Table of Contents

7.17 temp <sensor id>....20
7.18 max_temp....20
7.19 max_temp_reset....20
7.20 over_temp_count....21
8 Camera Commands....22
8.1 cam_gain <gain>....22
8.2 cam_exposure <time>....22
8.3 cam_info....22
9 Video Commands....23
9.1 video_mode <mode>....23
9.2 sdi_black <offset>....23
9.3 sdi_white <offset>....23
9.4 sdi_range <flag>....24
9.5 post_bright <offset>....24
9.6 post_cont <factor>....24
9.7 post_sat <factor>....25
9.8 post_hue <offset>....25
9.9 wb....25
9.10 awb <flag>....25
9.11 awb_speed <speed>....26
9.12 wb_preset <id>....26
9.13 gain_red <gain>....26
9.14 gain_blue <gain>....26
9.15 gain_green <gain>....27
9.16 black_master <offset-red> <offset-green> <offset-blue>....27
9.17 black_red <offset>....27
9.18 black_blue <offset>....27
9.19 black_green <offset>....27
9.20 flare <red level> <green level> <blue level>....27
9.21 filter_enable <flag>....28
9.22 filter_detail <level>....28
9.23 filter_denoise <level>....28
9.24 color_conv <c0> .. <c8>....28
9.25 color_cross <c0> .. <c8>....29
9.26 color_cross_offset <red_offset> <green_offset> <blue_offset>....30
9.27 color_space<color space>....30
9.28 mcc <flag>....31
9.29 mcc_opmode <flag>....31
9.30 mcc_blink <mask> [<period_ms>]....31
9.31 mcc_set [<saturation>] [<hue>]....31
9.32 lsc <enable> <k> <offset> <slope>....33
9.33 cnr_auto <flag>....34
10 Defect Pixel Correction....36
10.1 dpc <flag>....36
10.2 dpc_test_mode <mode>....36
10.3 dpc_auto_load....36
11 Knee Function....37

1 Overview

This manual describes the usage of the ATOM one mini camera system.

The general operation of the device is described in chapter 2. Information on firmware updates is found in chapter 3.

We provide an open-source, free of charge GUI application which gives easy access to most camera functions, for more details on the GUI see chapter 4.

Chapter 5 and 6 describe how to connect with the device and how the settings on the device are handled using terminal commands.

For details on the camera functions, please refer to the API chapters of this manual which start with chapter 7.

2 General Operation

For regular operation the camera needs only a power supply. For controlling and updating the camera a RS485 connection is required.

This can be done with the given RS485 USB converter. Connect the USB connector to your PC and let the operation system install the required drivers.

For RS485 an address is required to access the camera. The default address is “1”.

After powering up the camera the camera will boot. The boot process takes ~3 seconds.

2.1 Connectors

The ATOM one has one 6 pin Hirose HR10 connector for Power and RS485 communication.

2.1.1 Power and RS485 Connector

The power connector is male and will mate with a female hirose HR10A-7P-6S(73) .

Note: The pin order of HR10A of male and female is counted in different directions.

The following table shows the numbering and cable colors of used Dream Chip adapters.

HR10A-7P-6S(73)	Signal	Cable 6 wire	Cable 4 wire	USB RS485
6	Power in 6-36 Volt	Red	Red	-
5	GND	Brown	Brown	Black (gnd)
4	RS485_TX_P	Black	Looped to pin 1	
3	RS485_TX_N	Green	Looped to pin 2	
2	RS485_RX_N	Yellow	Black	Yellow (D-)
1	RS485_RX_P	Orange	Orange	Orange (D+)

2.2 Over-Temperature Protection

The ATOM one mini has a very compact design which does not allow for active cooling. Thus the device will get hot under operation. To protect the device the camera will go into a safe operational mode when the internal CPU temperature reaches 90°C, this will be

indicated by toggling the status LED between orange and red.

The camera will stay in this state until the CPU temperature drops below 50°C, it will then reboot automatically and resume operation. Alternatively you can power-cycle the device, but keep in mind that as long as you did not improve the cooling the camera will probably reach the temperature limit again after a short time.

To keep track of the current cpu temperature use the **`temp`** command. The camera also logs the maximum CPU temperature which can be read out using the **`max_temp`** command. You can reset the logged maximum temperature using the **`max_temp_reset`** command.

Every time the camera reaches the 90°C temperature limit, an over temperature counter will be incremented. You can read the counter value by using the **`over_temp_count`** command. This counter value is persistent and can not be reset by the user.

2.3 Status LED

The Status LED shows the status of the device.

Value	Status	Note
Purple blinking	Booting	Device is booting
Yellow blinking	Busy	Device is processing a command or loading settings
Red blinking (slow)	Bootloader	Device is in bootloader mode and can be updated or update is running
Toggle between Yellow and Red	Over Temperature	The device reached it's maximum temperature of 90°C and is in safe mode. Power cycle device or wait until temperature drops below 50°C to resume operation.
Blue blinking	Ready	Running, OK status

3 Firmware Update

Firmware can only be updated via a serial connection. Make sure, that the power supply of the camera and the computer for the update are in a stable state.

You can download the latest updated from our GitLab repository:

https://gitlab.com/dreamchip/provideo-downloads

3.1 Firmware Update via GUI

You can use the ProVideo GUI (see chapter 4) to update your cameras' firmware. This is generally a lot easier, but in some cases (e.g. incompatible GUI and firmware) not possible.

To perform the update simply connect your camera with the included RS485 USB adapter to your PC, start the ProVideo GUI, go to the Update Tab, choose the Firmware file and start the update.

3.2 Firmware Update via Script

The update requires a manual adaption of the update script.

3.2.1 Preparation for Update

Open the folder with the firmware update on your PC.

You need to know your COM port of your USB RS485 interface. This can be done by checking your "Devices and Printers" in your Windows system.

You also need to know your RS485 ID of your ATOM one camera. The default is 1.

Now you must edit the **update_firmware.bat** file and edit the to your settings.

The header of the files looks like this:

```
@echo off
REM SET YOUR COM PORT TO YOUR RS485 USB COM PORT
set COM_PORT=COM156
REM SET THE RS485_ID to your ATOM on Camera ID. Default is 1
set RS485_ID=1
```

Save your changes.

3.2.2 Update of Firmware

Make sure your **update_firmware.bat** reflects the correct COM port and RS485_ID as shown in the previous chapter.

Connect the RS485 USB adapter to your PC and to the ATOM one camera.

Power up the camera and wait until its fully started. This takes ~3 seconds until the LED on the backside is blinking blue.

Now start the **update_firmware.bat** file.

After 2 seconds the camera will go into update mode. When this happens the camera will no longer send an image and the LED on the backside starts blinking red.

After 5 more seconds the firmware update starts.

This process takes about ~10 minutes.

After the firmware update is done, power cycle the camera.

4 ProVideo GUI

Dream Chip provides a free of charge and open source control software (ProVideo GUI) for its camera devices. You should have received a copy of the software with your camera purchase. If not, or you want to check for an updated version please visit our GitLab download page:

https://gitlab.com/dreamchip/provideo-downloads

If you want to develop your own control software, feel free to check out our software repository:

https://gitlab.com/dreamchip/provideo-gui

A user manual for the GUI is part of the GUI download package.

5 Terminal Usage

5.1 Terminal

ATOM one mini can be controlled by a simple terminal connection. All commands consist of ASCI characters.

5.2 Terminal Settings

For flexible configuration ATOM one mini provides a RS485 terminal interface on the Power connector cable.

The default interface setting is:

- 115200 baud (adjustable)
- 8 bit data
- no parity
- 1 stop bit
- no flow control

The ATOM one mini accepts commands in text form and answers with text messages. Every command is confirmed by "OK" or "FAIL". Errors have to be handled by the user or host software which is used to control the device.

Depending on the **prompt** setting you get a prompt "=>" after start up. Now you can send commands to control the device or get status information.

Check the next chapters to get detailed information about available commands and options.

5.3 General Command Format

When the firmware is ready to accept commands, it sends its prompt ("=> ") if enabled with the **prompt** command. Every command line is accepted as a single text line, terminated by either CR or LF.

The command consists of a command string, followed by none or more parameters, separated by a single space.

The parameters can have different formats, depending of the command. Supported formats are:

- Signed decimal : -323, 422
- Hexadecimal: 0x35ff34aa
- String : any_string

The command will be executed and may produce some text output on one or more lines . Finally the status will be sent as a single line containing either "OK" or "FAIL". In case of failure an error code may be added in the same line, e.g. "FAIL 3".

The lines sent by the firmware are terminated with both CR and LF.

Most commands do have parameters. The parameters are depending of the given command.

IMPORTANT: Any command is subject of change. We will try not to change existing commands, but any command may change in the future. Details will be announced in release notes of upcoming software versions.

5.4 RS485 Addressing

On a RS485 Bus multiple ATOM one systems can be used with one RS485 Master controller.

To communicate with a slave, the following command format is defined for RS485:

<address> <command> <parameter>

Where address is the RS485 address of the given ATOM one device. Example command with address 1:

1 video_mode 5

The RS485 address can be set with the **rs485_addr** command, valid addresses range from 0 to 99. The Address 100 is reserved as a fail-safe address. All devices will always reply to commands send on address 100, as if it was their device address. This can be used to recover or change the device address if it is lost and no other communication is possible with the device.

If you want to find out which devices are currently attached to the RS485 bus, you can send the identify command over the fail-safe address 100, which will have all devices report back. See the identify command description for more details.

5.5 RS485 Broadcasting

To send commands to multiple cameras, which are connected to the same RS485 bus at once, they can be grouped in broadcast groups. Therefore each camera gets a second address, the broadcast address. The broadcast address is set with the **rs485_bc_addr** command, it can not be identical with the normal RS485 address, the default is “0”.

Once all cameras in the group are setup with the correct broadcast address, the same command format like for normal RS485 communication is used:

<address> <command> <parameter>

Where address now is the broadcast address.

By default the cameras do not reply to broadcast commands with the usual “OK” or “FAIL” messages, otherwise the RS485 bus would be full with replies from the different devices. To still be able to implement a handshake with a controller software during broadcast mode it is possible to select one camera as the broadcast master. This camera will, representative for all cameras, reply to commands as usual, while all other cameras are silent.

To set the broadcast master, use the command **rs485_bc_master**. Note that this

command may be send to an individual camera, or over the broadcast channel. Sending it over the broadcast channel ensures, that only one master is active in the system.

Reminder: All of this is only valid for communication over the broadcast address, communication over the device address is unaffected and works as usual.

Also see the descriptions of the **rs485_bc_addr** and **rs485_bc_master** commands for more details.

5.6 Inbuild Help

Type **help** to get a full list of supported commands. It is possible to get a detailed help for each command by typing "help <cmd>".

Example to get help for the **video_mode** command:

help video_mode<Enter>

video_mode <video mode> - set video mode

5.7 Command parser

The command parser supports abbreviated commands. This means that it is only required to type as many characters until the command is unique.

Example for equivalent commands:

save_settings

save

sa

For interactive operation this is useful to save typing time.

Note: It is not recommended to use the short commands in any control or application software. The reason is, that future extensions may make a current unique short command ambiguous.

6 Settings Handling

6.1 Set Functions

To change any settings run the matching command with its required parameters.

Example:

video_mode 5

OK

6.2 Get Functions

Any setting can be checked by running the function ***without parameters or followed by a question mark.*** The output of the **get** function represents a valid command followed by parameters. This string can be parsed by external applications to extract the current settings out of the system.

Example:

video_mode

video_mode 5 (returned)

video_mode ?

video_mode 5 (returned)

6.3 auto_save <flag>

Controls auto save.

Flag	Function
0	Auto save OFF
1 (reset)	Auto save ON

With this flag any settings change will instantly saved into the current settings bank for startup.

By turning this off, tests on the camera settings will not change / destroy internal saved settings.

In case auto save is OFF the save_settings command must be used to store the current setup.

Example:

auto_save 0

6.4 save_settings

ATOM one has a build in storage for settings. It is automatically loaded on start up.

The **save_settings** command saves the current settings into the start up configuration.

This function is only required in case auto_save is OFF.

Example:

save_settings

6.5 load_settings

At any time, the saved start up configuration can be loaded with **load_settings**.

This function is useful to try new settings. To discard the tried settings, simply run **load_settings** to restore the known good configuration.

6.6 reset_settings

To reset the system into factory default run **reset_settings**.

You must run **save_settings** to make the factory default the start up configuration.

Please note that all RS485 and RS232 settings, including the baudrate and device address, and the device name are not reset by calling **reset_settings**. This is done so that the communication to the device is not interrupted.

6.7 dump_settings

This command dumps all settings which are stored on the device in alphabetical order. This can for an example be used by a hardware controller to quickly get all settings from the device.

Please note that some commands like **dpc_auto_load** or **lut_interpolate** have no settings that can be dumped and this do not show up when you call **dump_settings**.

7 System Commands

7.1 rs485_baud <baud_rate>

Sets the baud rate for the RS485 terminal interface.

baud_rate
9600
14400
19200
57600
115200 (default)

7.2 rs485_addr <address>

Sets the address for the RS485 interface. It can not be set to the current value of the broadcast address, an attempt to do so will return “FAIL”.

Value	Default	Minimal	Maximal
Device Address	0	0	99

Note: Make sure, that each address is unique on your RS485 bus system.

If you have an address conflict, remove devices from the bus until the contention is solved.

The address 100 is reserved as a fail-safe address. All devices will always replay to commands send over this address, as if it was their current device address. This can be used to identify or change the device address if it has been lost.

7.3 rs485_bc_addr <address>

Sets the broadcast address for the RS485 interface. It can not be set to the current value of the device address, an attempt to do so will return “FAIL”.

Value	Default	Minimal	Maximal
Broadcast Address	0	0	99

To change the broadcast address of an already existing broadcast group, simply send a **rs_485_bc_addr** command over the broadcast channel.

7.4 rs485_bc_master <address>

This command enables the broadcast master mode on the camera with the given device address. If the command is transmitted as a get command (without an argument) it will not reply with the broadcast master address, but with a flag (0 or 1) that indicates whether this

camera currently is the broadcast master or not.

Transmitting a negative address will always disable the broadcast master mode.

Value	Default	Minimal	Maximal
Device Address of the Broadcast Master	Broadcast Master Disabled	0	99

The functionality is explained in the following example:

Two cameras are connected to the same RS485 bus. Camera 1 has the device address 1 and broadcast address 0, camera two has device address 2 and broadcast address 0. To enable broadcast master mode on camera 1 and disable it on camera 2 send the following command using the broadcast channel:

0 rs485_bc_master 1

Where 0 is the broadcast address of both cameras and 1 is the device address of the new broadcast master. You will not receive an “OK” message for this command, but all following commands that are send over the broadcast channel will be answered by camera 1.

Note: If the commands are not transmitted over the broadcast channel, it might happen that multiple broadcast masters exist in the system.

To disable broadcast master mode on all cameras in the broadcast group send:

0 rs485_bc_master -1

7.5 rs485_term <flag>

Enables or disables the internal termination of the RS485 interface.

Flag	Function
0	Disable Termination
1 (default)	Enable Termination

7.6 prompt <flag>

Set prompt mode.

Flag	Function
0 (reset)	No prompt
1	'=>' prompt

7.7 reboot

Does a system reboot (warm start).

A full reboot takes depending of configuration and camera about 30 seconds.

7.8 fw_update

Set the system into USB firmware update mode. This mode is usually controlled by the Windows update software.

7.9 version

Dumps a detailed version information about the system with system ID and firmware version.

The output looks like this:

```
platform        : cooper
device name     : ATOM one mini
system-id       : 00420035-3333470D-30373739-FFFFFFFF
hw revision     : 00000379
system validity: LICENSED
feature mask HW: 0019A014
feature mask SW: FFFFFFFF
resolution mask: 00000000-000007FF-00000000
loader version : 321 (1)
sw-release-id   : V1.0.0
sw-release-date: 2018-02-14 15:51:36 +0100 (Mi, 14 Feb 2018)
sw-build-date   : 2018-02-14 17:57:42
```

7.10 identify

Dumps essential system information including the platform, RS485 configuration and the device name. This command can be used to identify all devices which are connected to one RS485 bus by sending it to device address 100 (the fail-safe device address).

Each device will wait until it is its turn before sending its ID string. The higher the RS485 address of the device, the longer the device will wait before posting its stats, this ensures that the bus does not get corrupted.

Each device will report the following parameters:

- Platform ID: This will be **cooper** for ATOM one mini cameras (this is identical to the platform which is reported by the `version` command)
- RS485 ID, RS485 Broadcast Address and RS485 Broadcast Master as described in chapters 5.4 and 5.5
- Device Name: Name which can be set by the user, see chapter 7.11 for more details (this is identical to the name which is reported by the `version` command)

Example output with two devices attached to one bus:

```
100 identify
id: cooper 1 0 0 ATOM one - Left
OK
id: cooper 55 0 0 ATOM one - Front
OK
```

7.11 name <name string>

The device name, which is shown in the output of the **`version`** command and in the GUI can be changed using a hidden command, which is not shown in the output of the **`help`** command. The name can consist of a maximum of 5 words which, in total, have a length of 32 characters, including spaces.

If the name shall be changed persistently, make sure to execute the **`save`** command after changing the name. The name will not be reset by the **`reset`** command, but it might be reset after a firmware update.

Name String	Default	Reset	Possible Values
Name of the device which is shown in version and GUI	ATOM one 2k	Is not changed through reset	Up to 5 words with 32 characters in total

7.12 flip <mode>

Sets the camera flipping or rotation mode.

Mode	Function
0 (reset)	Normal (no flip)
1	Vertical flip
2	Horizontal flip
3	Rotated by 180°

7.13 log_mode <mode>

Use this command to switch the device into LOG mode. In this mode the camera will use a fixed gamma curve on both outputs (depending on the used mode, see table below). When leaving LOG mode (setting it to “0”) the previously defined settings for the gamma curve will be restored automatically.

Mode	Function
0 (reset)	**Off:** The user can define the gamma curve.
1	**HLG:** A fixed Hybrid LOG gamma curve is used and the gain is internally adjusted to correctly expose the image. This will limit your gain range (the minimum gain is

	doubled).
2	**PQ:** A fixed PQ gamma curve is used, the user can setup the maximum brightness using the “pq_max_brightness” command (see chapter 7.14).

When using a LOG gamma curve more details in bright image areas are preserved, but the image will have a “flat” look. Post processing is needed to generate a naturally looking image.

Please note that when LOG mode is active, you can not adjust the gamma curve. For more details on gamma curve and lookup-table management see chapter 12.

7.14 pq_max_brightness <max>

This command is used to adjust the brightness of the camera image in PQ log mode (see “log_mode” command above for more details). The maximum brightness can be set from 0% to 100%.

When set to 100% the image will contain all highlights but the image will look over exposed on most displays since they can not display the maximum brightness. At the default setting of 50% most monitors should display the image correctly (which means without clipping).

Value	Reset	Minimal	Maximal
Max	50	1	100

7.15 audio_enable <flag>

Enables the internal microphone and embeds the audio signal in the SDI output signal.

Audio Enable	Function
0 (reset)	Audio is disabled
1	Audio is embedded in the SDI signal

7.16 audio_gain <gain>

Change the volume of the audio stream which is embedded in the SDI output signal. The gain is given as a 4.12 fixed point number. The default is a gain of 1.0. To increase the volume set a gain greater than 1.0, to decrease it, set a gain below 1.0.

To convert float to 4.12 fixed point multiply with 4096. Divide by 4096 to convert 4.12 fixed point to float.

Value	Floating Point	4.12 Fixed Point Hex	4.12 Fixed Point Decimal
Minimum	0.0	0x0	0
Reset	1.0	0x1000	4096
Maximum	15.99	0xFFFF	65535

Examples:

To set a gain of 2.0 (double the volume) use:

```
audio_gain 8192
```

or:

```
audio_gain 0x2000
```

To set a gain of 0.5 (half the volume) use either:

```
audio_gain 2048
```

or

```
audio_gain 0x800
```

7.17 temp <sensor id>

Dumps camera temperature values in degree centigrade.

Temperature Sensor ID	Function
0	CPU temperature (with 0.1°C accuracy)

The output has the format **temp <id> <value> <name>,** see the following examples:

```
temp 0
temp 0 61.7 CPU
OK
```

7.18 max_temp

Dumps the maximum logged temperature and the maximum allowed temperature (shut-down temperature) of the camera. This command can be used to evaluate how close the device gets to it's maximum internal temperature of 90°C. Example:

```
max_temp
max_temp 61 90
OK
```

The first value is the maximum temperature that has been logged since the last time the user has run the **max_temp_reset** command (see below). The second value is the overall maximum temperature, it can not be reset by the user. The third value is the shutdown temperature, the device will go into a safe mode when it is reached and the over temperature counter will be incremented.

7.19 max_temp_reset

Resets the logged maximum CPU temperature value to the current temperature.

Example:

```
max_temp_reset
OK
```

7.20 over_temp_count

Dumps the over temperature counter. This counter is incremented every time the device goes into safe mode to avoid over temperature (CPU temperature > 90°C). It is persistent and can not be reset by the user.

Example:

```
over_temp_count
over_temp_count 3
OK
```

8 Camera Commands

ATOM one mini supports different camera models. Some of the following function may not be available depending of the given camera module.

8.1 cam_gain <gain>

Sets analog gain for the sensor.

Value	Reset	Minimal	Maximal
gain	1000	1000	106666

Minimum: 1000 = 1x = 0dB = **30 ISO**

Maximum: 106666 = 107x = 40.6bB = **3200 ISO**

To calculate the ISO value from the analog gain you have to divide it by 1000 and then multiply it with the ISO at gain 1. To read out the ISO at gain 1 you can use the cam_info command (see below). For the ATOM one mini it will report a value of 30.

Examples:

Gain **1000** → ISO = 1000 / 1000 * 30 = **30**

Gain **53333** → ISO = 53333 / 1000 * 30 = **1600**

Gain **106666** → ISO = 106666 / 1000 * 30 = **3200**

You can read out the gain at any time, also when the auto exposure control is enabled.

8.2 cam_exposure <time>

Sets the exposure time/shutter width in **microseconds**. If you change the video mode and the current shutter time is bigger than the maximum possible shutter time for the new video mode, the shutter time will be clipped accordingly.

You can read out the exposure at any time, also when the auto exposure control is enabled.

8.3 cam_info

This command reports the min / max gain and exposure and the ISO value of the camera at gain 1 (1000) in the following order:

minimum gain, maximum gain, minimum exposure, maximum exposure, minimum ISO

Example output:

```
cam_info 1000 16000 75 33333 30
```

9 Video Commands

9.1 video_mode <mode>

Set the output video mode of both output channel.

Mode ID	Resolution	Frame Rate
4	1920x1080	30
5	1920x1080	25
6	1920x1080	24
7	1920x1080	23.98
8	1920x1080	29.97
9	1920x1080	50
10	1920x1080	60
11	1920x1080	60i
12	1920x1080	50i
13	1920x1080	59.94i
14	1920x1080	59.94p

9.2 sdi_black <offset>

Sets the black level for SDI in legal range mode (see **sdi_range**).

An offset value of 0 will result in SDI black value of 64 (SMTP conform).

When changing this value the black level on the SDI interface can be set different than 64 (not SMTP conform).

This value changes the SDI range limiter and will stretch die output values to adapt to the new range.

Value	Reset	Minimal	Maximal
offset	0	-60	+60

9.3 sdi_white <offset>

Sets the white level for SDI in legal range mode (see sdi_range).

An offset value of 0 will result in an SDI white value of 940 (SMTP conform).

When changing this value the white level on the SDI interface can be set different than 940

(not SMTP conform).

This value changes the SDI range limiter and will stretch die output values to adapt to the new range.

Value	Reset	Minimal	Maximal
offset	0	-80	+79

9.4 sdi_range <flag>

Sets the SDI output range type.

Flag	Mode	Digital Code Range	Note
0	Legal	Y ranges from 64 + sdi_black offset Y ranges from 940 + sdi_white offset U/V range from 64 to 960	Used for broadcast and monitors with defined black and white levels.
1	Extended	Y/U/V range from 4 to 1019	Used for recoding with maximum dynamic.

9.5 post_bright <offset>

Set post processing brightness.

Y_out = Y + offset

Neutral value is 0.

Value	Reset	Minimal	Maximal
offset	0	-128	127

9.6 post_cont <factor>

Set post processing contrast.

Y_out = Y * factor / 128

Neutral value is 128.

Value	Reset	Minimal	Maximal
factor	128	0	255

Example Pseudo Code:

```
float contrast = 1.23f;
float c =  contrast * 128.0f;          // c = 157.44
int value = (int) round( c );          // value = 157
```

 # post_cont 157

9.7 post_sat <factor>

Set post processing color saturation.

Cb,r_out = Cb,r * factor / 128

Neutral value is 128.

Value	Reset	Minimal	Maximal
value	128	0	255

Example Pseudo Code:

```
float saturation = 1.23f;
float s =  saturation * 128.0f;          // s = 157.44
int value = (int) round( s );            // value = 157
# post_sat 157
```

9.8 post_hue <offset>

Set post processing color hue offset angle.

Cb' = Cb * cos (offset * 90 / 128) + Cr * sin (offset * 90 / 128)

Cr' = -Cb * sin (offset * 90 / 128) + Cr * cos (offset * 90 / 128)

Neutral value is 0.

Value	Reset	Minimal	Maximal
value	0	-128 (-90 degree)	127 (+89 degree)

Example Pseudo Code:

```
float hue = 22.1f;                       // hue = 22.1°
float h =  hue * 128.0f / 90.0f;         // h = 31.43111...
int value = (int) round( h );            // value = 31
# post_hue 31
```

9.9 wb

Triggers single shot white-balance.

9.10 awb <flag>

Enables continuous auto white balance.

Flag	Function
0 (reset)	disable

1	enable

9.11 awb_speed <speed>

Sets the control speed of the continuous auto white balance.

Value	Reset	Minimal	Maximal
speed	0	0 (slow)	2 (fast)

9.12 wb_preset <id>

Set calibrated white balance presets (gains and color-cross matrices).

List of calibrated presets:

id	illumination	color temperature
0	horizon	2200K
1	candle light (A)	2700K
2	fluorescent (TL84)	3700K
3	fluorescent (CWF)	4000K
4	daylight sunny (D50)	5000K
5	daylight shadow (D62)	6200K
6	daylight cloudy (D65)	6500K

9.13 gain_red <gain>

Set gain factor for red component for selected output channel.

This function is for basic color correction or white balance.

red_out = red * gain / 256

Value	Reset	Minimal	Maximal
gain	256	0	4095

Example Pseudo Code:

```
float gain = 1.23f;
float gain_scaled = gain * 256.0f;          // factor = 314.88
int value = (int) round( gain_scaled );     // value = 315
```

 # gain_red 315

9.14 gain_blue <gain>

Equivalent to gain_red.

9.15 gain_green <gain>

Equivalent to gain_red.

9.16 black_master <offset-red> <offset-green> <offset-blue>

Set the black-level offset for red, green and blue-components for selected output channel. The processing is done in linear RGB domain (pre gamma).

red_out = (red_in – offset-red) *4095 / (4095 – offset-red)

Value	Reset	Minimal	Maximal
offset	0	-2047	2048

Examples:

set black-level for all components to 100

 # black_master 100

set black-level for red to 10, for green to 20 and blue to 30

 # black_master 10 20 30

9.17 black_red <offset>

Set offset for red component for selected output channel as black level setting. The processing is done in the linear BAYER domain (pre debayering).

The offset is defined as singed value. A value of zero

is treated as neutral.

red_out = red - offset

Value	Reset	Minimal	Maximal
offset	0	-4096	4095

9.18 black_blue <offset>

Equivalent to black_red.

9.19 black_green <offset>

Equivalent to black_red.

9.20 flare <red level> <green level> <blue level>

Set flare compensation level (= Defogging). The processing is done in the linear BAYER domain (pre debayering).

The level is defined as unsigned value. A value of zero is treated as neutral (disable flare).

$$red_{out} = \frac{4095 * (red_{in} - level * - red)}{4095 - level * - red}$$

Value	Reset	Minimal	Maximal
offset	0	0	65535

Example Pseudo Code:

```
float lvl_percent = 0.1f;          // 10% compensation
float lvl = lvl_percent*65536.0f;  // lvl = 6553.6
int value = (int) round( lvl );    // value = 6554
```

=> flare 6554 6554 6554

To set same compensation level for all components use:

=> flare 6554

9.21 filter_enable <flag>

Enable function for the filter functions.

Flag	Function
0	disable
1 (reset)	enable

9.22 filter_detail <level>

Set the detail enhance level.

Value	Reset	Minimal	Maximal
level	0	0	65

9.23 filter_denoise <level>

Set the denoise level.

Value	Reset	Minimal	Maximal
level	0	0	65

9.24 color_conv <c0> .. <c8>

Sets the color conversion matrix.

The following formula is used for the conversion.

```
Y  = (c0 *r + c1*g + c2 *b) / 4096 + 64
Cb = (c3 *r + c4*g + c5 *b) / 4096 + 512
Cr = (c6 *r + c7*g + c8 *b) / 4096 + 512
```

Value	Reset	Minimal	Maximal
C0	871	-8192	8191
C1	2929	-8192	8191
C2	296	-8192	8191
C3	-469	-8192	8191
C4	-1579	-8192	8191
C5	2048	-8192	8191
C6	2048	-8192	8191
C7	-1860	-8192	8191
C8	-188	-8192	8191

Example Pseudo Code:

float coeff7 = -0.454152908f;

float coeff7_scaled = coeff7*4096.0f; // factor = -1860.210...

int value = (int) round(coeff7_scaled); // value = -1860

 # color_conv 871 2929 296 -469 -1579 2048 2048 -1860 -188

9.25 color_cross <c0> .. <c8>

Sets the color cross talk matrix.

The Cross-Talk block in ISP is meant for correction of cross talk effects and color space shifts inside the camera sensor.

The cross talk compensation unit performs a regular RGB to R'G'B' color space conversion, to compensate the cross talk between color components of the image sensor. Also corrections of the color space and saturation can be done by appropriate matrix coefficients.

Therefore this block offers the ability to correct each pixel value with a matrix operation which looks like this:

R' = (R x c0 + G x c1 + B x c2) / 4096

G' = (R x c3 + G x c4 + B x c5) / 4096

B' = (R x c6 + G x c7 + B x c8) / 4096

Value	Reset	Minimal	Maximal
C0	4096	-32768	32767
C1	0	-32768	32767

C2	0	-32768	32767
C3	0	-32768	32767
C4	4096	-32768	32767
C5	0	-32768	32767
C6	0	-32768	32767
C7	0	-32768	32767
C8	4096	-32768	32767

Example Pseudo Code:

float coeff7 = 1.123f;

float coeff7_scaled = coeff7*4096.0f; // factor = 4599.808

int value = (int) round(coeff7_scaled); // value = 4600

 # color_cross 4096 0 0 0 4096 0 0 4600 4096

9.26 color_cross_offset <red_offset> <green_offset> <blue_offset>

Sets the color cross talk offset in the FPGA.

In addition to the matrix multiplication an offset can be added to the pixel values for R, G and B separately. This offset is applied after the matrix multiplication.

The offset values are two's complement integer numbers with a range of -2048 (0x800) to 2047 (0x7FF). 0 is represented as 0x000.

Value	Reset	Minimal	Maximal
red_offset	0	-2048	2047
green_offset	0	-2048	2047
blue_offset	0	-2048	2047

9.27 color_space<color space>

Configures the image pipeline for the selected color space. This is a high-level command which will change both the color cross talk an color conversion matrix.

If you are using a custom color cross talk matrix (see chapter 9.25) it will be automatically converted when switching the color space, but due do rounding errors in the conversion each switch will result in a more altered color cross table.

The color conversion matrix (see chapter 9.24) will always be set to the default for the selected color space, this means any changes the user has made will be overwritten when the color space is changed.

Value	Color Space	Usually used for
0 (reset)	Rec.709	HD / SDR
1	Rec.2020	UHD / HDR

9.28 mcc <flag>

Multimatrix color correction control.

Flag	Function
0	disable
1 (reset)	enable

9.29 mcc_opmode <flag>

Sets the number of color angles/segments for the multi matrix.

The segment 0 is at the top of the color circle with the color red.

Please refer to the GUI as reference.

Flag	Function
0	12
1	16
2 (reset)	24
3	32

9.30 mcc_blink <mask> [<period_ms>]

This function lets the select multi matrix color range blink with a given period time.

This can be used to check if the selected colors are correct.

Value	Minimal	Maximal	Comment
mask	0	0xFFFFFFFF	Each bit of the mask represents the blink mode of one phase of the MCC. Setting it to 1 enables blinking for this phase.
period_ms	0	0xFFFFFFFF	Sets the period time of the blinking. The blink mode will be toggled each period_ms / 2 ms. This parameter is optional, if left empty, a period time of 1s will be used.

9.31 mcc_set [<saturation>] [<hue>]

Sets parameters for multimatrix color correction.

The number of angles are depending of the mcc_opmode. This table shows 24 angles in detail compared with to a sony camera with 24 angles.

For any segment the saturation and the hue can be set.

Value	Minimal	Maximal	Comment
segment	0	23	Index of color segment. Each Segement has 15 degree in color space. This table shows the mapping of index compared to Sony names and angles. Segment = Angle = Color 17 = 0° BLUE 18 = 15° B+ 19 = 30° B++ 20 = 45° MG- 21 = 60° MG 22 = 75° MG 23 = 90° MG + 24 = 105° R- 01 = 120° R 02 = 135° R+ 03 = 150° R/YL 04 = 165° YL- 05 = 180° YL 06 = 195° YL+ 07 = 210° YL/G 08 = 225° G- 09 = 240° G 10 = 255° G 11 = 270° G+ 12 = 285° CY- 13 = 300° CY 14 = 315° CY+ 15 = 330° CY/B 16 = 345° B-
Saturation	0 0.0	65535 3.999938965	See notes below
Hue	-32768 -180 degree	32768 179.99450 degree	See notes below

With no parameter it dumps all 24 segments.

=> mcc_set

mcc_set 0 16384 0

mcc_set 1 16384 0

mcc_set 2 16384 0

mcc_set 3 16384 0

mcc_set 4 16384 0

…

OK

You can dump one segment by specifying one segment only.

=> mcc_set 3

mcc_set 3 16384 0

OK

Examples of setting hue and saturation of a color phase:

```
float saturation = 1.23f;          // saturation of 1.23
float s = saturation * 4096.0f;    // s = 5038.08
int value = (int) round( s );      // value = 5038

float hue = 1.1f;                  // hue rotation by =+1.1°
float h = hue * 16384 / 90;        // h = 200.2489
int value = (int) round( h );      // value = 200

float hue = -4.3f;                 // hue rotation=-4.3°
float h = hue * 16384 / 90;        // h = -782.7911
int value = (int) round( h );      // value = -783
```

9.32 lsc <enable> <k> <offset> <slope>

Due to the physical properties of the lenses used in optical imaging a reduction of the luminescence occurs from the middle of the image sensor to its borders. This is commonly known as vignetting. This effect can be separated into natural and artificial vignetting, both can be corrected using the Lens Shading Correction module. Please note that this is an advanced feature that requires the right measurement equipment to be setup correctly (see setup instructions at the end of this chapter).

The lens shading correction uses a correction function to increase the gain in the outer image areas. The parameter k configures the natural vignetting compensation (cos4 compensation), a higher value will result in a higher compensation. The parameters offset and slope configure the artificial vignetting compensation. If the value for offset is increased, the radius where the compensation starts will be increased (moved to the image border). The higher the slope, the stronger is the compensation. The values k and slope range from 0.0 to 2.0, the offset from 0.0 to 1.0.

Please note that all values have to be passed as fixed point numbers in Q2.30 format, that

means a value of 0.67 equals: 0.67 * 2^30 = 719407022

Value	Minimal	Maximal	Default	Comment
Enable	0	1	0	Enables or disables the lens shading correction module
K	0.0 (0)	2.0 (2147483648)	0	Corrects natural vignetting, has to be given in Q2.30 fixed point format
Offset	0.0 (0)	1.0 (1073741824)	0	Corrects artificial vignetting, has to be given in Q2.30 fixed point format
Slope	0.0 (0)	2.0 (2147483648)	0	Corrects artificial vignetting, has to be given in Q2.30 fixed point format

To setup the lens shading correction, follow these steps:

1. Point the camera at a homogeneous light source.
2. Connect the camera to a wave monitor and select a line in the middle of the image, you should see a decrease in luminescence towards the edges of the image.
3. Enable lens shading correction, start with the parameters K, Offset and Slope set to 0.
4. Now turn up the K factor, this should correct the lens shading in the middle area of the image, the edges will probably still be not ideally illuminated. Make sure to not overcompensate, this will create a wavelike appearance of the luminescence on the monitor.
5. Set the Offset to 0.5 and set to slope to a high value like 1.5, you should now clearly see where the compensation starts. Now adjust the slope until you hit the point where the luminescence starts decreasing. Finally decrease the slope until the result is not overcompensated anymore.
6. Make fine adjustments until you are satisfied with the result.

Please note that, depending on the optical lens used, the aperture and focal length have influence on the lens shading, so you should use your default setup for configuring the compensation. Also it might be helpful to turn of the auto exposure during the setup.

9.33 cnr_auto <flag>

In low light conditions the gain of the camera has to be increased, which will also increase image noise. Normal image noise (noise in the luminescence) can be removed by using the denoise filter (see chapter 9.23), but this will also make the image look more blurry.

To just reduce color noise in the image, the automatic color noise reduction (CNR) can be used. If enabled it will turn on additional color denoising, depending on the gain value.

Flag	Function
0 (reset)	disable

1	enable

10 Defect Pixel Correction

The ATOM one mini does not support live detection of defect pixels, but it features a very simply calibration procedure:

1. Start the camera, for best results let it run for a few minutes so that the sensor heats up, otherwise some defect pixels may not be found
2. Place a cap on the lens so that the image goes black
3. Run the **`dpc_auto_load`** command, this will take a few seconds
4. The defect pixel table will be stored in the persistent device memory and automatically loaded

10.1 dpc <flag>

Enables / Disables deflect pixel correction

Flag	Function
0	No correction
1 (reset)	Correction enabled

10.2 dpc_test_mode <mode>

Sets defect pixel test mode. By default the test mode is disabled. There are two test modes available: Calibration and Validation.

In Calibration mode the camera will output a black and white image with maximum gain, which is ideal to detect defect pixels.

In Validation mode the camera will output a black image with colored pixels at those positions where the defect pixel correction is currently correcting pixels.

Mode	Function
0 (reset)	Disabled
1	Calibration
2	Validation

10.3 dpc_auto_load

Automatically detects defect pixels and stores them in the the persistent memory. To check which pixels are currently being detected, set **`dpc_test_mode`** to validation mode.

You do not need to activate calibration mode before running the auto load command, it will do so automatically.

11 Knee Function

11.1 knee <flag> <knee_point> <knee_slope> <white_clip>

Knee function for highlight control.

For the knee function you can set 3 parameters:

Value	Reset	Minimal	Maximal
flag (enable)	0 (off)	0	1
knee_point [%]	85	50	100
knee_slope	140	100	200
white_clip [%]	109	100	109

 # enable knee at 85% with a slope of 1.4 and white clip at 109%

=> knee 1 85 140 109

OK

12 Look-up Table Management

The lookup table has a bit depth of 16 Bit (both input and output values). There are four ways to setup the lookup table:

1. The LUTs are fully programmable with 24 sample points. All intermediate values are calculated with a spline interpolator.
2. The LUTs can be programmed by specifying a set of parameters which will be used to calculate a gamma curve like it is specified in the REC.709 standard. This will then automatically compute the 24 sample points needed for method 1 and start the spline interpolator.
3. The LUTs can be fast configured by specifying only the desired gamma value of the REC.709 gamma function.
4. The LUTs can be configured with a fixed configuration of three presets which include the default REC.709 gamma curve and two HDR gamma curves: PQ and HLG.

Method 1 and 2 are intended for offline calibration of the camera. Method 3 is intended for usage during device operation. Method 4 can be used if the gamma table is not changed during production.

For method 1 and 2 the user can store 5 independent presets with 24 sample points each. The default is the standard REC.709 curve, but the presets can be changed by the user. For method 3 and 4 presets are not available.

The LUT's can be configured for each color component separately (Red, Green, Blue) when using method 1. Method 2 and 3 will setup all components identically.

12.1.1 How to setup a lookup table using interpolation

When using interpolation switch to lut mode 0: **lut_mode 0**

Method 1: For **manual LUT** setup the following steps are required.

1. Disable LUT: **lut_enable 0 0**
2. Select preset storage: **lut_preset 0**
3. Clear current LUT configuration: **lut_reset**
4. Add your sample points: **lut_sample 100 200 200 400 500 2000 <...>**
5. Calculate LUT curve: **lut_interpolate**
6. Enable LUT again: **lut_enable 0 1**

Method 2: For **functional LUT** setup the following steps are required.

The functional LUT will calculate sample points based on a function.

1. Disable LUT: **lut_enable 0 0**
2. Select preset storage: **lut_preset 0**
3. Set your sample points (includes clear): **lut_fun_rec709 <...>**

4. Calculate LUT curve: **lut_interpolate**
5. Enable LUT again: **lut_enable 0 1**

Switch Presets: For **preset LUT** setup the following steps are required.

Presets are user sets to be loaded.

1. Disable LUT: **lut_enable 0 0**
2. Select preset storage: **lut_preset 0**
3. Calculate LUT: **lut_interpolate**
4. Enable LUT again: **lut_enable 0 1**

12.1.2 How to setup a lookup table using the fast gamma function

When using fast gamma setup switch to lut mode 1: **lut_mode 1**

Method 3: The **fast gamma** lut setup will calculate the lut using a function by specifying only the desired gamma value:

1. Make sure LUT is enabled: **lut_enable 0 1**
2. Setup gamma value: **lut_fast_gamma <...>**

12.1.3 How to setup a lookup table using the fixed gamma mode

When using the fixed gamma mode switch to lut mode 2: **lut_mode 2**

Method 4: The fixed gamma lut setup will calculate the lut by using different gamma functions:

1. Make sure LUT is enabled: **lut_enable 0 1**
2. Setup fixed gamma curve: **lut_fixed_mode <...>**

12.2 lut_enable <index> <flag>

Enable function for look up table.

Index	Function
0	LUT-1 on SDI-1

Flag	Function
0	Disable (linear)
1 (reset)	Enable

Examples:

```
# enable LUT of SDI-1
=> lut_enable 0 1
```

```
OK
# dump current LUT states
=> lut_enable
lut_enable 0 1
OK
```

12.3 lut_mode <mode>

Selects the LUT operational mode.

Value	Reset	Minimal	Maximal
Mode	0	0	2

There are three operational modes available:

Mode	Description
0	Table based using interpolation. The user has to specify a table with a maximum of 24 values. Intended for offline calibration.
1	Fast gamma mode where the user only specifies the desired gamma value. Can be used for gamma changes during runtime.
2	Fixed gamma mode where the user selects one of three fixed gamma tables.

12.4 lut_preset <index>

Selects the preset storage for the current LUT interpolator.

Value	Reset	Minimal	Maximal
index	0	0	4

The ATOM one can internally handle 5 presets. Any preset can be modified by the customer.

Index	Factory preset
0	REC709
1	Linear
2	Linear
3	Linear
4	Linear

Example:

```
=> lut_preset 0
OK
```

12.5 lut_sample <xi_0> <yi_0> ... <xi_7> <yi_7>

Defines the sample points in a lookup table. The x-value is the input value, the y-value is output.

Value	Minimal	Maximal	Comment
xi_n	0	65535	16 bit input value
yi_n	0	65535	16 bit output value

Sets the up to 8 sample points of 24 for the look up definition for all colors.

For setting more than 8 points, the function can be called multiple times.

In case a x-position is set twice, the previous value is overwritten.

When more than 24 points are defined, an error message is dumped.

If the gamma curve is set with this function, **save_settings** will start the system with this values on boot up.

Note: To read back the current sample points you must use the **lut_sample_<color>** Command. **lut_sample** will not return sample data.

Examples:

```
# reset sample points and add start/end point
=> lut_reset
OK
# dump sample points
=> lut_sample_red
lut_sample_red 0 0 65535 65535
OK
# overrule start and end point
lut_sample_red 0 1000 65535 64000
OK
# start curve interpolation
=> lut_interpolate
OK
```

12.6 lut_sample_red <xi_0> <yi_0> ... <xi_7> <yi_7>

Same as **lut_sample**, but only for red component.

12.7 lut_sample_green <xi_0> <yi_0> ... <xi_7> <yi_7>

Same as **lut_sample**, but only for red component.

12.8 lut_sample_blue <xi_0> <yi_0> ... <xi_7> <yi_7>

Same as **lut_sample**, but only for red component.

12.9 lut_interpolate

Interpolates all look up tables based on the given sample points.

12.10 lut_interpolate_red

Interpolates the red look up table based on the given sample points.

12.11 lut_interpolate_green

Interpolates the red look up table based on the given sample points.

12.12 lut_interpolate_blue

Interpolates the red look up table based on the given sample points.

12.13 lut_reset

Clears all look up sample points.

12.14 lut_reset_red

Clears all red look up sample points.

12.15 lut_reset_green

Clears all green look up sample points.

12.16 lut_reset_blue

Clears all blue look up sample points.

12.17 lut_fun_rec709 <threshold> <linear-contrast> <linear-brightness> <contrast> <gamma> <brightness>

Sets sample points according to REC.709 for the current LUT.

Existing sample points will be cleared before run.

Sets the gamma curve for all 3 colors.

If the gamma curve is set with this function, **save_settings** will start the system with this

values on boot up.

Value	Minimal	Maximal	REC.709
Threshold	0	1000	18
Contrast (linear)	0	10000	4500
Brightness (linear)	-1000	1000	0
Contrast (non-linear)	0	10000	1099
Gamma (non-linear)	0	1000	450
Brightness (non-linear)	-1000	1000	-99

Note: The values are normalized to a range from 0.0 to 1.0 and multiplied by a scaling coefficient of 1000.

The following image shows the normalized REC.709 gamma curve and it's transition from linear to non-linear (power function) part.

The linear part ranges from 0 to <0.018 and is computed by the following formula:

$$V_{\text{out}} = 4.5 * V_{\text{in}}$$

The maximum value in this range is:

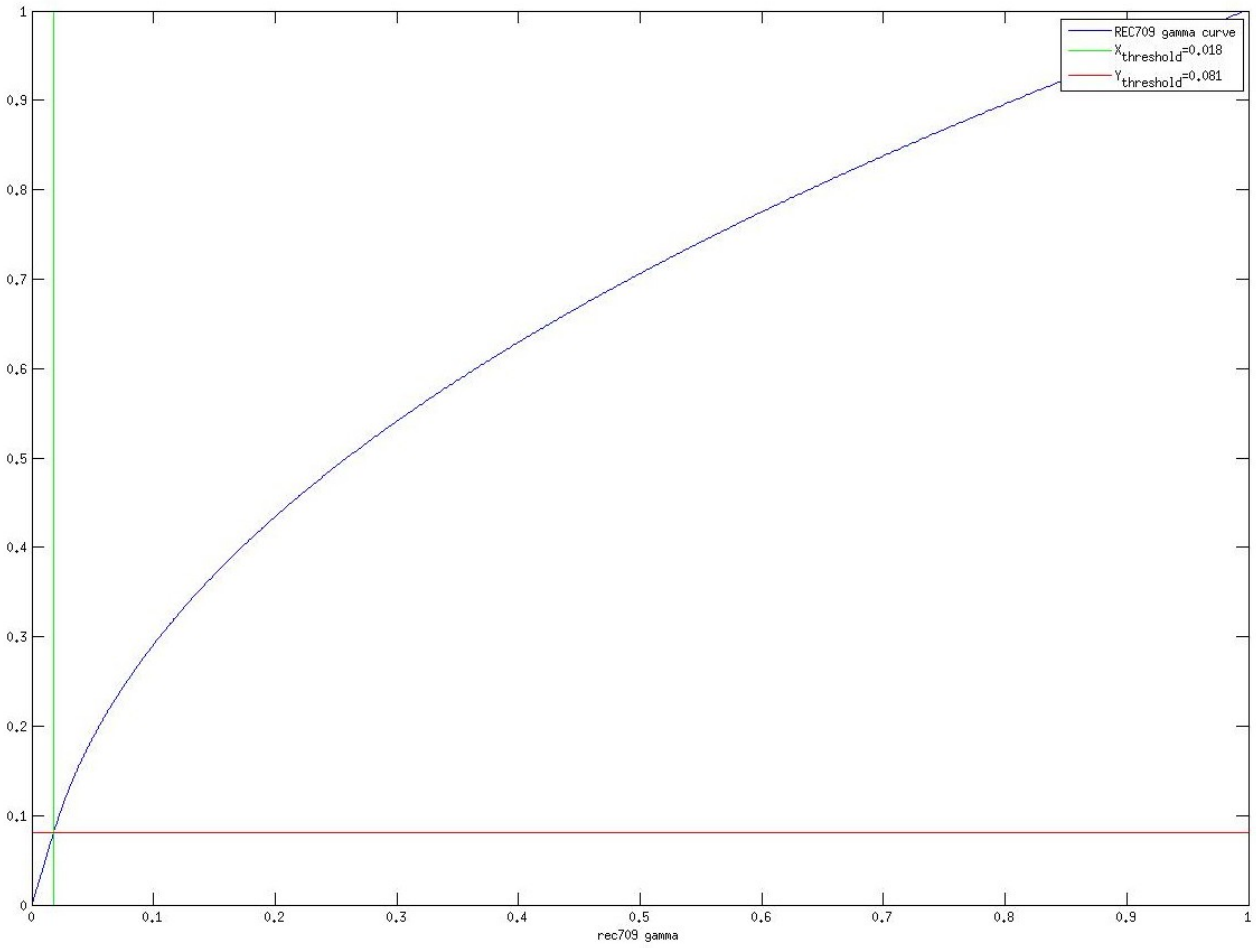
$$V_{\text{out,min}} = \lim_{V_{\text{in}} \to 0.018} (4.5 * V_{\text{in}}) = 4.5 * 0.018 = 0.081$$

The non-linear part ranges from 0.018 to 1 and is computed by the following formula:

$$V_{\text{out}} = 1.099 * V_{\text{in}}^{0.45} - 0.099$$

The minimum value in this range is:

$$V_{\text{out,min}} = 1.099 * 0.018^{0.45} - 0.099 = 0.081\ldots$$



To avoid a discontinuity in the gamma-function select the brightness (offset) for the non-linear part by the following formula:

$$V_{\text{out, linear max}} \approx V_{\text{out, non-linear min}}$$

$$Brightness_{\text{non-linear}} \approx (Contrast_{linear} * threshold + Brightness_{linear}) - Contrast_{\text{non-inear}} * threshold^{Gamma}$$

Example Pseudo Code to setup a REC.709 gamma curve:

```
float threshold         = 0.018;
float linear_contrast   = 4.5f;
float linear_brightness = 0.0f;
float gamma             = 0.45f;
float contras           = 1.099f;
float brightness        = -0.099f;
int t  = (int) round( threshold * 1000.0f );          // t  = 18
int lc = (int) round( linear_contrast * 1000.0f );    // lc = 4500
```

```
int lb = (int) round( linear_brightness * 1000.0f );      // lb = 0
int c  = (int) round( contrast * 1000.0f );               // c  = 1099
int g  = (int) round( linear_contrast * 1000.0f );        // g  = 450
int b  = (int) round( brightness * 1000.0f );             // b  = -99
```

Example curve for REC.709:

```
=> lut_gamma 18 4500 0 1099 450 -99
OK
```

12.18 lut_fast_gamma <gamma>

The fast gamma function uses the same formula as the **lut_fun_rec709** command, but the user only has to specify the desired gamma value. All other values are fixed or computed as needed.

Value	Minimal	Maximal	REC.709
Gamma	1100	3000	2222

Note: The value is normalized to a range from 0.0 to 1.0 and multiplied by a scaling coefficient of 1000. Setting a value of 2222 will result in the default REC.709 gamma curve.

12.19 lut_fixed_mode <mode>

The fixed gamma mode allows to choose between three fixed gamma tables which are shown in the table below.

Mode	Description
0 (default)	REC.709 gamma curve
1	PQ gamma curve specified in ITU-R BT.2100 which can be used for HDR content
2	HLG gamma curve specified in ITU-R BT.2100 which can be used for HDR content

13 Image statistic commands

13.1 stat_rgb

Dumps the RGB mean values.

Example:

stat_rgb

stat_rgb 1040 896 1200

Output description:

- Average Red = 1040 (12 bit)
- Average Green = 896 (12 bit)
- Average Blue = 1200 (12 bit)

13.2 stat_hist <mode>

Set histogram statistics mode.

Mode	Histogram statistic
0	disable
1	RGB combined histogram
2	R histogram
3	G histogram
4	B histogram
Y	Y (luminance) histogram

Note: The histogram is measured in 16 intensity intervals. The interval width is 2^12/16. The first interval ranges from intensity level 0 to 255, the 2nd interval from 256 to 511, the 3rd interval from 512 to 767 and so on.

Example:

enable RGB combined histogram

stat_hist 1

OK

show histogram statistic

stat_hist

system dumps

stat_hist 1 134120 160777 70308 40823 31308 21310 33414 50317 48382 30265 23750 17518 9922 3485 2658 7213

Output description:

Parameter	Value	Description
mode	1	RGB combined histogram
Interval-0	134120	Interval-0 contains 134120 pixel.
Interval-1	160777	Interval 1 contains 160777 pixel.
Interval-2	70308	Interval 2 contains 70308 pixel.
Interval-3	40823	Interval 3 contains 40823 pixel.
...	...	...

13.3 stat_exp <mode>

Set exposure statistics mode.

Mode	Exposure statistic
0	disable
1	Y - Luminance average (12 bit)

Note: The exposure statistic is measured in 5x5 (25) sub-windows which results into 25 mean luminance/exposure values. The width of a sub-window is resolution width divided by 5. The height of a sub window is resolution height divided by 5.

Following table shows the numbering of the statistic values:

Y0	Y1	Y2	Y3	Y4
Y5	Y6	Y7	Y8	Y9
Y10	Y11	Y12	Y13	Y14
Y15	Y16	Y17	Y18	Y19
Y20	Y21	Y22	Y23	Y24

Example:

enable luminance statistics

stat_exp 1

OK

show exposure statistic

stat_exp

system dumps

stat_exp 1 3872 3872 3872 3872 3888 3872 3872 3872 3872 3888 3872 3872 3872 3888 3888 3872 3888 3888 3872 3888 3872 3888 3888 3872 3888

Output description:

Parameter	Value	Description
mode	1	Exposure statistic enabled
sub-window 0	3872	Sub window 0 (upper left corner) has a mean luminance of 3872.
sub-window 1	3872	Sub window 1 has a mean luminance of 3872.
...	…	...
sub-window 24	3888	Sub window 24 (lower right corner) has a mean luminance of 3888

14 SDI Time code

14.1 timecode <hour> <minute> <second>

The SDI signal has an embedded time code. Since ATOM one has no real time clock, the time code can be set manually.

Sets SDI time code.

Example:

get current time code

timecode

system dumps:

timecode 0 57 24
OK

set sdi time code to 2h 10m 53s

timecode 2 10 53

system acknowledges:

OK

14.2 timecode_hold <flag>

This command can be used to hold the timecode on the SDI output. This can be used to trigger recording in an external flash recorder. Note that the timecode will continue to run in the camera, this means as soon as hold is released the SDI timecode will jump to the current value.

Flag	Timecode Hold
0	Normal operation, SDI timecode is running
1	Hold, SDI timecode is unchanged until hold is released

15 Auto Exposure

15.1 aec <enable> <setPoint> <speed> <clmTolerance> <costGain> <costTint> <costApt> <taf> <maxGain> <useCustomWeights>

Configures auto exposure control.

The auto exposure is fully controllable by the user. Depending of the application the user can decide how the exposure control should work.

The target luminance is set by the setPoint, this will determine how bright the image looks. A higher setPoint results in a brighter exposed image.

By default the auto exposure does not use the full available gain range. Use the maxGain parameter to change the maximum sensor gain which is used by the algorithm. Please note, that high maximum gain values can make the auto exposure control unstable.

If the lighting conditions allow it, the exposure control will avoid flicker effects caused by artificial (indoor) lights by only using shutter times which are multiples of a given flicker frequency taf. For countries with power frequencies of 50 Hz (e.g. Europe) you should use a taf value of 10000 µs, in countries with a 60 Hz power frequency (e.g. USA), use a value of 8333 µs.

The speed determines how fast the exposure control reacts to luminescence changes. A higher value means a slow reaction to changes.

The sensitivity determines how big a luminescence change has to be before the auto exposure control reacts to it. A higher value means a bigger change in luminescence is needed.

By settings a “cost” factors for gain, exposure or aperture the user can set priorities how the exposure should work. The parameter with the lowest costs is most preferred by the auto exposure control, the parameter with the highest costs will be kept as low as possible.

Setting a cost factor to 0 will remote this parameter out of the automatic loop. The user must set this factor to fixed value. Check the examples below to get an idea how it works.

To use a customized aec algorithm set the “useCustomWeights” flag. For more information on how to setup the custom weighting, see chapter 15.2.

Function call:

`aec <enable> <setPoint> <speed> <clmTolerance> <costGain> <costTint> <costApt> <taf> <maxGain>`

Parameter	Description	Remarks
enable	0: disable, 1: enable aec	
setPoint	Target luminance	Range 256...3000, Default 1000
speed	Control speed	Range 3…30, Default: 10
clmTolerance	Sensitivity (threshold)	Range 10...500. Default: 50
costGain	Cost factor analog gain	Range 0, 250...8000. Default: 8000

costTint	Cost factor exposure	Range 0, 250...8000. Default: 250
costApt	Cost factor aperture	Range 0, 250...8000. Default: 0 (disabled)
taf	Anti Flicker period in [µs]	Range 5000...20000, Default: 10000 Following values are usually used: 10000 for 50 Hz (e.g. for EU countries) 8333 for 60 Hz (e.g. for USA)
maxGain	Maximum sensor gain which is used by the auto exposure control	See the cam_gain command for the valid range, by default a value of 53333 is set which equals a gain of 1600 ISO
useCustomWeights	0: disabled, 1: enabled	By default the AEC uses an optimized algorithm to determine the needed exposure which also takes into account back light lighting conditions. If you want to use a more simple but customizable algorithm, enable this option. See chapter 15.2 for more details on how to setup the custom weights.

Examples:

disable aec completely

```
aec 0
```

enable aec with luminance 2500, don't touch the other parameters

```
aec 1 2500
```

enable aec with luminance 2500, 60 Hz anti flicker period
and other parameters set to default values

```
aec 1 2500 10 50 8000 250 0 8333
```

no iris, gain fixed only shutter control

```
aec 1 895 10 50 0 1000
cam_gain 1000
```

no iris, exposure fixed only gain control

```
aec 1 895 10 50 1000 0
cam_exposure 5000
```

limit gain to 15000, other parameters at default

```
aec 1 895 10 50 8000 250 0 10000 15000
```

Please note that aperture control is only available on some OEM devices.

15.2 aec_weight <index> <weight>

By default the AEC uses an optimized algorithm to determine the needed exposure which

also takes into account back light lighting conditions. This algorithm should work fine for most use cases, but some might require manually tweaking the auto exposure. To do this use the aec command and set the “useCustomWeights” flag (see chapter 15.1 for more details).

Value	Minimal	Maximal	Description
index	1	25	Index of the field which weight shall be changed
weight	1	25	Weight of the field at the given index

In custom mode the auto exposure uses a 5x5 grid which is equally split across the image. In each field of the grid the average brightness is measured. For each field a weight can be specified, which determines how strong the average brightness of this field is taken into account by the auto exposure algorithm.

The following table shows the index of each field of the grid:

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

By default all fields are weighted equally (with a factor of 1). If, for an example, the top-right corner of the image is the most important part of the currently filmed scene, but it is over- or underexposed with the default setting, a solution is to increase the weight of the fields 4,5, 9 and 10 to a higher value, e.g. 5:

```
aec_weight 4 5
aec_weight 5 5
aec_weight 9 5
aec_weight 10 5
```

This will take the average brightness of those fields more into account, leading to this parts being correctly exposed, at the cost that the rest of the image will not be perfectly exposed.

15.3 stat_ae

Dump auto exposure statistics

Example:

Get stat_ae information
stat_ae <z> <lumaDeviation> <semSetPoint> <meanLuma> <meanLumaObject> <meanHistogram> <clmHistogramSize> <sumHistogram> <maxExposure>

<newExposure> <targetExposure> <realExposure> <gain> <tint> <aperture>

Parameter	Description
z	Backlight factor x 1000
lumaDeviation	Control deviation
semSetPoint	Modified target value
meanLuma	Mean luminance
meanLumaObject	Mean luminance of object region
meanHistogram	Mean luminance determined by histogram
clmHistogramSize	Histogram bin count
sumHistogram	Sum of histogram values
maxExposure	Maximum abstract exposure value
newExposure	New calculated abstract exposure value
targetExposure	New calculated abstract exposure value after low pass filter
realExposure	Actual abstract exposure value
gain	Analog gain
tint	Exposure time
aperture	Aperture factor

16 Live indicator

The live indicator is an animated symbol that shows that the camera or a video transmitting is working (“no freeze frame”).

16.1 osd <mask>

Sets type of overlay. The mask value controls the overlay. 0 disables the overlay.

Bit	Value	Minimal
0	1	enable/disable 0° wing live indicator
1	2	enable/disable 180° wing live indicator
2	4	enable/disable bar code indicator

Example:

enable all indicators

=>osd 7

16.2 osd_bmp <id>

Sets wing shape

ID	Minimal
0	Round wing indicator
1	Triangle wing indicator

16.3 osd_color <id> <alpha> <y> <cb> <cr>

Sets wing color.

Value	Minimal
id	0= 0° wing, 1= 180° wing
alpha	Transparence of wing 0=transparent, 1023 100% overlay
y	The color is in YUV format. Y component 0..1023
cb	The color is in YUV format. U component 0..1023
cr	The color is in YUV format. V component 0..1023

16.4 osd_wnd <hoffs> <voffs> <size>

Sets indicator window position

Value	Minimal
hoffs	Horizontal position (4...(1917-window size))
voffs	Vertical position (1..(1080-window size))
size	Window size

16.5 osd_speed <step>

Sets animation speed. Rotate life indicator every "<step>" * frames by 45°.

Value	Range
step	1...256