file relative to another

- Syntax: string resolve_filename(string filename[, string base])

Resolve name of file <filename> relative to <base> and return the result.

5.17 render_queue_function: Return paint function for render queue

- Syntax: function render_queue_function(RENDERQUEUE rq)

Return function that renders render queue <rq>.

- Handy for paint callback if one is using render queues updated in other callbacks. As in:

```
handle = callback.paint:register(render_queue_function(my_rq));
```

5.18 identify_class: Identify class of object

- Syntax: string identify_class(userdata object)

Identifies the class of userdata <object>, if possible. If no identification is possible, returns "unknown".

5.19 lookup_class: Lookup class by name

- Syntax: classobj lookup_class(string name)

Looks up class corresponding to <name>, if possible. If not found, returns nil. The classobj has following fields:

- _static_methods: Return static method names
- _class_methods: Return class method names
- <static-function-name>: The specified static function.

5.20 all_classes: Get list of all classes

- Syntax: string... all_classes()

Get names of all classes available.

5.21

5.22 icnov: Class ICONV

See class ICONV.

5.23 filereader: Class FILEREADER

See class FILEREADER.

6 Table bit:

Bitwise logical functions and related.

6.1 bit.none/bit.bnot: Bitwise none or NOT function

- Syntax: number bit.none(number...)
- Syntax: number bit.bnot(number...)

48-bit bitwise NOT / NONE function (set bits that are set in none of the arguments).

6.2 bit.any/bit.bor: Bitwise any or OR function

- Syntax: number bit.any(number...)
- Syntax: number bit.bor(number...)

48-bit bitwise OR / ANY function (set bits that are set in any of the arguments).

6.3 bit.all/bit.band: Bitwise all or AND function

- Syntax: number bit.all(number...)
- Syntax: number bit.band(number...)

48-bit bitwise AND / ALL function (set bits that are set in all of the arguments).

6.4 bit.parity/bit.bxor: Bitwise parity or XOR function

- Syntax: number bit.parity(number...)
- Syntax: number bit.bxor(number...)

48-bit bitwise XOR / PARITY function (set bits that are set in odd number of the arguments).

6.5 bit.lrotate: Rotate a number left

- Syntax: number bit.lrotate(number base[, number amount[, number bits]])

Rotate <bits>-bit (max 48, default 48) number <base> left by <amount> (default 1) places.

6.6 bit.rrotate: Rotate a number right

- Syntax: number bit.rrotate(number base[, number amount[, number bits]])

Rotate <bits>-bit (max 48, default 48) number <base> right by <amount> (default 1) places.

6.7 bit.lshift: Shift a number left

- Syntax: number bit.lshift(number base[, number amount[, number bits]])

Shift <bits>-bit (max 48, default 48) number <base> left by <amount> (default 1) places. The new bits are filled with zeroes.

6.8 bit.lrshift: Shift a number right (logical)

- Syntax: number bit.lrshift(number base[, number amount[, number bits]])

Shift <bits>-bit (max 48, default 48) number <base> logically right by <amount> (default 1) places. The new bits are filled with zeroes.

6.9 bit.arshift: Shift a number right (arithmetic)

- Syntax: number bit.arshift(number base[, number amount[, number bits]])

Shift <bits>-bit (max 48, default 48) number <base> logically right by <amount> (default 1) places. The new bits are shifted in with copy of the high bit.

6.10 bit.extract: Extract/shuffle bits from number

- Syntax: number bit.extract(number base[, number bit0[, number bit1,...]])

Returns number that has bit0-th bit as bit 0, bit1-th bit as 1 and so on.

- Note: Bit numbers up to 51 should work reliably (then things start falling apart due to double precision issues).
- Note: There are two special bit positions, true and false, standing for always set bit and always clear bit.

6.11 bit.value: Construct number with specified bits set

- Syntax: number bit.value([number bit1[, number bit2,...]])

Returns bitwise OR of 1 left shifted by <bit1> places, 1 left shifted by <bit2> places and so on. As special value, nil argument is no-op.

6.12 bit.test: Test if bit is set

- Syntax: boolean bit.test(number a, number bit)

Tests if bit <bit> is set in <a>. If it is set, returns true, otherwise false.

6.13 bit.testn: Test if bit is clear

- Syntax: boolean bit.testn(number a, number bit)

Tests if bit <bit> is set in <a>. If it is clear, returns true, otherwise false.

6.14 bit.test_any: Test if any bit is set

- Syntax: boolean bit.test_any(number a, number b)

Returns true if bitwise and of <a> and <b> is nonzero, otherwise false.

6.15 bit.test_all: Test if all bits are set

- Syntax: boolean bit.test_all(number a, number b)

Returns true if bitwise and of <a> and <b> equals <b>, otherwise false.

6.16 bit.popcount: Population count

- Syntax: number bit.popcount(number a)

Returns number of set bits in <a>.

6.17 bit.clshift: Chained left shift

- Syntax: (number, number) bit.clshift(number a, number b, [number amount,[number bits]])

Does chained left shift on <a>, <b> by <amount> positions (default 1), assuming numbers to be of specified number of bits <bits> (default 48).

6.18 bit.crshift: Chained right shift

- Syntax: (number, number) bit.crshift(number a, number b, [number amount,[number bits]])

Does chained right shift on <a>, <b> by <amount> positions (default 1), assuming numbers to be of specified number of bits <bits> (default 48).

6.19 bit.flagdecode: Decode bitfield into flags

- Syntax: string bit.flagdecode(number a, number bits, [string on, [string off]])

Return string of length bits where ith character is ith character of on if bit i is on, otherwise ith character of off. Out of range reads give last character.

- Note: <on> defaults to '*' if empty.
- Note: <off> defaults to '-' if empty.

6.20 bit.rflagdecode: Decode bitfield into flags

- Syntax: string bit.rflagdecode(number a, number bits, [string on, [string off]])

Like bit.flagdecode, but outputs the string in the opposite order (most significant bit first).

6.21 bit.swap{,s}{,h,d,q}word: Swap word endian

- Syntax: number bit.swapword(number n)
- Syntax: number bit.swaphword(number n)
- Syntax: number bit.swapdword(number n)
- Syntax: number bit.swapqword(number n)
- Syntax: number bit.swapsword(number n)
- Syntax: number bit.swapshword(number n)
- Syntax: number bit.swapsdword(number n)
- Syntax: number bit.swapsqword(number n)

Swap endianess of (un)signed integer <n>.

6.22 bit.compose: Compose multi-byte number

- Syntax: number bit.compose(number n...)

Return $n_1 + 256n_2 + 256^2n_3 + \ldots$

6.23 bit.binary_ld_{{u,s}{8,16,24,32,64},float,double}{l,b}e: Load binary integer

- Syntax: number bit.binary_ld_<type>le(string str, number pos);
- Syntax: number bit.binary_ld_<type>be(string str, number pos);

Load little (*le) or big (*be) endian binary number from position <pos> of string <str>. Type may be one of: u8, u16, u24, u32, u64, s8, s16, s24, s32, s64, float, double.

6.24 bit.binary_st_{{u,s}{8,16,24,32,64},float,double}{l,b}e: Store binary integer

- Syntax: string bit.binary_st_<type>le(number x);
- Syntax: string bit.binary_st_<type>be(number x);

Store specified number <x> as binary in string and return the result. Type may be one of: u8, u16, u24, u32, u64, s8, s16, s24, s32, s64, float, double.

6.25 bit.quotent: Integer quotent

- Syntax: number bit.quotent(number a, number b)

Calculate quotent a/b.

6.26 bit.multidiv: Divide and split among multiple divisiors

- Syntax: number... bit.multidiv(number v, number q...)

Does the following steps:

1. Set v' to <v>.
2. For each <q> q:
   (a) Calculate quotent(v'/q) and add that to numbers returned.
   (b) v' <- remainder(v'/q)
3. Add v' to numbers returned.

That is, it successively divides <v> by <q>s, and reduces <v> modulo <q> at each step. <v> may be floating point, <q>s are integers.

- E.g. bit.multidiv(time, 3600, 60) splits time into hours, minutes and seconds.
- E.g. bit.multidiv(a, b) calculates quotent and remainder of a/b.

6.27 bit.mul32: 32-bit multiply

- Syntax: number, number bit.mul32(number a, number b)

Multiply 32-bit numbers <a> and <b>. The first return value is low 32 bits of result, the second is high 32 bits.

7 Table classes:

7.1 classes.<foo>: The classobj for class <foo>

- Syntax: classes.<foo>

The classobj for class <foo>.

7.2 classes.<foo>._static_methods: Enumerate static methods

- Syntax: string... classes.<foo>._static_methods()

Returns all static methods of <foo> as strings.

7.3 classes.<foo>._class_methods: Enumerate static methods

- Syntax: string... classes.<foo>._class_methods()

Returns all class methods of <foo> as strings.

7.4 classes.<foo>.<bar>: Static method

- Syntax: variable classes.<foo>.<bar>(variable...)

Invokes static method <bar> of class <foo>.

8 Table gui:

8.1 gui.resolution: Get current resolution

- Syntax: (number, number) gui.resolution()

Returns 2-tuple (hresolution, vresolution).

8.2 gui.left_gap/gui.right_gap/gui.top_gap/gui.bottom_gap: Set edge gaps

- Syntax: number gui.left_gap(number gap)
- Syntax: number gui.right_gap(number gap)
- Syntax: number gui.top_gap(number gap)
- Syntax: number gui.bottom_gap(number gap)

Set the specified edge gap to specified value <gap> (max gap is 8191). If successful, old gap is returned.

8.3 gui.delta_left_gap/gui.delta_right_gap/gui.delta_top_gap/gui.delta_bottom_gap: Adjust edge gaps

- Syntax: number gui.delta_left_gap(number dgap)
- Syntax: number gui.delta_right_gap(number dgap)
- Syntax: number gui.delta_top_gap(number dgap)
- Syntax: number gui.delta_bottom_gap(number dgap)

Increase the specified edge gap by specified value <dgap> (max gap is 8191) and return the old gap (returns nothing on error).

8.4 gui.text/gui.textH/gui.textV,gui.textHV: Draw text

- Syntax: none gui.text(number x, number y, string text[, number fgc[, number bgc]])
- Syntax: none gui.textH(number x, number y, string text[, number fgc[, number bgc]])
- Syntax: none gui.textV(number x, number y, string text[, number fgc[, number bgc]])
- Syntax: none gui.textHV(number x, number y, string text[, number fgc[, number bgc]])

Draw specified text on the GUI (each character cell is 8 or 16 wide and 16 high). Parameters:

- x: X-coordinate to start the drawing from (and x-coordinate at begining of the lines).
- y: Y-coordinate to start the drawing from.
- text: The text to draw.
- fgc: Text color (default is 0xFFFFFF (white))
- bgc: Background color (default is -1 (transparent))

Note: The H variants draw at double width and V variants draw at double height.

8.5 gui.rectangle: Draw a rectangle

- Syntax: none gui.rectangle(number x, number y, number width, number height[, number thickness[, number outline[, number fill]]])

Draw rectangle on the GUI. Parameters:

- x: X-coordinate of left edge.
- y: Y-coordinate of upper edge.
- width: Width of rectangle.
- height: Height of rectangle.
- thickness: Thickness of outline (default is 1).
- outline: Color of outline (default is 0xFFFFFF (white))
- fill: Color of fill (default is -1 (transparent))

8.6 gui.solidrectangle: Draw a solid rectangle

- Syntax: none gui.rectangle(number x, number y, number width, number height[, number color])

Draw solid rectangle on the GUI. Parameters:

- x: X-coordinate of left edge.
- y: Y-coordinate of upper edge.
- width: Width of rectangle.
- height: Height of rectangle.
- color: Color of rectangle (default is 0xFFFFFF (white))

8.7 gui.box: Draw a 3D-effect box

- Syntax: none gui.box(number x, number y, number width, number height[, number thickness[, number outline1[,number outline2[, number fill]]]])

Draw rectangle with 3D effect on the GUI. Parameters:

- x: X-coordinate of left edge.
- y: Y-coordinate of upper edge.
- width: Width of rectangle.
- height: Height of rectangle.
- thickness: Thickness of outline (default is 1).
- outline1: First color of outline (default is 0xFFFFFF (white))
- outline2: First color of outline (default is 0x808080 (dark gray))
- fill: Color of fill (default is 0xC0C0C0 (light grayy))

8.8 gui.pixel: Draw a single pixel

- Syntax: none gui.pixel(number x, number y[, number color])

Draw one pixel on the GUI. Parameters:

- x: X-coordinate of the pixel
- y: Y-coordinate of the pixel
- color: Color of the pixel (default is 0xFFFFFF (white))

8.9 gui.crosshair: Draw a crosshair

- Syntax: none gui.crosshair(number x, number y[, number length[, number color]])

Draw a crosshair. Parameters:

- x: X-coordinate of the crosshair
- y: Y-coordinate of the crosshair
- length: Length of the crosshair lines (default 10).
- color: Color of the crosshair (default is 0xFFFFFF (white))

8.10 gui.line: Draw a line

- Syntax: none gui.line(number x1, number y1, number x2, number y2[, number color])

Draw a thin line. Parameters:

- x1: X-coordinate of one end.
- y1: Y-coordinate of one end.
- x2: X-coordinate of the other end.
- y2: Y-coordinate of the other end.
- color: Color of the line (default is 0xFFFFFF (white)).

8.11 gui.circle: Draw a (filled) circle

- Syntax: none gui.circle(number x, number y, number r[, number thick[, number border[, number fil]]])

Draw a circle. Parameters.

- x: X-coordinate of the center
- y: Y-coordinate of the center
- r: The radius of the circle
- thick: Border thickness
- border: Border color (default is 0xFFFFFF (white))
- fill: Fill color (default is -1 (transparent)).

8.12 gui.repaint: Arrange a repaint

- Syntax: none gui.repaint()

Request on_repaint() to happen as soon as possible.

8.13 gui.subframe_update: Enable/Disable subframe updates

- Syntax: none gui.subframe_update(boolean on)

Request subframe updates (calling on_paint() on subframes) to happen (<on>=true) or not happen (<on>=false).

8.14 gui.screenshot: Write a screenshot

- Syntax: none gui.screenshot(string filename)

Write PNG screenshot of the current frame (no drawings) to specified file <filename>.

8.15 gui.screenshot_bitmap: Write a screenshot to bitmap

- Syntax: DBITMAP gui.screenshot_bitmap()

Write PNG screenshot of the current frame (no drawings) to dbitmap and return the result.

8.16 gui.color: Compose a color.

- Syntax: number gui.color(number r, number g, number b[, number a])
- Syntax: number gui.color(string c)

Returns color (in notation Lua scripts use) corresponding to color (<r>,<g>,<b>), each component in scale 0-255. If <a> is specified, that is alpha (0 is fully transparent, 256(sic) is fully opaque). The default alpha is 256.

The form taking a string returns color corresponding color name.

8.17 gui.status: Set status variable

- Syntax: none gui.status(string name, string value)

Set status field "L[<name>]" to <value> in status area.

8.18 gui.rainbow: Rainbow color calculation

- Syntax: number gui.rainbow(number step, number steps[, number color])

Perform hue rotation of color <color> (default bright red), by <step> steps. The number of steps per full rotation is given by absolute value of <steps>.

If <step> is negative, the rotation will be counterclockwise.

8.19 gui.kill_frame: Kill video frame and associated sound

- Syntax: none gui.kill_frame()

Kills the currently dumped video frame + the associated sound. Only valid in on_video callback.

8.20 gui.set_video_scale: Set video frame scale

- Syntax: none gui.set_video_scale(number h, number v)

Sets the scale factors of current frame to <h>x<v>. Only valid in on_video callback.

8.21 gui.arrow: Draw an arrow

- Syntax: none gui.arrow(number x, number y, number length, number hwidth, number direction[, bool fill[, number color[, number twidth[, number hthick]]]])

Draws an arrow using color <color>. The tip of arrow is at (<x>, <y>). Other parameters:

1. <length>: The length of arrow tail.
2. <hwidth>: The width of arrow head. Should be odd.
3. <direction>: Direction of arrow. 0 is to right, +1 rotates 45 degrees counterclockwise.
4. <fill>: If true, fill the arrow head. Default false.
5. <twidth>: Tail width. Should be odd. Default 1.
6. <hthick>: Head thickness (only used if <fill> is false). Default is <twidth>.

8.22 gui.tiled_bitmap: Class TILEMAP

See class TILEMAP.

8.23 gui.palette: Class PALETTE

See class PALETTE.

8.24 gui.bitmap: Class BITMAP

See class BITMAP.

8.25 gui.dbitmap: Class DBITMAP

See class DBITMAP.

8.26 gui.font: Class CUSTOMFONT

See class CUSTOMFONT.

8.27 gui.renderctx: Class RENDERCTX

See class RENDERCTX.

8.28 gui.image: Class IMAGELOADER

See class IMAGELOADER.

8.29 gui.get_runmode: Get current emulator mode

- Syntax: string gui.get_runmode()

Gets the current emulator runmode. The possible ones are:

- quit: Emulator is quitting.
- normal: Emulator is running emulation
- load: Emulator is loading a movie/savestate
- advance_frame: Emulator is doing frame advance
- advance_subframe: Emulator is doing subframe advance
- skiplag: Emulator is skipping lag frames
- skiplag_pending: Emulator will start skipping lag frames next frame
- pause: Emulator is paused
- pause_break: Emulator is paused at breakpoint
- corrupt: Emulator can't run because corrupt emulation state
- unknown: Unknown state (should not happen).

9 table input

Input handling. Functions manipulating input are only available in on_input callback.

9.1 input.get: Read controller button/axis (deprecated)

- Syntax: number input.get(number controller, number index)

Read the specified index <index> (zero-based) from specified controller <controller> (zero-based).

9.2 input.set: Write controller button/axis (deprecated)

- Syntax: none input.set(number controller, number index, number value)

Write the specified index <index> (zero-based) from specified controller <controller> (zero-based), storing value <value>.

9.3 input.get2: Read controller button/axis

- Syntax: number input.get2(number port, number controller, number index)

Read the specified input tuple. Port 0 is system port.

9.4 input.set2: Write controller button/axis

- Syntax: input.set2(number port, number controller, number index, number value)

Write the specified input tuple. Port 0 is system port.

9.5 input.lcid_to_pcid2: Look up logical controller

- Syntax: (number, number) input.lcid_to_pcid2(number lcid)

Look up physical pcid pair (port, controller) corresponding to specified logical controller (1-based). Returns nothing if controller does not exist.

9.6 input.port_type: Look up port type

- Syntax: string input.port_type(number port)

Return type of specified port.

9.7 input.controller_info: Get information about controller

- Syntax: table input.controller_info(number port, number controller)

Get controller info for specified controller. If controller does not exist, returns nil. Otherwise returns a table with following fields:

- type (string): Type of the controller.
- class (string): Class of the controller.
- classnum (number): Number of the controller within its class (1-based)
- lcid (number): Logical controller number of the controller.
- button_count (number): Number of buttons on controller
- buttons (array): Array of following info about each button:
  - type (string): Type of button. Currently one of “null”, “button”, “axis”, “raxis”.
  - name (string): Name of button.
  - symbol (string): Symbol of button. Only present for type “button”.
  - hidden (boolean): True if hidden button.

9.8 input.veto_button: Veto a button press

- Syntax: none input.veto_button()

Signals that the button event should be vetoed. Only valid in on_button callback.

9.9 input.geta: Get all buttons for controller (deprecated)

- Syntax: (number, number...) input.geta(number controller)

Get input state for entiere controller. Returns n return values.

- 1st return value: Bitmask: bit i is set if i:th index is nonzero
- 2nd- return value: value of i:th index.

9.10 input.seta: Set all buttons for controller (deprecated)

- Syntax: none input.seta(number controller, number bitmask, number args...)

Set state for entiere controller. args is up to N values for indices (overriding values in bitmask if specified).

9.11 input.controllertype: Get controller type (deprecated)

- syntax: string input.controllertype(number controller)

Get the type of controller as string.

9.12 input.reset: Execute (delayed) reset

- Syntax: none input.reset([number cycles])

Execute reset. If <cycles> is greater than zero, do delayed reset. 0 (or no value) causes immediate reset.

- Note: Only available with subframe flag false.

9.13 input.raw: Return raw input data

- Syntax: table input.raw()

Returns table of tables of all available keys and axes. The first table is indexed by key name (platform-dependent!), and the inner table has the following fields:

- value: Last reported value for control
  - For keys: 1 for pressed, 0 for released.
  - For axes: -32767...32767.
  - For presure-sensitive buttons: 0...32767.
  - For hats: Bitmask: 1=>Up, 2=>Right, 4=>Down, 8=>Left.
  - For mouse: Coordinates relative to game area.
- ktype: Type of key (disabled, key, mouse, axis, hat, pressure).

9.14 input.keyhook: Hook a key

- Syntax: none input.keyhook(string key, boolean state)

Requests that keyhook events to be sent for key <key> (<state>=true) or not sent (<state>=false).

9.15 input.joyget: Get controls for controller

- Syntax: table input.joyget(number logical)

Returns table for current controls for specified logical controller <logical>. The names of fields vary by controller type.

- The buttons have the same name as those are referred to in other contexts in the emulator
- The analog axes are usually "xaxis" and "yaxis".
- Each field is numeric or boolean depending on axis/button.

9.16 input.joyset: Set controls for controller

- Syntax: none input.joyset(number controller, table controls)

Set the the state of specified controller to values specified in specified table.

- Each field can be boolean or number.
- Also, buttons allow strings, which cause value to be inverted.

9.17 input.lcid_to_pcid: Look up logical controller (deprecated)

- Syntax: (number, number, number) input.lcid_to_pcid(number lcid)

Returns the legacy pcid for controller (or false if there isn't one), followed by pcid pair. Returns nothing if controller does not exist.

10 Table keyboard

Various keybinding-related functions

10.1 keyboard.bind: Bind a key

- Syntax: none keyboard.bind(string mod, string mask, string key, string cmd)

Bind specified key with specified modifers to specified command.

10.2 keyboard.unbind: Unbind a key

- Syntax: none keyboard.unbind(string mod, string mask, string key)

Unbind specified key with specified modifers.

10.3 keyboard.alias: Set alias expansion

- Syntax: none keyboard.alias(string alias, string expansion)

Set expansion of given command.

11 Table subtitle

Subtitle handling

11.1 subtitle.byindex: Look up start and length of subtitle by index

- Syntax: (number, number) subtitle.byindex(number i)

Read the frame and length of ith subtitle. Returns nothing if not present.

11.2 subtitle.set: Write a subtitle

- Syntax: none subtitle.set(number f, number l, string txt)

Set the text of subtitle.

11.3 subtitle.get: Read a subtitle

- Syntax: string subtitle.get(number f, number l)

Get the text of subtitle.

11.4 subtitle.delete: Delete a subtitle

- Syntax: none subtitle.delete(number f, number l)

Delete specified subtitle.

12 Table hostmemory

Host memory handling (extra memory saved to savestates). Host memory starts empty.

- Reads out of range return false.
- Writes out of range extend the memory.

12.1 hostmemory.read: Read byte from host memory

- Syntax: number hostmemory.read(number address)

Reads byte from hostmemory slot address <address>.

12.2 hostmemory.write: Write byte to host memory

- Syntax: none hostmemory.write(number address, number value)

Writes hostmemory slot with value <value> 0-255.

12.3 hostmemory.read{,s}{byte,{,h,d,q}word}: Read from host memory

- Syntax: number hostmemory.readbyte(number address)
- Syntax: number hostmemory.readsbyte(number address)
- Syntax: number hostmemory.readword(number address)
- Syntax: number hostmemory.readsword(number address)
- Syntax: number hostmemory.readhword(number address)
- Syntax: number hostmemory.readshword(number address)
- Syntax: number hostmemory.readdword(number address)
- Syntax: number hostmemory.readsdword(number address)
- Syntax: number hostmemory.readqword(number address)
- Syntax: number hostmemory.readsqword(number address)

Read elements (big-endian) from given address <address>.

- byte is 1 element
- word is 2 elements
- hword is 3 elements
- dword is 4 elements
- qword is 8 elements.
- The 's' variants do signed read.

12.4 hostmemory.read{float,double}: Read from host memory

- syntax: number hostmemory.readfloat(number address)
- Syntax: number hostmemory.readdouble(number address)

Read elements (big-endian) floating-pont from given address <address>.

12.5 hostmemory.write{,s}{byte,{,h,d,q}word}: Write to host memory

- Syntax: number hostmemory.writebyte(number address, number value)
- Syntax: number hostmemory.writesbyte(number address, number value)
- Syntax: number hostmemory.writeword(number address, number value)
- Syntax: number hostmemory.writesword(number address, number value)
- Syntax: number hostmemory.writehword(number address, number value)
- Syntax: number hostmemory.writeshword(number address, number value)
- Syntax: number hostmemory.writedword(number address, number value)
- Syntax: number hostmemory.writesdword(number address, number value)
- Syntax: number hostmemory.writeqword(number address, number value)
- Syntax: number hostmemory.writesqword(number address, number value)

Write value <value> to elements (little-endian) starting from given address <address>.

- byte is 1 element
- word is 2 elements
- hword is 3 elements
- dword is 4 elements
- qword is 8 elements.
- The 's' variants do signed write.

12.6 hostmemory.write{float,double}: Write to host memory

- syntax: none hostmemory.readfloat(number address, number value)
- Syntax: none hostmemory.readdouble(number address, number value)

Write elements (big-endian) floating-pont to given address <address>, storing <value>.

13 Table movie

Movie handling

13.1 movie.currentframe: Get current frame number

- Syntax: number movie.currentframe()

Return number of current frame.

13.2 movie.framecount: Get move frame count

- Syntax: number movie.framecount()

Return number of frames in movie.

13.3 movie.lagcount: Get current lag count

- Syntax: number movie.lagcount()

Return number of lag frames recorded so far.

13.4 movie.readonly: Is in playback mode?

- Syntax: boolean movie.readonly()

Return true if in playback mode, false if in recording.

13.5 movie.rerecords: Movie rerecord count

- Syntax: number movie.rerecords()

Returns the current value of rerecord count.

13.6 movie.set_readwrite: Set recording mode.

- Syntax: none movie.set_readwrite()

Set recording mode (does not cause on_readwrite callback).

13.7 movie.frame_subframes: Count subframes in frame

- Syntax: number movie.frame_subframes(number frame)

Count number of subframes in specified frame <frame> (frame numbers are 1-based) and return that.

13.8 movie.read_subframes: Read subframe data (deprecated)

- Syntax: table movie.read_subframes(number frame, number subframe)

Read specifed subframe in specified frame and return data as array.

13.9 movie.read_rtc: Read current RTC time

- Syntax: (number, number) movie.read_rtc()

Returns the current value of the RTC as a pair (second, subsecond).

13.10 movie.unsafe_rewind: Fast movie rewind to saved state

- Syntax: none movie.unsafe_rewind([UNSAFEREWIND state])

Start setting point for unsafe rewind or jump to point of unsafe rewind.

- If called without argument, causes emulator to start process of setting unsafe rewind point. When this has finished, callback on_set_rewind occurs, passing the rewind state to lua script.
- If called with argument, causes emulator rewind to passed rewind point as soon as possible. recording mode is implicitly activated.

The following warnings apply to unsafe rewinding:

- There are no safety checks against misuse (that's what "unsafe" comes from)!
- Only call rewind from timeline rewind point was set from.
- Only call rewind from after the rewind point was set.

13.11 movie.to_rewind: Load savestate as rewind point

- Syntax: UNSAFEREWIND movie.to_rewind(string filename)

Load specified savestate file <filename> as rewind point and return UNSAFEREWIND corresponding to it.

- Note: This operation does not take emulated time.

13.12 movie.copy_movie/INPUTMOVIE::copy_movie: Copy movie to movie object

- Syntax: INPUTMOVIE movie.copy_movie([INPUTMOVIE/string movie])
- Syntax: INPUTMOVIE INPUTMOVIE::copy_movie()

Copies specified movie or branch <movie>/current object (if none or nil, the active movie) as new movie object.

13.13 movie.get_frame/INPUTMOVIE::get_frame: Read specified frame in movie.

- Syntax: INPUTFRAME movie.get_frame([INPUTMOVIE/string movie,] number frame)
- Syntax: INPUTFRAME INPUTMOVIE::get_frame(number frame);

Get INPUTFRAME object corresponding to specified frame in specified movie or branch.

13.14 movie.set_frame/INPUTMOVIE::set_frame: Write speicifed frame in movie.

- Syntax: none movie.set_frame([INPUTMOVIE/string movie,] number frame, INPUTFRAME data)
- Syntax: none INPUTMOVIE::set_frame(number frame, INPUTFRAME data)

Set data in specified frame.

- Note: Past can't be edited in active movie.

13.15 movie.get_size/INPUTMOVIE::get_size: Get size of movie

- Syntax: integer movie.get_size([INPUTMOVIE/string movie])
- Syntax: integer INPUTMOVIE::get_size()

Return number of subframes in specified movie or branch.

13.16 movie.count_frames/INPUTMOVIE::count_frames: Count frames in movie

- Syntax: number movie.count_frames([INPUTMOVIE/string movie])
- Syntax: number INPUTMOVIE::count_frames()

Return number of frames in movie.

13.17 movie.find_frame/INPUTMOVIE::find_frame: Find subframe corresponding to frame

- Syntax: number movie.find_frame([INPUTMOVIE/string movie], number frame)
- Syntax: number INPUTMOVIE::find_frame(number frame)

Returns starting subframe of given frame (frame numbers are 1-based). Returns -1 if frame number is bad.

13.18 movie.blank_frame/INPUTMOVIE::blank_frame: Return a blank frame

- Syntax: INPUTFRAME movie.blank_frame([INPUTMOVIE/string movie])
- Syntax: INPUTFRAME INPUTMOVIE::blank_frame()

Return blank INPUTFRAME with frame type from specified movie.

13.19 movie.append_frames/INPUTMOVIE::append_frames: Append blank frames

- Syntax: none movie.append_frames([INPUTMOVIE/string movie,] number frames)
- Syntax: none INPUTMOVIE::append_frames(number frames)

Append specified number <frames> of frames.

13.20 movie.append_frame/INPUTMOVIE::append_frame: Append a frame

- Syntax: none movie.append_frame([INPUTMOVIE/string movie,] INPUTFRAME frame)
- Syntax: none INPUTMOVIE::append_frame(INPUTFRAME frame)

Append specified frame <frame>. Past of current movie can't be edited.

13.21 movie.truncate/INPUTMOVIE::truncate: Truncate a movie.

- Syntax: none movie.truncate([INPUTMOVIE/string movie,] number frames)
- Syntax: none INPUTMOVIE::truncate(number frames)

Truncate the specified movie to specified number of frames.

13.22 movie.edit/INPUTMOVIE::edit: Edit a movie

- Syntax: none movie.edit([INPUTMOVIE movie,] number frame, number port, number controller, number control, number/bool value)
- Syntax: none movie.edit(string branch, number frame, number port, number controller, number control, number/bool value)
- Syntax: none INPUTMOVIE::edit(number frame, number port, number controller, number control, number/bool value)

Change specified control in specified frame in specified movie. Past can't be edited in active movie.

13.23 movie.copy_frames2: Copy frames between movies

- Syntax: none movie.copy_frames2([INPUTMOVIE/string dstmov,] number dst, [INPUTMOVIE/string srcmov,] number src, number count)

Copy specified number of frames between two movies. The copy proceeeds in forward direction.

13.24 movie.copy_frames/INPUTMOVIE::copy_frames: Copy frames in movie

- Syntax: none movie.copy_frames([INPUTMOVIE/string mov,] number dst, number src, number count, bool backwards)
- Syntax: none INPUTMOVIE::copy_frames(number dst, number src, number count, bool backwards)

Copy specified number of frames from one point in movie to another. If backwards is true, the copy will be done backwards.

13.25 movie.serialize/INPUTMOVIE::serialize: Serialize movie

- Syntax: none movie.serialize([INPUTMOVIE movie/string,] string filename, bool binary)
- Syntax: none INPUTMOIVE::serialize(string filename, bool binary)

Serialize given movie into file. If binary is true, binary format (more compact and much faster) is used.

13.26 movie.unserialize: Unserialize movie

- Syntax: INPUTMOVIE movie.unserialize(INPUTFRAME template, string filename, bool binary)

Unserialize movie from file. The given frame is used as template to decide the frame type. If binary is true, binary format is decoded (much faster).

13.27 movie.current_first_subframe: Return first subframe in current frame

- Syntax: number movie.current_first_subframe()

Returns first subframe in current frame.

13.28 movie.pollcounter: Return poll counter for speified control

- Syntax: number movie.pollcounter(number port, number controller, number control)

Returns number of times the specified control has been polled this frame.

13.29 movie.current_branch: Return current branch

- Syntax: string movie.current_branch()

Returns the name of the current branch.

13.30 movie.get_branches: Return names of all branches

- Syntax: string... movie.get_branches()

Returns the name of all branches.

13.31 movie.rom_loaded: Is ROM loaded?

- Syntax: boolean movie.rom_loaded()

Returns true if ROM is loaded, false otherwise.

13.32 movie.get_rom_info: Get info about loaded ROM

- Syntax: Table movie.get_rom_info()

Returns a table of tables. The outer table is indexed by ROM slot index. The inner table has the following fields:

- filename: The name of file
- hint: File name hint
- sha256: SHA-256 hash of ROM.

- xml_filename: The name of file for markup
- xml_hint: File name hint for markup
- xml_sha256: SHA-256 hash of ROM markup.

If there is no markup, the xml_* fields are absent.

13.33 movie.get_game_info: Get info about loaded game

- Syntax: Table movie.get_game_info()

Returns a table with following fields:

- core: The name of the core
- core_short: Short name of the core
- type: The name of the system type
- type_long: The long name of the system type
- region: The name of region
- region_long: The long name of region
- gametype: The gametype of movie.
- fps: Nominal fps as floating-point value
- fps_n: Exact nominal fps numerator
- fps_d: Exact nominal fps denominator.

13.34 INPUTFRAME::get_button: Get button

- Syntax: boolean INPUTFRAME::get_button(number port, number controller, number control)

Returns state of given button as boolean.

13.35 INPUTFRAME::get_axis: Get axis

- Syntax: number INPUTFRAME::get_axis(number port, number controller, number control)

Returns state of given axis as number.

13.36 INPUTFRAME::set_button/INPUTFRAME::set_axis: Set button or axis

- Syntax: none INPUTFRAME::set_button(number port, number controller, number control, number/bool value)
- Syntax: none INPUTFRAME::set_axis(number port, number controller, number control)

Set the given button/axis to given value.

13.37 INPUTFRAME::serialize: Serialize a frame

- Syntax: string INPUTFRAME::serialize()

Return string representation of frame.

13.38 INPUTFRAME::unserialize: Unserialize a frame

- Syntax: none INPUTFRAME::unserialize(string data)

Set current frame from given data.

13.39 INPUTFRAME::get_stride: Get movie stride

- Syntax: number INPUTFRAME::get_stride()

Return number of bytes needed to store the input frame. Mainly useful for some debugging.

14 Table settings

Routines for settings manipulation

14.1 settings.get: Get value of setting

- Syntax: string settings.get(string name)

Get value of setting <name>. If setting value can't be obtained, returns (nil, error message).

14.2 settings.set: Set value of setting

- Syntax: none settings.set(string name, string value)

Set value <value> of setting <name>. If setting can't be set, returns (nil, error message).

14.3 settings.get_all: Get values of all settings

- Syntax: table settings.get_all()

Return a table with all setting names as keys and all current values as values.

14.4 settings.get_speed: Get current speed

- Syntax: number/string settings.get_speed()

Return the current speed multiplier (1 is normal), or "turbo" if speed is set to turbo (this does not react to turbo toggle).

14.5 settings.set_speed: Set current speed

- Syntax: settings.get_speed(number spd)
- Syntax: settings.get_speed(string special)

Set the current speed multiplier (1 is normal). The speed may be positive multiplier or "turbo" for turbo speed.

15 Table memory

Contains various functions for managing memory

15.1 memory.vma_count: Count number of memory areas.

- Syntax: number memory.vma_count()

Returns the number of memory areas

15.2 memory.read_vma: Lookup memory area info by index

- Syntax: table memory.read_vma(number index)

Reads the specified memory area (indices start from zero). Trying to read invalid memory area gives nil. The return value is table with the following fields:

- region_name (string): The readable name of the memory area
- baseaddr (number): Base address of the memory area
- lastaddr (number): Last address in the memory area.
- size (number): The size of memory area in bytes.
- readonly (boolean): True of the memory area corresponds to ROM.
- iospace (boolean): True if the memory area is I/O space.
- native_endian (boolean): True if the memory area has native endian as opposed to little endian.

15.3 memory.find_vma: Find memory area info by address

- Syntax: table memory.find_vma(number address)

Finds the memory area containing specified address. Returns table in the same format as read_vma or nil if not found.

15.4 memory.read{,s}{byte,{,h,d,q}word}: Read memory

- Syntax: none memory.readbyte({string marea, number address|ADDRESS addrobj})
- Syntax: none memory.readword({string marea, number address|ADDRESS addrobj})
- Syntax: none memory.readhword({string marea, number address|ADDRESS addrobj})
- Syntax: none memory.readdword({string marea, number address|ADDRESS addrobj})
- Syntax: none memory.readqword({string marea, number address|ADDRESS addrobj})
- Syntax: none memory.readsbyte({string marea, number address|ADDRESS addrobj})
- Syntax: none memory.readsword({string marea, number address|ADDRESS addrobj})
- Syntax: none memory.readshword({string marea, number address|ADDRESS addrobj})
- Syntax: none memory.readsdword({string marea, number address|ADDRESS addrobj})
- Syntax: none memory.readsqword({string marea, number address|ADDRESS addrobj})

Reads the specified address <address> (if 's' variant is used, do undergo 2's complement).

15.5 memory.{,s}read_sg: Scatter/Gather read memory

- Syntax: none memory.read_sg(string/boolean/number...)
- Syntax: none memory.sread_sg(string/boolean/number...)

Perform (2s complement signed if using memory.sread_sg) scatter/gather read of memory. Each argument can be string, boolean or number:

- String: Set memory area addresses are relative to (e.g. 'WRAM').
- boolean: If true, increment relative address by 1, if false, decrement by 1. The new address is read as next higher byte.
- integer: Set the relative address to specified value and read the address as next higher byte.

15.6 memory.write_sg: Scatter/Gather write memory

- Syntax: none memory.write_sg(number value, string/boolean/number...)

Perform scatter/gather write of value <value> on memory. Each argument can be string, boolean or number:

- String: Set memory area addresses are relative to (e.g. 'WRAM').
- boolean: If true, increment relative address by 1, if false, decrement by 1. The new address is read as next higher byte.
- integer: Set the relative address to specified value and read the address as next higher byte.

15.7 memory.read{float,double}: Read memory

- Syntax: none memory.readfloat({string marea, number address|ADDRESS addrobj})
- Syntax: none memory.readdouble({string marea, number address|ADDRESS addrobj})

Reads the specified address <address>

15.8 memory.write{byte,{,h,d,q}word,float,double}: Write memory

- Syntax: none memory.writebyte({string marea, number address|ADDRESS addrobj}, number value)
- Syntax: none memory.writeword({string marea, number address|ADDRESS addrobj}, number value)
- Syntax: none memory.writehword({string marea, number address|ADDRESS addrobj}, number value)
- Syntax: none memory.writedword({string marea, number address|ADDRESS addrobj}, number value)
- Syntax: none memory.writeqword({string marea, number address|ADDRESS addrobj}, number value)
- Syntax: none memory.writefloat({string marea, number address|ADDRESS addrobj}, number value)
- Syntax: none memory.writedouble({string marea, number address|ADDRESS addrobj}, number value)

Writes the specified value <value> (negative integer values undergo 2's complement) to specified address <address>.

15.9 memory.map{{,s}{byte,{,h,d,q}word},float,double}: Map an array

- Syntax: userdata memory.map<type>({string marea, number base|ADDRESS addrobj}, number size)

Returns a table mapping specified memory aperture for read/write.

- Type may be one of: byte, sbyte, word, sword, hword, shword, dword, sdword, qword, sqword, float or double.

15.10 memory.hash_region: Hash region of memory

- Syntax: string memory.hash_region({string marea, number base|ADDRESS addrobj}, number size)

Hash <size> bytes starting from address <base> (relative to <marea>) and return the SHA-256.

15.11 memory.hash_region2: Hash region of memory

- Syntax: string memory.hash_region2({string marea, number base|ADDRESS addrobj}, number size[, number rows, number stride])

Hash <rows> blocks of <size> bytes starting from address <base> (relative to <marea>). The blocks are offset by <stride> from one another and return the SHA-256.

15.12 memory.hash_region_skein: Hash region of memory

- Syntax: string memory.hash_region_skein({string marea, number base|ADDRESS addrobj}, number size[, number rows, number stride])

Same as memory.hash_region2, but uses Skein-512-256 (v1.3; one of the SHA-3 finalists) as hash function.

15.13 memory.store: Store region of memory

- Syntax: none memory.store({string marea, number addr|ADDRESS addrobj}, number daddr[, number rows, number stride]

Copy memory starting from <addr> in memory area <marea> (each row being of size <size>, there being <rows> rows, and rows being separated by <stride> in memory) into savestate-saved memory area, starting from <daadr> (all rows are written back to back).

15.14 memory.storecmp: Compare and store region of memory

- Syntax: bool memory.storecmp({string marea, number addr|ADDRESS addrobj}, number daddr[, number rows, number stride]

Like memory.store, but returns true if target of copy already held the value that would be copied before the copy happened. Otherwise returns false (if target and source differ before copy).

15.15 memory.hash_state: Hash system state

- Syntax: string memory.hash_state()

Hash the current system state. Mainly useful for debugging savestates.

15.16 memory.readregion: Read region of memory

- Syntax: table memory.readregion({string marea, number base|ADDRESS addrobj}, number size)

Read a region of memory.

- Warning: If the region crosses memory area boundary, the results are undefined.

15.17 memory.writeregion: Write region of memory

- Syntax: none memory.writeregion({string marea, number base|ADDRESS addrobj}, number size, table data)

Write a region of memory.

- Warning: If the region crosses memory area boundary, the results are undefined.

15.18 memory.action: Run core action

- memory.action(string action, [<params>])

Run core action. The different models expect parameters as:

- string: String
- numeric: numeric
- enumeration: String
- boolean: String
- toggle: None.

15.19 memory.action_flags: Get core action flags

- memory.action_flags(string action)

Get value of action flags for core action <action>.

- Bit 0: Enabled?
- Bit 1: Selected (not all actions can be selected)?

15.20 memory.get_lag_flag: Get lag flag

- Syntax: boolean memory.get_lag_flag()

Get the value of core lag flag. True if this frame has been lag so far, false if poll has been detected.

15.21 memory.set_lag_flag: Set lag flag

- Syntax: none memory.set_lag_flag(boolean flag)

Set the value of core lag flag. This flag automatically gets cleared if poll is detected, but can be forcibly set or cleared if game so requires.

- Should only be used in on_frame_emulated callback.
- Setting or clearing this affects the emulator lag counter.

15.22 memory.{,un}register{read,write,exec}: (Un)Register read / write / execute callback

- Syntax: function memory.registerread({string marea, number addr|ADDRESS addrobj}, function fn);
- Syntax: function memory.registerwrite({string marea, number addr|ADDRESS addrobj}, function fn);
- Syntax: function memory.registerexec({string marea, number addr|ADDRESS addrobj}, function fn);
- Syntax: none memory.unregisterread({string marea, number addr|ADDRESS addrobj}, function fn);
- Syntax: none memory.unregisterwrite({string marea, number addr|ADDRESS addrobj}, function fn);
- Syntax: none memory.unregisterexec({string marea, number addr|ADDRESS addrobj}, function fn);

Add or remove callback on memory read, write or execute (depending on the function). <addr> is relative to <marea>. <fn> is the callback. The register* functions return <fn> (which can then be passed to unregister* functions.

- Not all cores support this, and it may be unsupported for some memory areas.
- The functions are passed two parameters: Address and value.

15.23 memory.{,un}registertrace: Set/Clear trace hook

- Syntax: function memory.registertrace(number processor, function fn);
- Syntax: none memory.unregistertrace(number processor, function fn);

Add or remove trace callback. <processor> is system-dependent processor number (0 is usually main CPU). The function arguments work like in other (un)register* functions.

- The functions are passed two parameters: Trace CPU and Trace event string.

15.24 memory.cheat: Set cheat

- Syntax: none memory.cheat({string marea, number addr|ADDRESS addrobj}, number value);
- Syntax: none memory.cheat({string marea, number addr|ADDRESS addrobj});

Set or clear cheat (value <value>) on address <addr> (relative to <marea>). If <value> is not speicified, clear a cheat.

- Not all cores support this, and it may be unsupported for some memory areas.

15.25 memory.setxmask: Set global execute hook mask

- Syntax: none memory.setxmask(number mask)

Set the global execute hook mask to <mask>. The meaning of each bit is system-dependent, but bit 0 should be the main CPU.

15.26 memory.getregister: Get register value

- Syntax: number/boolean memory.getregister(string register)

Get the value of named register.

15.27 memory.getregisters: Get register values

- Syntax: table memory.getregisters()

Get the value of all known registers as table.

15.28 memory.setregister: Set register value

- Syntax: none memory.setregister(string register, number/boolean value)

Set the value of named register.

15.29 memory.mmap: Class MMAP_STRUCT

See class MMAP_STRUCT

16 Table memory2

Contains newer memory functions.

16.1 memory2(): Get all memory area names.

- Syntax: table memory2()

Returns array of all valid memory area names.

16.2 memory2.<marea>:info: Get memory area info

- Syntax: table memory2.<marea>:info()

Return table describing given memory area. Includes fields address, size, last, readonly, special and endian.

16.3 memory2.<marea>:<op>: Read/Write memory

- Syntax: none memory2.<marea>:<op>(number offset, number value)
- Syntax: number memory2.<marea>:<op>(number offset)

Read/Write value from/to given memory area <marea> at given offset <offset> (must be in-range). The value written is <value>. <Op> is of form: [i][s]<type>, where:

- <type> is one of 'byte', 'word', 'hword', 'dword', 'qword', 'float', 'double'.
- 'i' signifies that the value is treated as opposite-to-normal endianess,
- 's' signifies that value is treated as signed (not available for floating-point).

16.4 memory2.<marea>:read: Scatter-gather value read

- Syntax: number memory2.<marea>:read(number addr...)

Read value from given memory area <marea> at byte offsets <addr>..., given in order of increasing significance. Value of true and false are special. True increments address by 1, and false decrements address by 1.

16.5 memory2.<marea>:sread: Signed scatter-gather value read

- Syntax: number memory2.<marea>:sread(number addr...)

Like memory2.<marea>:read, but reads signed values.

16.6 memory2.<marea>:write: Scatter-gather value write

- Syntax: number memory2.<marea>:write(number val, number addr...)

Write value <val> to given memory area <marea> at byte offsets <addr>..., given in order of increasing significance. Value of true and false are special. True increments address by 1, and false decrements address by 1.

16.7 memory2.<marea>:cheat: Set/Clear cheat

- Syntax: none memory2.<marea>:cheat(number addr, [number value])

Set/Clear cheat at offset <addr> of memory area <marea>. If <value> is given, cheat with specified value is set. Otherwise cheat on address is removed.

16.8 memory2.<marea>:sha256: SHA-256

- Syntax: string memory2.<marea>:sha256(number addr, number size[, number rows, number stride])

Compute SHA-256 of <rows> (default 1) chunks of <size> bytes each, starting from offset <addr> of area <marea>. The chunks are separated by <stride>.

16.9 memory2.<marea>:skein: Skein-512-256

- Syntax: string memory2.<marea>:skein(number addr, number size[, number rows, number stride])

Same as memory2.<marea>:sha256, except with Skein-512-256 as hash function.

16.10 memory2.<marea>:store{,cmp}: Copy region to Lua memory with compare

- Syntax: none memory2.<marea>:store(number addr, number daddr, number size[, number rows, number stride])
- Syntax: boolean memory2.<marea>:storecmp(number addr, number daddr, number size[, number rows, number stride])

Copy <rows> (default 1) chunks of <size> bytes each, starting from offset <addr> of area <marea>. The chunks are separated by <stride>. The target is Lua host memory, starting from offset <daddr>.

Additionally, the storecmp method returns false if target was modified (otherwise true).

16.11 memory2.<marea>:readregion: Read region

- Syntax table memory2.<marea>:readregion(number addr, number size)

Read <size> bytes starting from <addr> in <marea> and return as array.

16.12 memory2.<marea>:writeregion: Write region

- Syntax none memory2.<marea>:writeregion(number addr, table data)

Write array <data> to bytes starting from <addr> in <marea>.

16.13 memory2.<marea>:register{read,write,exec}: Register hook

- Syntax: function memory2.<marea>:registerread(number addr, function fn);
- Syntax: function memory2.<marea>:registerwrite(number addr, function fn);
- Syntax: function memory2.<marea>:registerexec(number addr, function fn);

Register debug callback <fn> of specified type at offset <addr> of memory area <marea>. Returns <fn>.

16.14 memory2.<marea>:unregister{read,write,exec}: Unregister hook

- Syntax: none memory2.<marea>:unregisterread(number addr, function fn);
- Syntax: none memory2.<marea>:unregisterwrite(number addr, function fn);
- Syntax: none memory2.<marea>:unregisterexec(number addr, function fn);

Unregister debug callback <fn> of specified type at offset <addr> of memory area <marea>.

17 Table random

Contains random number generation methods. These functions do not return reproducable results.

17.1 random.boolean: Random boolean

- Syntax: boolean random.boolean()

Returns true or false at random (50-50 chance).

17.2 random.integer: Random integer

- Syntax: number random.integer(number highplusone)
- Syntax: number random.integer(number low, number high)

With one argument, return random integer [0,<highplusone>) (upper end exclusive). With two arguments, return random integer [<low>,<high>] (both ends inclusive).

The returned numbers are from uniform distribution.

17.3 random.float: Random float

- Syntax: number random.float()

Returns random decimal number [0,1).

17.4 random.among: Random parameter

- Syntax: value random.among(value values...)

Returns random parameter value, picked at uniform. Multiple equivalent values are returned with higher chance.

17.5 random.amongtable: Random from table

- Syntax: value random.amongtable(table tab)

Returns random value from table <tab>. As in random.among, no equality testing is done.

18 Table zip

18.1 zip.enumerate: Enumerate members in zipfile

- Syntax: Table zip.enumerate(string filename[, boolean invert])

Returns table of files in zip archive <filename>. If <invert> is true, instead of returning array of names, returns table with keys being member names and values being true.

18.2 zip.writer: Class ZIPWRITER

See class ZIPWRITER.

19 Table callback

Various callback-related functions.

19.1 callback.register: Register a callback

- Syntax: function callback.register(string cbname, function cbfun);

Instruct function <cbfun> to be added to list of callbacks to call on event <cbname> (See section 23). The callback name does not have the 'on_' prefix (e.g. "paint"). Returns <cbfun>.

19.2 callback.unregister: Unregister a callback

- Syntax: function callback.unregister(string cbname, function cbfun);

Instruct function <cbfun> to be removed from list of callbacks to call on event <cbname>.

19.3 callback.<cbname>:register: Register callback

- Syntax: function callback.<cbname>:register(function cbfun)

Synonym for callback.register (section 19.1), albeit with callback name specified differently.

19.4 callback.<cbname>:unregister: Register callback

- Syntax: function callback.<cbname>:unregister(function cbfun)

Synonym for callback.unregister (section 19.2), albeit with callback name specified differently.

20 table bsnes

Various bsnes-specific functions.

20.1 bsnes.dump_sprite: Dump a sprite

- Syntax: BITMAP bsnes.dump_sprite(string marea, number addr, number width, number height[, number stride])

Dumps given sprite (in native format) from memory. memory area is usually "VRAM". <Width> and <height> are given in 8x8 blocks. <Stride> overrides row stride (default 512).

20.2 bsnes.dump_palette: Dump a palette

- Syntax: PALETTE bsnes.dump_palette(string marea, number addr, bool full256, bool first_trans)

Dumps a palette from memory. memory area is usually "CGRAM". If <full256> is true, 256 colors are dumped (otherwise 16). If <first_trans> is true, first color is forced transparent.

20.3 bsnes.enablelayer: Set layer visibility

- Syntax: none bsnes.enablelayer(number layer, number priority, boolean enabled)

Enable or disable specified layer at specified priority.

20.4 bsnes.redump_sprite: Redump a sprite

- Syntax: none bsnes.redump_sprite(BITMAP bitmap, string marea, number addr[, number stride])

Like bsnes.dump_sprite, but instead dumps to specified bitmap <bitmap>. The specified bitmap must have size multiple of 8x8.

20.5 bsnes.redump_palette: Redump a palette

- Syntax: none bsnes.dump_palette(PALETTE pal, string marea, number addr, bool first_trans)

Like bsnes.dump_palette, but instead dumps to specified palette <pal>. The specified palette must have either 16 or 256 colors.

21 extensions to table string

21.1 string.charU: string.char, UTF-8 version.

- Syntax: string string.charU(number n...)

Like Lua string.char(), but works in terms of Unicode codepoints. The returned string is UTF-8.

21.2 string.byteU: string.byte, UTF-8 version.

- Syntax: number... string.byteU(string str[, number i[, number j]])

Like string.byte(), but works in terms of Unicode codepoints. The input string <str> is assumed UTF-8.

21.3 string.regex: Match string against regular expression

- Syntax: boolean/string... string.regex(string regexp, string against)

Match POSIX-extended regular expression <regexp> against string <against>. If no match, false is returned. Otherwise if string has no subcaptures, true is returned. Otherwise each subcapture is returned as a string (in order of starting position).

21.4 string.hex: Transform integer into hex string

- Syntax: string string.hex(number n, [number digits])

Returns hexadecimal string representation of <n>, optionally padded with zeroes to <digits> digits (default is not to pad).

21.5 string.lpad: Pad string with spaces from left

- Syntax: string string.lpad(string x, number n)

Pad string <x> to <n> bytes by inserting spaces at start and return the result.

21.6 string.rpad: Pad string with spaces from right

- Syntax: string string.rpad(string x, number n)

Pad string <x> to <n> bytes by inserting spaces at end and return the result.

22 Table _SYSTEM

Contains copy of global variables from time of Lua initialization. Non-writeable.

23 Callbacks

Various callbacks to Lua that can occur.

23.1 on_paint: Screen is being painted

- Callback: on_paint(bool not_synth)

Called when screen is being painted. Any gui.* calls requiring graphic context draw on the screen.

- not_synth is true if this hook is being called in response to received frame, false otherwise.

23.2 on_video: Dumped video frame is being painted

- Callback: on_video()

Called when video dump frame is being painted. Any gui.* calls requiring graphic context draw on the video.

23.3 on_frame_emulated: Frame emulation complete

- Callback: on_frame_emulated()

Called when emulating frame has completed and on_paint()/on_video() calls are about to be issued.

23.4 on_frame: Frame emulation starting.

- Callback: on_frame()

Called on each starting whole frame.

23.5 on_rewind: Movie rewound to beginning

- Callback: on_rewind()

Called when rewind movie to beginning has completed.

23.6 on_pre_load: Load operation is about to start

- Callback: on_pre_load(string name)

Called just before savestate/movie load occurs (note: loads are always delayed, so this occurs even when load was initiated by lua).

23.7 on_err_Load: Load failed

- Callback: on_err_load(string name)

Called if loadstate goes wrong.

23.8 on_post_load: Load completed

- Callback: on_post_load(string name, boolean was_savestate)

Called on successful loadstate. was_savestate gives if this was a savestate or a movie.

23.9 on_pre_save: Save operation is about to start

- Callback: on_pre_save(string name, boolean is_savestate)

Called just before savestate save occurs (note: movie saves are synchronous and won't trigger these callbacks if called from Lua).

23.10 on_err_save: Save failed

- Callback: on_err_save(string name)

Called if savestate goes wrong.

23.11 on_post_save: Save completed

- Callback: on_post_save(string name, boolean is_savestate)

Called on successful savaestate. is_savestate gives if this was a savestate or a movie.

23.12 on_quit: Emulator is shutting down

- Callback: on_quit()

Called when emulator is shutting down.

23.13 on_input: Polling for input

Called when emulator is just sending input to bsnes core. Warning: This is called even in readonly mode, but the results are ignored.

23.14 on_reset: System has been reset

- Callback: on_reset()

Called when system is reset.

23.15 on_readwrite: Entered recording mode

- Callback: on_readwrite()

Called when moving into recording mode as result of "set-rwmode" command (note: moving to rwmode by Lua won't trigger this, as per recursive entry protection).

23.16 on_snoop/on_snoop2: Snoop core controller reads

- Callback: on_snoop(number port, number controller, number index, number value)
- Callback: on_snoop2(number port, number controller, number index, number value)

Called each time bsnes asks for input. The value is the final value to be sent to bsnes core (readonly mode, autohold and autofire have been taken into account). Might be useful when translating movies to format suitable for console verification. Note: There is no way to modify the value to be sent.

- On_snoop2 is called instead of on_snoop if defined. Reserves port 0 for system, having first user port be port 1.

23.17 on_keyhook: Hooked key/axis has been moved

- Callback: on_keyhook(string keyname, table state)

Sent when key that has keyhook events requested changes state. Keyname is name of the key (group) and state is the state (same kind as table values in input.raw).

23.18 on_idle: Idle event

- Callback: on_idle()

Called when requested by set_idle_timeout(), the timeout has expired and emulator is waiting.

23.19 on_timer: Timer event

- Callback: on_timer()

Called when requested by set_idle_timeout() and the timeout has expired (regardless if emulator is waiting).

23.20 on_set_rewind: Rewind point has been set

- Callback: on_set_rewind(UNSAFEREWIND r)

Called when unsafe rewind object has been constructed.

23.21 on_pre_rewind: Rewind is about to occur

- Callback: on_pre_rewind()

Called just before unsafe rewind is about to occur.

23.22 on_post_rewind: Rewind has occured

- Callback: on_post_rewind()

Called just after unsafe rewind has occured.

23.23 on_button: Button has been pressed

- Callback: on_button(number port, number controller, number index, string type)

Called on controller button press, with following parameters:

- port: Port number (0 is system)
- controller: Controller within port
- index: Index of button.
- type: Type of event, one of:
  - "pressed": Button was pressed.
  - "released": Button was released.
  - "hold": Held.
  - "unhold": Released from hold.
  - "type": Typing input on button.
  - "untype": Typing input undone.
  - "autofire <duty> <cycle>": Autofire with specifie duty and cycle.
  - "autofire": Stop autofire.
  - "analog": Analog action on axis.

23.24 on_movie_lost: Movie data is about to be lost

- Callback: on_movie_lost(STRING kind)

Called just before something would happen that could lose movie data. Kind can be:

- readwrite: Switching to recording mode.
- reload: ROM is being reloaded in recording mode.
- load: New movie is being loaded.
- unsaferewind: Unsafe rewind is happening.

23.25 on_latch: Latch line is rising

- Callback: on_latch(<core-dependent-parameters>)

Called when latch line for controller is rising. Some cores may not support this.

24 System-dependent behaviour

24.1 bsnes core

- Registers are: pbpc, pb, pc, r0, r1, r2, r3, r4, r5, a, x, y, z, s, d, db, p, e, irq, wai, mdr, vector, aa, rd, sp, dp, p_n, p_v, p_m, p_x, p_d, p_i, p_z, p_c, ppu_display_disabled, ppu_oam_priority, ppu_bg_tilesize[0], ppu_bg_tilesize[1], ppu_bg_tilesize[2], ppu_bg_tilesize[3], ppu_bg3_priority, ppu_mosaic_enabled[0], ppu_mosaic_ ppu_mosaic_enabled[2], ppu_mosaic_enabled[3], ppu_vram_incmode, ppu_mode7_vflip, ppu_mode7_hflip, ppu_window1_enabled[0], ppu_window1_enabled[1], ppu_window1_enabled[2], ppu_window1_enabled[3], ppu_win ppu_window1_enabled[5], ppu_window1_invert[0], ppu_window1_invert[1], ppu_window1_invert[2], ppu_window1 ppu_window1_invert[4], ppu_window1_invert[5], ppu_window2_enabled[0], ppu_window2_enabled[1], ppu_window ppu_window2_enabled[3], ppu_window2_enabled[4], ppu_window2_enabled[5], ppu_window2_invert[0], ppu_wind ppu_window2_invert[2], ppu_window2_invert[3], ppu_window2_invert[4], ppu_window2_invert[5], ppu_bg_enable ppu_bg_enabled[1], ppu_bg_enabled[2], ppu_bg_enabled[3], ppu_bg_enabled[4], ppu_bgsub_enabled[0], ppu_bgs ppu_bgsub_enabled[2], ppu_bgsub_enabled[3], ppu_bgsub_enabled[4], ppu_window_enabled[0], ppu_window_ena ppu_window_enabled[2], ppu_window_enabled[3], ppu_window_enabled[4], ppu_sub_window_enabled[0], ppu_sub_window_enabled[1], ppu_sub_window_enabled[2], ppu_sub_window_enabled[3], ppu_sub_window_ena ppu_addsub_mode, ppu_direct_color, ppu_color_mode, ppu_color_halve, ppu_color_enabled[0], ppu_color_enab ppu_color_enabled[2], ppu_color_enabled[3], ppu_color_enabled[4], ppu_color_enabled[5], ppu_mode7_extbg, ppu_pseudo_hires, ppu_overscan, ppu_oam_interlace, ppu_interlace, ppu_latch_hcounter, ppu_latch_vcounter, ppu_counters_latched, ppu_time_over, ppu_range_over, ppu_ppu1_mdr, ppu_ppu2_mdr, ppu_bg_y[0], ppu_bg_y[1], ppu_bg_y[2], ppu_bg_y[3], ppu_ioamaddr, ppu_icgramaddr, ppu_display_brightness, ppu_oam_bas ppu_oam_nameselect, ppu_oam_tdaddr, ppu_oam_baseaddr, ppu_oam_addr, ppu_oam_firstsprite, ppu_oam_lat ppu_bg_mode, ppu_mosaic_size, ppu_mosaic_countdown, ppu_bg_scaddr[0], ppu_bg_scaddr[1], ppu_bg_scaddr[ppu_bg_scaddr[3], ppu_bg_scsize[0], ppu_bg_scsize[1], ppu_bg_scsize[2], ppu_bg_scsize[3], ppu_bg_tdaddr[0], ppu_bg_tdaddr[1], ppu_bg_tdaddr[2], ppu_bg_tdaddr[3], ppu_bg_ofslatch, ppu_m7_hofs, ppu_m7_vofs, ppu_bg_hofs[0], ppu_bg_hofs[1], ppu_bg_hofs[2], ppu_bg_hofs[3], ppu_bg_vofs[0], ppu_bg_vofs[1], ppu_bg_vofs ppu_bg_vofs[3], ppu_vram_mapping, ppu_vram_incsize, ppu_vram_addr, ppu_mode7_repeat, ppu_m7_latch, ppu_m7a, ppu_m7b, ppu_m7c, ppu_m7d, ppu_m7x, ppu_m7y, ppu_cgram_addr, ppu_cgram_latchdata, ppu_window1_left, ppu_window1_right, ppu_window2_left, ppu_window2_right, ppu_window_mask[0], ppu_window_mask[1], ppu_window_mask[2], ppu_window_mask[3], ppu_window_mask[4], ppu_window_mask[5], ppu_color_mask, ppu_colorsub_mask, ppu_color_r, ppu_color_g, ppu_color_b, ppu_color_rgb, ppu_scanlines, ppu_hcounter, ppu_vcounter, ppu_vram_readbuffer, ppu_oam_itemcount, ppu_oam_tilecount,
- on_latch has no parameters
- CPU 0 is S-CPU, 1 is S-SMP.
- Cheats are supported for ROM, SRAM, WRAM, BSXFLASH, SLOT{A,B}_{RAM,ROM}.
- Read/Write/Execute hooks are supported for ROM, SRAM, WRAM, BSXFLASH, SLOT{A,B}_{RAM,ROM}.
- Memory areas are: WRAM, APURAM, VRAM, OAM, CGRAM, RTC, DSPRAM, DSPPROM, DSPDROM, SRAM, ROM, BUS, PTRTABLE, CPU_STATE, PPU_STATE, SMP_STATE, DSP_STATE, BSXFLASH, BSX_RAM, BSX_PRAM, SLOTA_ROM, SLOTB_ROM, SLOTA_RAM, SLOTB_RAM, GBCPU_STATE, GBROM, GBRAM, GBWRAM, GBHRAM.

24.2 gambatte core

- Registers are: wrambank, cyclecounter, pc, sp, hf1, hf2, zf, cf, a, b, c, d, e, f, h, l
- on_latch is not supported
- CPU 0 is main CPU.
- Cheats are supported for ROM, SRAM and WRAM.
- Read/Write/Execute hooks are supported for ROM (read/execute only), SRAM and WRAM.
- Memory areas are: SRAM, WRAM, VRAM, IOAMHRAM, ROM, BUS.