MagicLUA*

* a.k.a. The Ultimate Camera Bricking Tool from Magic Lantern

Scripting Guide

http://magiclantern.wikia.com/MagicLUA

January 6, 2011

Contents

Magic Lantern 0.2 for Canon 550D, Firmware 1.0.9 LUA Scripting Guide

WARNING

```
************************************************
*                                              *
*  WITH GREAT POWER COMES GREAT RESPONSABILITY. *
*                                              *
*  BAD SCRIPTS CAN DAMAGE YOUR CAMERA EASILY    *
*                                              *
*  DO NOT RUN ANY SCRIPT ON YOUR CAMERA         *
*  UNLESS YOU FULLY UNDERSTAND HOW IT WORKS!!!  *
*                                              *
************************************************
```

See the LUA CHDK page for more info.

Overview

Scripting is a very powerful tool which allows you to customize your camera without requiring you to have low-level programming skills or special development environments which are hard to setup.

Scripting support on Magic Lantern is in **very early stages of development**. What does this mean?

- It is easier than ever to damage your camera, either with poorly written scripts or with bugs in the scripting engine.
- You should **NOT** run any script unless you fully understand what it does and how it works.
- You should **NOT** run any of the script-enabled ML builds if you are not confortable reading this.

To repeat this important point:

```
************************************************
*                                              *
* THIS IS DANGEROUS AND MIGHT DAMAGE YOUR CAMERA. *
*   IF IT BREAKS, YOU GET TO KEEP BOTH PIECES.  *
*                                              *
************************************************
```

The scripting engine available for Magic Lantern is called **MagicLUA**. It is a (slightly) stripped down version of LUA, with an API which tries to expose the most important functionality of Magic Lantern.

Features

- Script console for standard output
- take photos, start/stop movies
- Can run some simple scripts like intervalometer, lens info or rack focus
- Can easily crash your camera (or maybe worse)

Limitations

- Integer math (we may switch to double if all goes well)
- No standard input
- No file I/O (work in progress)
- Standard ML modules are not enabled due to stability problems

Running LUA scripts

Running scripts on the camera

User scripts should be placed under the `scripts` directory on your SD card, should have short 8.3 file names and should end in `.lua`. They will be auto-detected at camera startup. At most 7 scripts can be present (because more scripts won't fit in the ML menu).

Subfolders of `scripts` will contain libraries.

Go to the `Script` menu, select a script and:

- press `DISP` to show the source code
- press `SET` to run the script

Testing LUA scripts on the PC

It is usually faster to find bugs in your scripts if you test them on the PC. You have the following options:

- Use standard LUA; ML has a library which provides dummy functions for testing.
- Use MagicLUA standalone interpreter, which is compiled from the same source code as the scripting engine which runs on the camera. This shares (almost) all the limitations of the in-camera LUA (like integer math).

To test a script, make sure the required executables are in your `PATH` and use one of these commands:

```
magiclua interv.lua
```

or:

```
lua -lmagic/magic interv.lua
```

To compile MagicLUA standalone interpreter, go to `lua` directory in ML source code and type:

```
make magiclua
```

LUA introduction

Lightweight scripting language.

Hello World:

```
print("Hello, World!")
```

Comments:

```
-- this is a comment
-- [[ this is a
      big comment ]]
```

Everything is a table (well... almost).

Control structures:

```
if 1==2 then print "eq" else print "ne" end

for i = 1,5 do
    print(i)
end

while(1)
    print("tick")
    msleep(1000)
end

repeat
    print("tick")
    msleep(1000)
until 1==2
```

Tables:

```
a = {10,11,12}
b = {x=10, y=11, z=12}
print(a[1], b["x"], b.x)
print(#a)
for k,v in pairs(b) do
    print(k,v)
end
```

Magic Lantern LUA scripting API

Global functions

msleep(ms) Pauses for `ms` miliseconds and allows other tasks to run.

shoot() Takes a picture.

cls() Clears the script console.

call(funcname, arg) Calls an eventproc (a function from the camera firmware which can be called by name). See Eventprocs. Dangerous.

Some small helpers

printf(fmt, ...) Similar to `printf` from C. Shorthand for `print(string.format(fmt, ...))`.

dir(obj) Similar to `dir` from Python. Returns a string with the items of a table.

lens

See also `lens.h` and `lens.c` from Magic Lantern API.

lens.take_picture(wait_ms) Takes a picture and waits max. `wait_ms` milisec-
onds for the operation to complete. Returns last job state. See `PROP_LAST_JOB_STATE` in `property.h`.

lens.focus(mode,step) Send a focus command. Will block if it is not safe to
send the focus command. Look in ML source code for usage examples.

lens.focus_start(dir) Start the lens focus task; `dir` is +/- 1.

lens.focus_stop() Stop the lens focus task.

lens.format_dist(d) Format a distance in mm into something useful.

lens.iso [**get/set**] ISO value. Range: 100 ... 25600; 0 = ISO Auto.

lens.shutter [**get/set**] Shutter speed (1/x). Range: 30...4000. Long expo-
sures are not supported.

lens.aperture [**get/set**] Aperture value x 10. E.g. for F1.4, the value is 14.

lens.ae [**get/set**] Exposure compensation, in 1/8 EV steps, signed.

lens.wb_mode [**get/set**] White balance mode (see `PROP_WB_MODE` from `property.h`).

lens.wb_kelvin [**get/set**] Kelvin temperature for white balance. Setting this
also changes `wb_mode` to WB_KELVIN.

lens.picstyle [**get**] Current picture style index (1...9).

lens.contrast [**get/set**] Contrast from current picture style.

lens.sharpness [**get/set**] Sharpness from current picture style.

lens.saturation [**get/set**] Saturation from current picture style.

lens.color_tone [**get/set**] Color tone from current picture style.

lens.name Lens name (string).

lens.focal_len [**get**] Lens focal length (mm).

lens.focus_dist [**get**] Lens focus distance (cm).

lens.hyperfocal [**get**] Hyperfocal distance, in mm. See Circle of Confusion on
Wikipedia. For computations, CoC = 0.029 mm.

lens.dof_near [**get**] In mm.

lens.dof_far [**get**] In mm.

lens.job_state [get] See `PROP_LAST_JOB_STATE`.

lens.raw_aperture [get/set] In 1/8 EV steps?

lens.raw_shutter [get/set] In 1/8 EV steps.

lens.raw_iso [get/set] In 1/8 EV steps; 0 = Auto ISO.

lens.raw_picstyle [get/set] See `PROP_PICTURE_STYLE`.

movie

movie.start() Start movie recording. Calls `MovieStart` eventproc only if camera is in Movie mode and not recording.

movie.end() Stop movie recording. Calls `MovieEnd` eventproc only if camera is in Movie mode and is recording.

prop

Access to properties.

prop[p] Get/set the value of a property. See `property.h` and Properties.

`WARNING!` Setting incorrect values to properties may be dangerous. Many (all?) properties are stored in non-volatile memory, and setting incorrect values may result in strange camera behavior and/or impossibility to restore correct values.

cfn

Access to custom functions (CFn).

cfn[i] Get/set the value of custom function with number `i` (from 0 to 13 in 550D).

audio

level [get] Current audio level

ic_read(cmd) Read a value from the audio chip.

ic_write(cmd) Write a value to the audio chip.