G-series Lua API
V8.45

Overview and Reference
Contents

Contents..........................................................................................................................................................2
Overview..........................................................................................................................................................3
Reference.......................................................................................................................................................4
Standard Lua 5.1 Libraries..................................................................................................................................41
Appendix A....................................................................................................................................................42
Overview

The G-series Lua API is a set of functions using the Lua programming language and provides advanced scripting functionality for the G-series line of gaming keyboards.

This document assumes a working knowledge of the Lua programming language. Further information can be obtained from www.lua.org.

Every G-series Profile has a default Lua script bound to it which can be edited and customized. The script is invoked using an event handler, OnEvent. Users may examine the various events exposed in this handler to perform their desired actions.
Reference

Functions

OnEvent ................................................................. 5
GetMKeyState .......................................................... 7
SetMKeyState ........................................................... 8
Sleep ............................................................ 9
OutputLogMessage ....................................................... 10
GetRunningTime ......................................................... 11
GetDate ........................................................... 12
ClearLog ........................................................ 13
PressKey ........................................................ 14
ReleaseKey ......................................................... 15
PressAndReleaseKey .................................................... 16
IsModifierPressed .................................................... 17
PressMouseButton ..................................................... 18
ReleaseMouseButtonDown ............................................. 19
PressAndReleaseMouseButtonDown .................................. 20
IsMouseButtonPressed .................................................. 21
MoveMouseTo ........................................................ 22
MoveMouseWheel ...................................................... 23
MoveMouseRelative .................................................. 24
MoveMouseToVirtual .................................................. 25
GetMousePosition ..................................................... 26
OutputLCDMessage ..................................................... 27
ClearLCD ........................................................ 28
PlayMacro ........................................................ 29
AbortMacro .......................................................... 30
IsKeyLockOn .......................................................... 31
SetBacklightColor .................................................... 32
OutputDebugMessage .................................................. 33
SetMouseDPITable ...................................................... 34
SetMouseDPITableIndex ................................................ 35
EnablePrimaryMouseButtonEvents .................................. 36
SetSteeringWheelProperty ............................................ 37
G13 Programming ...................................................... 38
OnEvent

The `OnEvent()` function serves as the event handler for the script. You will need to implement this function.

```plaintext
function OnEvent(event, arg [, family])
end
```

**Parameters**

- **event**
  
  String containing the event identifier.

- **arg**
  
  Argument correlating to the appropriate identifier.

- **family**
  
  Family of device creating the hardware event. Empty if event is not hardware specific. Use this if you need to distinguish input from multiple devices.

<table>
<thead>
<tr>
<th>Family</th>
<th>Devices</th>
</tr>
</thead>
<tbody>
<tr>
<td>“kb”</td>
<td>Keyboard devices (G15, G11, G19, etc)</td>
</tr>
<tr>
<td>“lhc”</td>
<td>Left handed controllers (G13, etc)</td>
</tr>
<tr>
<td>“mouse”</td>
<td>Supported gaming mouse (G300, G400, etc)</td>
</tr>
</tbody>
</table>

**Return Values**

None

**Remarks**

The following is the list of identifiers and their arguments:

<table>
<thead>
<tr>
<th>Event</th>
<th>arg</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;PROFILE_ACTIVATED&quot;</td>
<td>None</td>
<td>Profile has been activated. This is the first event seen.</td>
</tr>
<tr>
<td>&quot;PROFILE_DEACTIVATED&quot;</td>
<td>None</td>
<td>Profile has been deactivated. This is the last event seen.</td>
</tr>
<tr>
<td>&quot;G_PRESSED&quot;</td>
<td>1=G1 18=G18 n=Gn</td>
<td>G Key pressed</td>
</tr>
<tr>
<td>&quot;G_RELEASED&quot;</td>
<td>1=G1 18=G18 n=Gn</td>
<td>G Key released</td>
</tr>
<tr>
<td>&quot;M_PRESSED&quot;</td>
<td>1=M1 2=M2 3=M3</td>
<td>M Key pressed</td>
</tr>
<tr>
<td>&quot;M_RELEASED&quot;</td>
<td>1=M1</td>
<td>M Key released</td>
</tr>
<tr>
<td>Event Type</td>
<td>Event Details</td>
<td>Note</td>
</tr>
<tr>
<td>------------------------------------</td>
<td>--------------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| "MOUSE_BUTTON_PRESSED"             | 2 = Mouse Button 2 3 = Mouse Button 3 4 = Mouse Button 4 ...                   | Mouse Button Pressed
NOTE: Left Mouse Button (1) is not reported by default. Use 'EnablePrimaryMouseButtonEvents' to override this. |
| "MOUSE_BUTTON_RELEASED"            | 2 = Mouse Button 2 3 = Mouse Button 3 4 = Mouse Button 4 ...                   | NOTE: Left Mouse Button (1) is not reported by default. Use 'EnablePrimaryMouseButtonEvents' to override this. |

**Example**

```
-- This is the primary event handler
-- You must implement this function

function OnEvent(event, arg)
    if (event == "PROFILE_ACTIVATED") then
        -- profile has been activated
    end

    if (event == "PROFILE_DEACTIVATED") then
        -- profile has been deactivated
    end

    if (event == "G_PRESSED" and arg == 1) then
        -- G1 has been pressed
    end

    if (event == "G_RELEASED" and arg == 1) then
        -- G1 has been released
    end

    if (event == "M_PRESSED" and arg == 1) then
        -- M1 has been pressed
    end

    if (event == "M_RELEASED" and arg == 1) then
        -- M1 has been released
    end

    if (event == "MOUSE_BUTTON_PRESSED" and arg == 6) then
        -- Mouse Button 6 has been pressed
    end

    if (event == "MOUSE_BUTTON_RELEASED" and arg == 6) then
        -- Mouse Button 6 has been released
    end
end
```
GetMKeyState

GetMKeyState() returns the current state of the M keys.

\texttt{mkey GetMKeyState(\texttt{[family]});} \\

Parameters

\texttt{family}

Optional family name of device if you want to distinguish between multiple attached devices. Default is “kb”.

<table>
<thead>
<tr>
<th>Family</th>
<th>Devices</th>
</tr>
</thead>
<tbody>
<tr>
<td>“kb”</td>
<td>Keyboard devices (G15, G11, G19, etc)</td>
</tr>
<tr>
<td>“lhc”</td>
<td>Left handed controllers (G13, etc)</td>
</tr>
</tbody>
</table>

Return Values

\texttt{mkey}

\[1 = M1, 2 = M2, 3 = M3\]

Remarks

Example

\texttt{-- Get the current M Key state}

\texttt{current_mkey = GetMKeyState()}
SetMKeyState

SetMKeyState() sets the current state of the M keys. NOTE: Calling GetMKeyState immediately afterwards, will likely return the previous state. Use the OnEvent handler to determine when the operation has completed.

```
mkey SetMKeyState(mkey, [family]);
```

Parameters

mkey

1 = M1, 2 = M2, 3 = M3

family

Optional family name of device if you want to distinguish between multiple attached devices. Default is "kb".

<table>
<thead>
<tr>
<th>Family</th>
<th>Devices</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;kb&quot;</td>
<td>Keyboard devices (G15, G11, G19, etc)</td>
</tr>
<tr>
<td>&quot;lhc&quot;</td>
<td>Left handed controllers (G13, etc)</td>
</tr>
</tbody>
</table>

Return Values

None

Remarks

Example

```
-- Set the current M Key state to M1 when G1 is pressed

function OnEvent(event, arg)
    if (event == "G_PRESSED" and arg == 1) then
        SetMkeyState(1);
    end
end
```
Sleep

Sleep() will cause the script to pause for the desired amount of time.

Sleep( timeout );

Parameters

timeout

Total time to sleep in milliseconds.

Return Values

nil

Remarks

Scripting runs on separate thread than the main Profiler, so pausing the script will not affect it.

You can use this function to simulate delays.

Deactivation of the profiler will wait 1 second for the script to finish, after which the script will be forcefully aborted. Take precaution if using a long timeout.

Example

-- Sleeping for 20 milliseconds
Sleep(20)
**OutputLogMessage**

*OutputLogMessage()* will send log messages into the script editor.

```
OutputLogMessage( ... );
```

**Parameters**

message

Printf style, formatted string containing the message.

**Return Values**

nil

**Remarks**

Mirror of string.format().

You must manually insert a carriage return "\n" to denote end of line.

**Example**

```bash
-- Send out "Hello World"

OutputLogMessage("Hello World %d\n", 2007)
```
GetRunningTime

GetRunningTime() returns the total number of milliseconds elapsed since the script has been running.

elapsed GetRunningTime();

Parameters

None

Return Values

elapsed

Integer value containing the elapsed time in milliseconds.

Remarks

You can use this to calculate timing in your script.

Example

-- Display the script running time

OutputLogMessage("This script has been running for: %d ms", GetRunningTime())
**GetDate**

Use `GetDate()` to retrieve the formatted date

```lua
date GetDate ([format [, time]])
```

**Parameters**

- `format`
  Optional date format string.

- `time`
  Optional time table.

**Return Values**

- `date`
  A string or a table containing the user's machine's current date and time (or the time represented by time), formatted according to the given string format. If one wishes to supply your own format string, then it uses the same rules as strftime(). The special string *t tells the date() function to return a table.

**Remarks**

Mirror of os.date().

**Example**

```lua
-- Display the current date/time
OutputLogMessage("Today's date/time is: %s\n", GetDate())
```
**ClearLog**

The **ClearLog()** function clears the output window of the script editor.

```
ClearLog()
```

**Parameters**

None.

**Return Values**

nil

**Remarks**

None.

**Example**

```-- Clear the script editor log
OutputLogMessage("This message will self destruct in 2 seconds\n")
Sleep(2000)
ClearLog()
```
**PressKey**

The `PressKey()` function is used to simulate a keyboard key press. NOTE: Calling `IsModifierPressed` or `IsKeyLockOn` immediately afterwards for a simulated modifier or lock key will likely return the previous state. It will take a few milliseconds for the operation to complete.

```c
PressKey( scancode [,scancode] );
PressKey( keyname [,keyname] );
```

**Parameters**

- **scancode**
  
  Specifies the numerical scancode of the key to be pressed.

- **keyname**
  
  Specifies the predefined keyname of the key to be pressed.

**Return Values**

- **nil**

**Remarks**

If multiple keys are provided as arguments, all keys will be simulated with a press.

For scancode and keyname values, refer to Appendix A.

**Example**

```c
-- Simulate "a" pressed using the scancode
PressKey(30)

-- Simulate "a" pressed using the keyname
PressKey("a")

-- Simulate "a" and "b" being pressed
PressKey("a", "b")
```
**ReleaseKey**

The `ReleaseKey()` function is used to simulate a keyboard key release.

```
ReleaseKey( scancode [,scancode] );
ReleaseKey( keyname [,keyname] );
```

**Parameters**

- **scancode**
  Specifies the numerical scancode of the key to be pressed.

- **keyname**
  Specifies the predefined keyname of the key to be pressed.

**Return Values**

- **nil**

**Remarks**

If multiple keys are provided as arguments, all keys will be simulated with a release.

For scancode and keyname values, refer to Appendix A.

**Example**

```-- Simulate "a" released using the scancode
ReleaseKey(30)
-- Simulate "a" released using the keyname
ReleaseKey("a")
-- Simulate "a" and "b" being released
ReleaseKey("a", "b")```
PressAndReleaseKey

The **PressAndReleaseKey**() function is used to simulate a keyboard key press followed by a release. NOTE: Calling **IsModifierPressed** or **IsKeyLockOn** immediately afterwards for a simulated modifier or lock key will likely return the previous state. It will take a few milliseconds for the operation to complete.

```c
ReleaseKey( scancode [,scancode] );
ReleaseKey( keyname [,keyname] );
```

**Parameters**

- **scancode**
  
  Specifies the numerical scancode of the key to be pressed.

- **keyname**
  
  Specifies the predefined keyname of the key to be pressed.

**Return Values**

- **nil**

**Remarks**

If multiple keys are provided as arguments, all keys will be simulated with a press and a release.

For scancode and keyname values, refer to Appendix A.

**Example**

-- Simulate "a" pressed and released using the scancode
PressAndReleaseKey(30)

-- Simulate "a" pressed and released using the keyname
PressAndReleaseKey("a")

-- Simulate "a" and "b" being pressed and released
PressAndReleaseKey("a", "b")
IsModifierPressed

The IsModifierPressed() function is used to determine if a particular modifier key is currently in a pressed state.

```plaintext
boolean IsModifierPressed ( keyname );
```

**Parameters**

- **keyname**

  Specifies the predefined keyname of the modifier key to be pressed. The name must be one of the following:

<table>
<thead>
<tr>
<th>Modifier</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;lalt&quot;, &quot;ralt&quot;, &quot;alt&quot;</td>
<td>Left, right, or either Alt key</td>
</tr>
<tr>
<td>&quot;lshift&quot;, &quot;rshift&quot;, &quot;shift&quot;</td>
<td>Left, right, or either Shift key</td>
</tr>
<tr>
<td>&quot;lctrl&quot;, &quot;rctrl&quot;, &quot;ctrl&quot;</td>
<td>Left, right, or either Ctrl key</td>
</tr>
</tbody>
</table>

**Return Values**

True if the modifier key is currently pressed, false otherwise.

**Remarks**

None.

**Example**

```plaintext
-- Press a specific modifier
PressKey("lshift")

if IsModifierPressed("shift") then
   OutputLogMessage("shift is pressed.
end

-- Release the key so it is no longer pressed
ReleaseKey("lshift")

if not IsModifierPressed("shift") then
   OutputLogMessage("shift is not pressed.
end
```
**PressMouseButton**

The `PressMouseButton()` function is used to simulate a mouse button press. NOTE: Calling `IsMouseButtonPressed` immediately afterwards, will likely return the previous state. It will take a few milliseconds for the operation to complete.

```
PressMouseButton( button )
```

**Parameters**

`button`

Button identifier. Use the following table:

<table>
<thead>
<tr>
<th>Button value</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Left Mouse Button</td>
</tr>
<tr>
<td>2</td>
<td>Middle Mouse Button</td>
</tr>
<tr>
<td>3</td>
<td>Right Mouse Button</td>
</tr>
<tr>
<td>4</td>
<td>X1 Mouse Button</td>
</tr>
<tr>
<td>5</td>
<td>X2 Mouse Button</td>
</tr>
</tbody>
</table>

**Return Values**

nil

**Remarks**

None

**Example**

```
-- Simulate left mouse button press
PressMouseButton(1)

-- Simulate right mouse button press
PressMouseButton(3)
```
**ReleaseMouseButton**

The **ReleaseMouseButton**() function is used to simulate a mouse button release.

```plaintext
ReleaseMouseButton( button )
```

**Parameters**

- `button`

  Button identifier. Use the following table:

<table>
<thead>
<tr>
<th>Button value</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Left Mouse Button</td>
</tr>
<tr>
<td>2</td>
<td>Middle Mouse Button</td>
</tr>
<tr>
<td>3</td>
<td>Right Mouse Button</td>
</tr>
<tr>
<td>4</td>
<td>X1 Mouse Button</td>
</tr>
<tr>
<td>5</td>
<td>X2 Mouse Button</td>
</tr>
</tbody>
</table>

**Return Values**

- `nil`

**Remarks**

None

**Example**

```plaintext
-- Simulate a left mouse button click (press and release)

PressMouseButton(1)
ReleaseMouseButton(1)
```
PressAndReleaseMouseButton

The `PressAndReleaseMouseButton()` function is used to simulate a mouse button press followed by a release. NOTE: Calling `IsMouseButtonPressed` immediately afterwards, will likely return the previous state. It will take a few milliseconds for the operation to complete.

```c
PressAndReleaseMouseButton( button )
```

### Parameters

**button**

Button identifier. Use the following table:

<table>
<thead>
<tr>
<th>Button value</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Left Mouse Button</td>
</tr>
<tr>
<td>2</td>
<td>Middle Mouse Button</td>
</tr>
<tr>
<td>3</td>
<td>Right Mouse Button</td>
</tr>
<tr>
<td>4</td>
<td>X1 Mouse Button</td>
</tr>
<tr>
<td>5</td>
<td>X2 Mouse Button</td>
</tr>
</tbody>
</table>

### Return Values

`nil`

### Remarks

None

### Example

```c
-- Simulate a left mouse button click (press and release)
PressAndReleaseMouseButton(1)
```
IsMouseButtonPressed

The IsMouseButtonPressed() function is used to determine if a particular mouse button is currently in a pressed state.

```java
boolean IsMouseButtonPressed( button )
```

**Parameters**

- **button**

  Button identifier. Use the following table:

<table>
<thead>
<tr>
<th>Button value</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Left Mouse Button</td>
</tr>
<tr>
<td>2</td>
<td>Middle Mouse Button</td>
</tr>
<tr>
<td>3</td>
<td>Right Mouse Button</td>
</tr>
<tr>
<td>4</td>
<td>X1 Mouse Button</td>
</tr>
<tr>
<td>5</td>
<td>X2 Mouse Button</td>
</tr>
</tbody>
</table>

**Return Values**

True if the button is currently pressed, false otherwise.

**Remarks**

None

**Example**

```java
-- Press a mouse button
PressMouseButton(1)

if IsMouseButtonPressed(1) then
  OutputLogMessage("Left mouse button is pressed.\n");
end

-- Release the button so it is no longer pressed
ReleaseMouseButton(1)

if not IsMouseButtonPressed(1) then
  OutputLogMessage("Left mouse button is not pressed.\n");
```
**MoveMouseTo**

The `MoveMouseTo()` function is used to move the mouse cursor to an absolute position on the screen. NOTE: Calling `GetMousePosition` immediately afterwards, will likely return the previous state. It will take a few milliseconds for the operation to complete.

```plaintext
MoveMouseTo( x, y, )
```

**Parameters**

- **x**
  Normalized X coordinate between 0 (farthest left) and 65535 (farthest right)

- **y**
  Normalized y coordinate between 0 (farthest top) and 65535 (farthest bottom)

**Return Values**

- **nil**

**Remarks**

If multiple monitors are present, use `MoveMouseToVirtual`.

**Example**

```plaintext
-- Move mouse to upper, left corner
MoveMouseTo(0, 0)

-- Move mouse to center of screen
MoveMouseTo(32767, 32767)

-- Move mouse to lower, right corner
MoveMouseTo(65535, 65535)
```
**MoveMouseWheel**

The **MoveMouseWheel()** function is used to simulate mouse wheel movement.

<table>
<thead>
<tr>
<th>MoveMouseWheel( click )</th>
</tr>
</thead>
</table>

**Parameters**

- **click**
  
  Number of mouse wheel clicks.

**Return Values**

- **nil**

**Remarks**

  Positive values denote wheel movement upwards (away from user).
  
  Negative values denote wheel movement downwards (towards user).

**Example**

```plaintext
-- Simulate mouse wheel 3 clicks up
MoveMouseWheel(3)

-- Simulate mouse wheel 1 click down
MoveMouseWheel(-1)
```
MoveMouseRelative

The MoveMouseRelative() function is used to simulate relative mouse movement. NOTE: Calling GetMousePosition immediately afterwards, will likely return the previous state. It will take a few milliseconds for the operation to complete.

MoveMouseRelative( x, y, )

Parameters

x

Movement along the x-axis

Y

Movement along the y-axis

Return Values

nil

Remarks

Positive x values simulate movement to right.

Negative x values simulate movement to left.

Positive y values simulate movement downwards.

Negative y values simulate movement upwards.

Example

-- Simulate relative mouse movement upwards in 1 pixel increments
for i = 0, 50 do
    MoveMouseRelative(0, -1)
    Sleep(8)
end
**MoveMouseToVirtual**

The `MoveMouseToVirtual()` function is used to move the mouse cursor to an absolute position on a multi-monitor screen layout. NOTE: Calling `GetMousePosition` immediately afterwards, will likely return the previous state. It will take a few milliseconds for the operation to complete.

**MoveMouseToVirtual( x, y, )**

**Parameters**

- **x**
  
  Normalized X coordinate between 0 (farthest left) and 65535 (farthest right)

- **y**
  
  Normalized y coordinate between 0 (farthest top) and 65535 (farthest bottom)

**Return Values**

- nil

**Remarks**

If multiple monitors are present, use MoveMouseToVirtual.

**Example**

---

- Move mouse to upper, left corner of virtual desktop
  MoveMouseToVirtual(0, 0)

- Move mouse to center of virtual desktop
  MoveMouseToVirtual (32767, 32767)

- Move mouse to lower, right corner of virtual desktop
  MoveMouseToVirtual (65535, 65535)
GetMousePosition

The `GetMousePosition()` function returns the normalized coordinates of the current mouse cursor location.

```plaintext
x, y  GetMousePosition()
```

**Parameters**

None

**Return Values**

- **x**
  
  Normalized X coordinate between 0 (farthest left) and 65535 (farthest right)

- **y**
  
  Normalized y coordinate between 0 (farthest top) and 65535 (farthest bottom)

**Remarks**

**Example**

```plaintext
-- Get the current mouse cursor position
x, y = GetMousePosition();

OutputLogMessage("Mouse is at %d, %d\n", x, y);
```
OutputLCDMessage

The **OutputLCDMessage**() function is used to add a line of text on to the LCD.

```markdown
OutputLCDMessage( text [,timeout] )
```

**Parameters**

- `text`
  String to display

- `timeout`
  Timeout in milliseconds, after which the message will disappear

**Return Values**

- `nil`

**Remarks**

Up to 4 lines of text can be displayed at once. The default timeout is 1 second.

**Example**

```markdown
-- Display some text with default timeout
OutputLCDMessage("Hello world")

-- Display some text for 2 seconds
OutputLCDMessage("Hello world", 2000)
```
ClearLCD

The **ClearLCD()** function clears the script display on the LCD.

`ClearLCD()`

**Parameters**

none

**Return Values**

nil

**Remarks**

**Example**

```plaintext
-- Clear the LCD and then display 2 lines of text
ClearLCD()
OutputLCDMessage("Hello world1")
OutputLCDMessage("Hello world2")
```
**PlayMacro**

The **PlayMacro** () function is used to play an existing macro.

```
PlayMacro( macroname )
```

**Parameters**

- **macroname**
  
  Name of existing macro belonging to the current profile.

**Return Values**

- **nil**

**Remarks**

If the function is called while another script macro is playing, no action is taken. In other words, only one script macro may be playing at any given time.

If the function is called while the same script macro is playing, the macro is queued.

**Example**

```
-- Play an existing macro
PlayMacro("my macro");
```
AbortMacro

The **AbortMacro()** function is used to abort any macro started from a script.

### Parameters

None

### Return Values

nil

### Remarks

Any keys still pressed after a call to PlayMacro will be released. Macros playing outside the script will continue to play.

### Example

```plaintext
-- Start a macro
PlayMacro("my macro")

-- Wait for 100ms and then abort any playing macro
AbortMacro()
```
**IsKeyLockOn**

The `IsKeyLockOn()` function used to determine if a particular lock button is currently in an enabled state.

`IsKeyLockOn( key )`

**Parameters**

- **key**

  key name. Use the following table:

<table>
<thead>
<tr>
<th>Key name</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;scrolllock&quot;</td>
<td>Scroll Lock</td>
</tr>
<tr>
<td>&quot;capslock&quot;</td>
<td>Caps Lock</td>
</tr>
<tr>
<td>&quot;numlock&quot;</td>
<td>Number Lock</td>
</tr>
</tbody>
</table>

**Return Values**

True if the lock is currently enabled, false otherwise.

**Remarks**

None.

**Example**

```-- Check if the numlock is on and turn it off if it is
if ( IsKeyLockOn("numlock") ) then
    PressAndReleaseKey("numlock");
end```
SetBacklightColor

The `SetBacklightColor()` function is used to set the custom backlight color of the device (if the device supports custom backlighting).

`SetBacklightColor(red, green, blue, [family])`

Parameters

- **red**
  - Red intensity (0 – 255)
- **green**
  - Green intensity (0 – 255)
- **blue**
  - Blue intensity (0 – 255)
- **family**
  - Optional family name of device if you want to distinguish between multiple attached devices. Default is “kb”.

<table>
<thead>
<tr>
<th>Family</th>
<th>Devices</th>
</tr>
</thead>
<tbody>
<tr>
<td>“kb”</td>
<td>Keyboard devices (G15, G11, G19, etc)</td>
</tr>
<tr>
<td>“lhc”</td>
<td>Left handed controllers (G13, etc)</td>
</tr>
</tbody>
</table>

Return Values

- nil

Remarks

None.

Example

```bash
-- Set the backlight to red
SetBacklightColor(255, 0, 0);

-- Set the backlight color for all left handed controllers to blue
SetBacklightColor(0, 0, 255, “lhc”);
```
**OutputDebugMessage**

*OutputDebugMessage()* will send log messages to the Windows debugger.

```plaintext
OutputDebugMessage( ... );
```

**Parameters**

**Message**

Printf style, formatted string containing the message.

**Return Values**

nil

**Remarks**

Mirror of string.format().

You must manually insert a carriage return "\n" to denote end of line.

Use tools like Dbg View for viewing these messages.

**Example**

-- Send out "Hello World"

```plaintext
OutputDebugMessage("Hello World %d\n", 2007)
```
SetMouseDPITable

SetMouseDPITable() sets the current DPI table for a supported gaming mouse

SetMouseDPITable({value1, value2, value3}, [index]);

Parameters

DPI Array

Array of DPI values

DPI Index

Optional 1-Based index to DPI to apply as the current DPI.

Return Values

nil

Remarks

If the index is not specified, the first entry is used as the current DPI

A maximum of 16 entries are allowed.

Activating a profile with per-profile DPI settings will override any previously applied DPI.

Example

-- Set our DPI values to {500, 1000, 1500, 2000, 2500}
-- By default, 500 DPI will be set as the current DPI
SetMouseDPITable({500, 1000, 1500, 2000, 2500})

-- Set our DPI values to {500, 2500} and set the second value as the current DPI
SetMouseDPITable({500, 2500}, 2)
SetMouseDPITableIndex

SetMouseDPITableIndex() sets the current DPI table index for a supported gaming mouse

SetMouseDPITableIndex(index);

Parameters

Index

1-Based index into the DPI Table

Return Values

nil

Remarks

If SetMouseDPITable was not called prior to this, the current DPI table for the mouse is used.

A maximum of 16 entries are allowed.

Activating a profile with per-profile DPI settings will override any previously applied DPI.

Example

-- Set our initial DPI values to {500, 1000, 1500, 2000, 2500}  
SetMouseDPITable({500, 1000, 1500, 2000, 2500})

-- Set the current DPI to the 3rd item in the table (1500 DPI)  
SetMouseDPITableIndex(3);
EnablePrimaryMouseButtonEvents

EnablePrimaryMouseButtonEvents() enables event reporting for mouse button 1.

EnablePrimaryMouseButtonEvents(enable);

Parameters

enable

1 or true to enable event reporting for mouse button 1

0 or false to disable event reporting for mouse button 1

Return Values

nil

Remarks

The primary mouse button is not reported by default for performance issues.

Example

-- Enable event reporting for mouse button 1
EnablePrimaryMouseButtonEvents(true);

-- Disable event reporting for mouse button 1
EnablePrimaryMouseButtonEvents(false);
SetSteeringWheelProperty

SetSteeringWheelProperty() sets a steering wheel property.

SetSteeringWheelProperty(device, property, value);

Parameters

device

<table>
<thead>
<tr>
<th>Device</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“G29”</td>
<td>Logitech G29 Steering Wheel</td>
</tr>
<tr>
<td>“G920”</td>
<td>Logitech G920 Steering Wheel</td>
</tr>
</tbody>
</table>

property

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>“operatingRange”</td>
<td>Operating range of wheel from 40 to 900. Default is 900.</td>
</tr>
<tr>
<td>“combinedPedals”</td>
<td>Combines the brake and accelerator into a single axis. The accelerator is on the + axis, and the brake is on the – axis. Default is false.</td>
</tr>
<tr>
<td>“defaultCenteringSpring”</td>
<td>Plays a persistent spring on top of any game forces. Default is false.</td>
</tr>
<tr>
<td>“defaultCenteringSpringStrength”</td>
<td>Sets the strength of the default centering spring from 0-100.</td>
</tr>
</tbody>
</table>

Return Values

nil

Remarks

Changing the value does not affect the

Example

-- Set the operating range to 200 degrees for the G29
SetSteeringWheelProperty("G29", "operatingRange", 200);

-- Enable combined pedals on the G920
SetSteeringWheelProperty("G920", "combinedPedals", true);
G13 Programming

The G13 game panel has an analog joystick that can have a mouse function assigned to it. The speed of the mouse can be adjusted through either the profiler options panel in the settings window, or through the Lua scripting language. The following are the new Lua functions for mouse speed control:

**SetMouseSpeed()**

**Parameters**

- **New mouse speed**
  
  Absolute mouse speed 32 to 255.

**Return Values**

- nil

**Remarks**

- none

**Example**

```lua
--Set Mouse speed to 128
SetMouseSpeed(128);
```

**GetMouseSpeed()**

**Parameters**

- **Current mouse speed**
  
  Absolute mouse speed 32 to 255.

**Return Values**

- Current emulated mouse speed.

**Remarks**

- none

**Example**

```lua
--Get Mouse speed
OutputLogMessage("The Mouse Speeed is: %d\n", GetMouseSpeed());
```
**IncrementMouseSpeed()**

**Parameters**

Mouse speed increment

**Return Values**

nil

**Remarks**

Resultant mouse speed will be clamped to a maximum of 255

**Example**

```
--Increase Mouse speed by 10
IncrementMouseSpeed(10);
```

---

**DecrementMouseSpeed()**

**Parameters**

Mouse speed decrement

**Return Values**

nil

**Remarks**

Resultant mouse speed will be clamped to a minimum of 32

**Example**

```
-- Decrease Mouse speed by 10
DecrementMouseSpeed(10);
```
The G13 mouse functionality does not support any native buttons, e.g. left button, center button, etc. Mouse buttons must be programmed via Lua. Here is an example of generic Lua code to effect mouse button operation:

```lua
if event=="G_PRESSED" and arg==x then
    PressMouseButton( y );
end

if event=="G_RELEASED" and arg==x then
    ReleaseMouseButton( y );
```
Standard Lua 5.1 Libraries

The following standard library functions are supported:

- string.byte
- string.char
- string.dump
- string.find
- string.format
- string.gmatch
- string.gsub
- string.len
- string.lower
- string.match
- string.rep
- string.reverse
- string.sub
- string.upper
- table.concat
- table.insert
- table.maxn
- table.remove
- table.sort
- math.abs
- math.acos
- math.asin
- math.atan
- math.atan2
- math.ceil
- math.cos
- math.deg
- math.exp
- math.floor
- math.fmod
- math.frexp
- math.huge
- math.ldexp
- math.log
- math.log10
- math.max
- math.min
- math.modf
- math.pi
- math.pow
- math.rad
- math.random
- math.randomseed
- math.sin
- math.sinh
- math.sqrt
- math.tan
- math.tanh
Appendix A

Table of scancodes and keynames used in `PressKey()`, `ReleaseKey()`, `IsModifierPressed()`.

<table>
<thead>
<tr>
<th>Keyname</th>
<th>Scancode (hex)</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;escape&quot;</td>
<td>0x01</td>
</tr>
<tr>
<td>&quot;f1&quot;</td>
<td>0x3b</td>
</tr>
<tr>
<td>&quot;f2&quot;</td>
<td>0x3c</td>
</tr>
<tr>
<td>&quot;f3&quot;</td>
<td>0x3d</td>
</tr>
<tr>
<td>&quot;f4&quot;</td>
<td>0x3e</td>
</tr>
<tr>
<td>&quot;f5&quot;</td>
<td>0x3f</td>
</tr>
<tr>
<td>&quot;f6&quot;</td>
<td>0x40</td>
</tr>
<tr>
<td>&quot;f7&quot;</td>
<td>0x41</td>
</tr>
<tr>
<td>&quot;f8&quot;</td>
<td>0x42</td>
</tr>
<tr>
<td>&quot;f9&quot;</td>
<td>0x43</td>
</tr>
<tr>
<td>&quot;f10&quot;</td>
<td>0x44</td>
</tr>
<tr>
<td>&quot;f11&quot;</td>
<td>0x57</td>
</tr>
<tr>
<td>&quot;f12&quot;</td>
<td>0x58</td>
</tr>
<tr>
<td>&quot;f13&quot;</td>
<td>0x64</td>
</tr>
<tr>
<td>&quot;f14&quot;</td>
<td>0x65</td>
</tr>
<tr>
<td>&quot;f15&quot;</td>
<td>0x66</td>
</tr>
<tr>
<td>&quot;f16&quot;</td>
<td>0x67</td>
</tr>
<tr>
<td>&quot;f17&quot;</td>
<td>0x68</td>
</tr>
<tr>
<td>&quot;f18&quot;</td>
<td>0x69</td>
</tr>
<tr>
<td>&quot;f19&quot;</td>
<td>0x6a</td>
</tr>
<tr>
<td>&quot;f20&quot;</td>
<td>0x6b</td>
</tr>
<tr>
<td>&quot;f21&quot;</td>
<td>0x6c</td>
</tr>
<tr>
<td>&quot;f22&quot;</td>
<td>0x6d</td>
</tr>
<tr>
<td>&quot;f23&quot;</td>
<td>0x6e</td>
</tr>
<tr>
<td>&quot;f24&quot;</td>
<td>0x76</td>
</tr>
<tr>
<td>&quot;printscreen&quot;</td>
<td>0x137</td>
</tr>
<tr>
<td>&quot;scrolllock&quot;</td>
<td>0x46</td>
</tr>
<tr>
<td>&quot;pause&quot;</td>
<td>0x146</td>
</tr>
<tr>
<td>&quot;tilde&quot;</td>
<td>0x29</td>
</tr>
<tr>
<td>&quot;1&quot;</td>
<td>0x02</td>
</tr>
<tr>
<td>&quot;2&quot;</td>
<td>0x03</td>
</tr>
<tr>
<td>&quot;3&quot;</td>
<td>0x04</td>
</tr>
<tr>
<td>&quot;4&quot;</td>
<td>0x05</td>
</tr>
<tr>
<td>&quot;5&quot;</td>
<td>0x06</td>
</tr>
<tr>
<td>&quot;6&quot;</td>
<td>0x07</td>
</tr>
<tr>
<td>&quot;7&quot;</td>
<td>0x08</td>
</tr>
<tr>
<td>&quot;8&quot;</td>
<td>0x09</td>
</tr>
<tr>
<td>&quot;9&quot;</td>
<td>0x0a</td>
</tr>
<tr>
<td>&quot;0&quot;</td>
<td>0x0b</td>
</tr>
<tr>
<td>&quot;minus&quot;</td>
<td>0x0c</td>
</tr>
<tr>
<td>&quot;equal&quot;</td>
<td>0x0d</td>
</tr>
<tr>
<td>&quot;backspace&quot;</td>
<td>0x0e</td>
</tr>
<tr>
<td>&quot;tab&quot;</td>
<td>0x0f</td>
</tr>
<tr>
<td>&quot;q&quot;</td>
<td>0x10</td>
</tr>
<tr>
<td>&quot;w&quot;</td>
<td>0x11</td>
</tr>
<tr>
<td>&quot;e&quot;</td>
<td>0x12</td>
</tr>
<tr>
<td>&quot;r&quot;</td>
<td>0x13</td>
</tr>
<tr>
<td>&quot;t&quot;</td>
<td>0x14</td>
</tr>
<tr>
<td>&quot;y&quot;</td>
<td>0x15</td>
</tr>
<tr>
<td>Unicode Character</td>
<td>ASCII Code</td>
</tr>
<tr>
<td>-------------------</td>
<td>------------</td>
</tr>
<tr>
<td>&quot;u&quot;</td>
<td>0x16</td>
</tr>
<tr>
<td>&quot;i&quot;</td>
<td>0x17</td>
</tr>
<tr>
<td>&quot;o&quot;</td>
<td>0x18</td>
</tr>
<tr>
<td>&quot;p&quot;</td>
<td>0x19</td>
</tr>
<tr>
<td>&quot;lbracket&quot;</td>
<td>0x1a</td>
</tr>
<tr>
<td>&quot;rbracket&quot;</td>
<td>0x1b</td>
</tr>
<tr>
<td>&quot;backslash&quot;</td>
<td>0x2b</td>
</tr>
<tr>
<td>&quot;capslock&quot;</td>
<td>0x3a</td>
</tr>
<tr>
<td>&quot;a&quot;</td>
<td>0x1e</td>
</tr>
<tr>
<td>&quot;s&quot;</td>
<td>0x1f</td>
</tr>
<tr>
<td>&quot;d&quot;</td>
<td>0x20</td>
</tr>
<tr>
<td>&quot;n&quot;</td>
<td>0x21</td>
</tr>
<tr>
<td>&quot;g&quot;</td>
<td>0x22</td>
</tr>
<tr>
<td>&quot;h&quot;</td>
<td>0x23</td>
</tr>
<tr>
<td>&quot;j&quot;</td>
<td>0x24</td>
</tr>
<tr>
<td>&quot;k&quot;</td>
<td>0x25</td>
</tr>
<tr>
<td>&quot;l&quot;</td>
<td>0x26</td>
</tr>
<tr>
<td>&quot;semicolon&quot;</td>
<td>0x27</td>
</tr>
<tr>
<td>&quot;quote&quot;</td>
<td>0x28</td>
</tr>
<tr>
<td>&quot;enter&quot;</td>
<td>0x1c</td>
</tr>
<tr>
<td>&quot;lshift&quot;</td>
<td>0x2a</td>
</tr>
<tr>
<td>&quot;non_us_slash&quot;</td>
<td>0x56</td>
</tr>
<tr>
<td>&quot;x&quot;</td>
<td>0x2c</td>
</tr>
<tr>
<td>&quot;x&quot;</td>
<td>0x2d</td>
</tr>
<tr>
<td>&quot;c&quot;</td>
<td>0x2e</td>
</tr>
<tr>
<td>&quot;v&quot;</td>
<td>0x2f</td>
</tr>
<tr>
<td>&quot;b&quot;</td>
<td>0x30</td>
</tr>
<tr>
<td>&quot;n&quot;</td>
<td>0x31</td>
</tr>
<tr>
<td>&quot;m&quot;</td>
<td>0x32</td>
</tr>
<tr>
<td>&quot;comma&quot;</td>
<td>0x33</td>
</tr>
<tr>
<td>&quot;period&quot;</td>
<td>0x34</td>
</tr>
<tr>
<td>&quot;slash&quot;</td>
<td>0x35</td>
</tr>
<tr>
<td>&quot;rshift&quot;</td>
<td>0x36</td>
</tr>
<tr>
<td>&quot;lctr1&quot;</td>
<td>0x1d</td>
</tr>
<tr>
<td>&quot;lgu&quot;</td>
<td>0x15b</td>
</tr>
<tr>
<td>&quot;lalt&quot;</td>
<td>0x38</td>
</tr>
<tr>
<td>&quot;spacebar&quot;</td>
<td>0x39</td>
</tr>
<tr>
<td>&quot;ralt&quot;</td>
<td>0x138</td>
</tr>
<tr>
<td>&quot;rgui&quot;</td>
<td>0x15c</td>
</tr>
<tr>
<td>&quot;appkey&quot;</td>
<td>0x15d</td>
</tr>
<tr>
<td>&quot;rctrl&quot;</td>
<td>0x11d</td>
</tr>
<tr>
<td>&quot;insert&quot;</td>
<td>0x152</td>
</tr>
<tr>
<td>&quot;home&quot;</td>
<td>0x147</td>
</tr>
<tr>
<td>&quot;pageup&quot;</td>
<td>0x149</td>
</tr>
<tr>
<td>&quot;delete&quot;</td>
<td>0x153</td>
</tr>
<tr>
<td>&quot;end&quot;</td>
<td>0x14f</td>
</tr>
<tr>
<td>&quot;pagedown&quot;</td>
<td>0x151</td>
</tr>
<tr>
<td>&quot;up&quot;</td>
<td>0x148</td>
</tr>
<tr>
<td>&quot;left&quot;</td>
<td>0x14b</td>
</tr>
<tr>
<td>&quot;down&quot;</td>
<td>0x150</td>
</tr>
<tr>
<td>&quot;right&quot;</td>
<td>0x14d</td>
</tr>
<tr>
<td>&quot;numlock&quot;</td>
<td>0x45</td>
</tr>
<tr>
<td>&quot;numslash&quot;</td>
<td>0x135</td>
</tr>
<tr>
<td>Numeral</td>
<td>ASCII Value</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>&quot;numminus&quot;</td>
<td>0x4a</td>
</tr>
<tr>
<td>&quot;num7&quot;</td>
<td>0x47</td>
</tr>
<tr>
<td>&quot;num8&quot;</td>
<td>0x48</td>
</tr>
<tr>
<td>&quot;num9&quot;</td>
<td>0x49</td>
</tr>
<tr>
<td>&quot;numplus&quot;</td>
<td>0x4e</td>
</tr>
<tr>
<td>&quot;num4&quot;</td>
<td>0x4b</td>
</tr>
<tr>
<td>&quot;num5&quot;</td>
<td>0x4c</td>
</tr>
<tr>
<td>&quot;num6&quot;</td>
<td>0x4d</td>
</tr>
<tr>
<td>&quot;num1&quot;</td>
<td>0x4f</td>
</tr>
<tr>
<td>&quot;num2&quot;</td>
<td>0x50</td>
</tr>
<tr>
<td>&quot;num3&quot;</td>
<td>0x51</td>
</tr>
<tr>
<td>&quot;numenter&quot;</td>
<td>0x11c</td>
</tr>
<tr>
<td>&quot;num0&quot;</td>
<td>0x52</td>
</tr>
<tr>
<td>&quot;numperiod&quot;</td>
<td>0x53</td>
</tr>
</tbody>
</table>