RGB-IR LED Panel Command Set

Steve Sawtelle  
Lead Electrical Engineer, ID&F  
(571) 209-4158

# System Overview

The command set for the RGB-IR LED panels is common to all versions of the 5” x 5” and 7”  
 x 7” panels designed at Janelia Research Campus of HHMI.

**Firmware Development**

The firmware is written in Arduino with Teensyduino extensions. Several libraries are used:

Cmd.h – This has been modified from the version available on Github to add features needed by the firmware. It processes commands sent over USB-Serial and is used to set up parameters and create actions.

Watchdog.h – This is a library written by Peter Polidoro to set up and control a watchdog timer in case the firmware hangs up.

EEPROM.h – This library is used to save setups in non-volatile memory, such as offsets and gains for each quadrant to make the quadrants light intensity more even (especially at low levels).

Several Teensy varieties are supported. The preferred boards are the Teensy 3.2 and 4.0. The LC can be used but has limited memory for Experiment Command Steps. The firmware will compile correctly based on the chosen processor.

The RGB firmware is command driven. The processing of commands is handled by the cmd.h library, which will call the appropriate function once the command line is parsed. These functions end in ‘cmd’. The commands can set up parameters, query settings, or cause action. Commands are ASCII and case insensitive. Parameters are separated by spaces or commas. The command is terminated with a linefeed and carriage return.

Pulse timing or a loaded experiment is based off a millisecond clock, so the basic accuracy of the board is one millisecond. The processing of pulse timing or a loaded experiment is controlled via a four-phase state machine. Each phase runs independently and is associated with one of the three color channels or digital channel. For each phase, an initial state is called, which sets up parameters and timing for the next state. A flag is set to note a change in state and the next state will wait until the last state has completed before it executes its function and sets up the next state.

There are several compiler defines that affect the operation of the firmware:

DEBUG – turns on debugging information on USB-serial output.

SYNC\_PIN\_OUT – if defined, the SYNC pin is set up as output, otherwise it set up as trigger input.

SYNC\_ACTIVE\_LOW – if defined, the SYNC pin is active low.

# Command Set

The host communications is via a USB-serial link. The Teensy serial driver must be loaded on the host (available from PJRC.com). The link is baud-independent. The communications protocol is based on an ASCII commands with various parameters. Commands are not case sensitive and must be terminated with a carriage return character (‘\r’, 0x0D). Basic error checking is done and ‘cmderr n’ is returned with ‘n’ being a negative integer that can be used to determine the type of error. Commands may return other information to the Host to indicate status. These are still under development.

NOTE: the 7” x 7” RGB panels do not have addressable ‘Quadrants’. For these boards, the ‘Quadrant’ value should be left blank or set to ‘1111’ to select all quadrants.

The following commands are supported:

**RESET** – reset all boards to their power-on state. This should always be sent at the beginning of a session or after power has been cycled.

**RED b** **p q** (or) - control for red (chrimson) LEDs

**CHR b** **p q**

**BLUE b p q** (or) – control for blue LEDs

**BLU b** **p q**

**GREEN b p q** (or) – control for green LEDs

**GRN b p q**

**IR b** **p q** – control for infrared LEDs

Where:

‘b’ is the brightness as a percentage (0-100). A floating-point value can be used.

‘p’ (optional) is the panel to be changed. No value (or 0) will change all panels. 1-4 will select that panel

‘q’ = quadrant(s): 0 = all, or bit pattern (“0011” means turn on quadrants 1 and 2)

These commands control the LED intensity for red, blue, green, or infrared, respectively. For RGB LEDs, very low values (<1%) may show different intensities across the LEDs since they are close to their turn-off state. The RGB LEDs use linear drivers, so the intensity is very linear down to low levels. Note that for RGB LEDs this command only sets intensity, it does not turn the LED on. Use ON and OFF to control LED state. For IR LEDs, the settings should be above 15% due to the way the driver for these LEDs works. To turn off IR, set the intensity to 0%, as IR LEDs do not have separate ON or OFF commands.

Note: If low, equal, intensities are needed, the maximum current can be modified with a hardware change (see Hardware Technical Document for the board), or the offset value for each quadrant can be adjusted by the Offset command.

Note: the current board design uses separate LEDs for each color, enabling one to choose any mix of available colors. If colors other than red, green, or blue are used, then on will need to keep track of which custom color is associated with each channel name.

**ON p q** – turn on RGB LEDs to their last setting, optional panel(s) and quadrant(s)

**OFF p q –** turn off RGB LEDs, optional panel(s) and quadrant(s)

When the RGB LEDs are set to an intensity, that value is stored. Use ON and OFF to turn RGB LEDs on and off.

‘p’ = panel(s): 0 = all, or 1, 2, 3, or 4 for an individual panel

‘q’ = quadrant(s): 0 = all, or bit pattern (“0011” means turn on quadrants 1 and 2)

**PULSE width, period, number, off, wait, iterations,** **color** - pulse setup command

The PULSE command sets up a pulse train to light Optogenetic LEDs that have been set up with an ON or PATT command. The pulse has: on delay, pulse width, period, number of cycles, off delay, wait time, and number of iterations. WAIT

width: The pulse width (on time) in milliseconds from 1 to 30000

period: The period of the pulse (on time plus off time) in milliseconds from 1 to 30000. If the width and period are equal, then the LEDs are on constantly during the pulse train.

number: The number of pulses in a pulse train. One pulse train starts with an on time and ends ‘number’ of pulse later after the last off time.

off: The off time after the pulse train completes in milliseconds from 0 to 30000. This provides a means to have a dead time between pulse trains if iterations are used. Note that the total off time from when the last pulse on time ends to the next pulse on time starts is equal to: off + period – width.

wait: Delay the start of the pulse sequence in seconds from 0 to 120. This delays the start of the very first pulse after the RUN command is sent. It is not repeated until RUN is resent. A floating-point value can be used here to set timing to the millisecond level.

iterations: The number of time the pulse train is repeated from 0 to 30000. A ‘0’ value runs the pulse train continuously until a STOP command is sent.

color: Which color the pulse command is for (‘R’, ‘G’, and/or ‘B’) – the three characters can be used in any combination. If different pulse setups are required for different colors, then use separate PULSE commands.

The diagram below shows how each parameter fits into the pulse train:

A diagram of a pulse command

Description automatically generated

**RUN** c- Start running the pulse sequence, optional color(s)

* c – color(s) any combination of ‘R’, ‘G’, ‘B’

**STOP** c- Stop running the pulse sequence and reset to the beginning, optional color(s)

* c – color(s) any combination of ‘R’, ‘G’, ‘B’

**PAUSE** – Pause the current pulse sequence

**MARKER** b – test the marker LEDs. Each marker is lit in turn for two seconds (red, blue, green)

**BLINK** t c – turn the marker LEDs on for ‘t’ milliseconds. This pattern will also be repeated at the beginning of Each experiment Step.

* c – color(s) any combination of ‘R’, ‘G’, ‘B’

**DIGITAL** p – set up digital outputs according to bit pattern ‘p’

**DON** – turn on digital outputs that were enabled with the DIGITAL command

**DOFF** – turn off digital outputs

**SYNC** f – send sync pulses out Sync connector at frequency ‘f’ (floating point). If ‘f’ = 0, then turn of sync output. If ‘f’ is blank, return current sync rate.

**SYNCON** – start sync signal at previously specified rate

**???** - Returns the version number and other parameters

**HELP** - Returns a list of the available commands

**EXPERIMENT COMMANDS**

A complete experiment can be downloaded to the board, consisting of individual steps that run one after the other. Steps can be loaded in any order, with ‘1’ being the first step. An experiment will stop at the end of the steps (or the first missing step).

**addOneStep**(parameters): upload one experiment step to the buffer with the following parameters:

StepNumber,

RedIntensity, RedPulsePeriod, RedPulseWidth, RedPulseNum, RedOffTime, RedIteration,

GrnIntensity, GrnPulsePeriod, GrnPulseWidth, GrnPulseNum, GrnOffTime, GrnIteration,

BluIntensity, BluPulsePeriod, BluPulseWidth, BluPulseNum, BluOffTime, BluIteration,

DelayTime, Duration(step), [optional] pattern )

Where:

The Intensity, Period, Width, Number, Off Time, Iteration are the same as the PULSE command

Delay Time is the same as the PULSE command but is common for all three colors

Duration is the total duration of the step

Pattern is the quadrant pattern use (16 bits)

**getExperimentsteps** (or **GS**): returns a list of entered steps

**removeAllSteps**: remove all steps

**runExperiment** (or **RX**): run the entered experiment

**stopExperiment** (or **SX**): stop the experiment in progress

**getExperimentStatus** (or **GX**): returns the current step being processed during a run

**stepOrder** n: add a series of numbers, one at a time, designating the sequence the entered Experiment Steps should be run in (if not used, the experiment is run in the step sequence, one time). For example, if 2 steps are entered, sending:

steporder 2

steporder 1

steporder 1

steporder 2

will run step 2, then step 1, then step 1 again, then step 2 when the runExperiment command is sent.

Sending stepOrder 0 will reset the steps, sending stepOrder with no parameter will list the step order.

**CALIBRATION COMMANDS**

The RGB\_IR board should have uniform light output. However, the 5x5 board, with separate quadrant control, may show some differences at low light levels. If an experiment requires background subtraction, then the brightness of each quadrant can be adjusted so that, at a given IR setting, the backlight is more uniform. The light levels can be calibrated using the OFFSET and GAIN commands. These values are stored in non-volatile memory and will be loaded at power up. The offset values represent digital-to-analog converter (DAC) counts. The DAC can range from 0 to 4095. The brightness is calculated as follows:

output value = set value \* GAIN + OFFSET

**Gain**

The GAIN command was added to enter in gain values. These values default to 1.0 and can range from 0.8 and 1.2, but generally, gain changes of 5% or less are enough. A new board has the gains all set to 1.0. There are sixteen gain values: one for each quadrant and for each color. The IR gain is probably the most important as it sets the continuous backlight that is used for automatic tracking. The color LEDs are normally used for optogenetics, and small differences are not as critical.

**GAIN** c, q, g – set the gain value for a quadrant

Where:

‘c’ is color to be adjusted (‘R’, ‘B’, ‘G’, or ‘I’).

‘q’ = quadrant(s): 0 , 1, 2, or 3

‘g’ is the gain multiplier (0.8 to 1.2)

If the GAIN command is sent without parameters, the current settings are returned:

GAIN

BLU 1.00 1.00 1.00 1.00

GRN 1.00 1.00 1.00 1.00

RED 1.00 1.00 1.00 1.00

IR 1.00 1.00 1.00 1.00

To change the gain of a quadrant, the GAIN command has parameters of Color, Quadrant, and Gain:

GAIN BLUE 0 1.1

The color can be spelled out or a shortened version can be used (RED, GRN, BLU, IR) or, simply, just the first letter is needed (R,G,B,I). The commands are case insensitive.

The Quadrant is a value between 0 and 3 inclusive, starting with the quadrant closest to where the cables attach, and increasing clockwise:

A picture containing graphical user interface

Description automatically generated

An example is the Fly Bubble Rig, where the quadrants can be counted relative to the hole at one corner:

A picture containing chart

Description automatically generated

The Gain is a floating-point value between 0.8 and 1.2 inclusive. Lower gain reduces the relative brightness of the selected quadrant.

An Intensity Command for the chosen color and quadrant must be sent to see the effect of a gain change. So, if IR is changed for quadrant 1, then sending:

IR 30

Will update all the IR’s with appropriate gains.

Choosing the correct gains is somewhat interactive. The easiest method is to set the chosen color to a set intensity for all quadrants slightly below the saturation level, for example:

IR 40

Log a video or single frame of the board using BIAS or other camera software. Open that file in FIJI or other analysis software. Look at profiles of adjacent quadrants:

A screenshot of a computer

Description automatically generated

Here a profile line was drawn through the quadrants 2 and 3. Do this to the other three combinations. It will be clear which quadrants are different than the others. If one quadrant is higher intensity, it’s gain can be lowered, or if one quadrant is lower, it’s gain can be increased. 5% changes are usually enough. Change the IR level to make the gain take effect and repeat until the profiles are sufficiently uniform.

Once the profiles are uniform, the gain changes are complete. The GAIN command, without parameters, can be used to list the changes made. These changes are stored in non-volatile memory, so they are reinstated after a power cycle or reset. However, if the device is reprogrammed with updated firmware, the values may be lost. It would be wise to make note of the values once changes are finalized.

**Offset**

A zero offset can be applied to each quadrant in a similar way to gain. This is primarily used to get more even brightness at very low settings (less than 2%). Offset values can range between 0 and 20 and is in units of digital-to-analog converter (DAC) counts. The DAC has a range of 0 to 4095 counts.

**OFFSET** c, q, o – set the offset value for a quadrant.

Where:

‘c’ is color to be adjusted (‘R’, ‘B’, ‘G’, or ‘I’).

‘q’ = quadrant(s): 0, 1, 2 or 3

‘o’ is the offset value (0 to 20)

If no parameters are specified, then the current settings are returned.

# Intraboard Commands

This section describes the commands sent between boards. These commands are not directly available to the host but are included here to provide detail on how host commands are relayed to the other panels. The intraboard communications is based on a TTL serial link using the Teensy Serial1 port. Each panel has an input and output line. All boards have a Teensy installed, but only one (main panel) is connected to the host. The host sends all commands to all or any other panel(s) via the main panel. The main panel interprets the command from the host, and if is destined for one or more other panels, it sends the command on via Serial1. The incoming serial connection to the panel is always connected to the micro’s Serial1 receive port and the outgoing serial connection leaving a panel is routed through a TX/RX switch to route signals. The panel’s output signal can be switched between a direct connection to the input (Parallel Mode) or connected to the panel’s Serial1 transmit port (Series Mode).

Initially, all panels boot up in Parallel Mode. On reset, the main panel will send a Series command to put the attached panels in daisy-chain mode. This is necessary so that the boards can enumerate so they know what position they are in. An initial Enumerate1 command by the main panel propagates to the second panel, which configure itself as panel 2, and it the sends out an Enumerate2 to the second panel. This panel then configures itself as panel 3, and sends an Enumerate3 command to the last panel. After allowing time for the commands to propagate, the main panel sends a Parallel command to set the panels back to normal operation. Then any commands from the main panel are passed directly to all panels simultaneously to minimize delay of commands between panels. Each command is a single 8-bit value. The 256 possible commands are organized as follows:

Commands

00pp bbbb Turn off selected quadrants on selected panels

pp = panel (00 = all, 01 = 1, 10 = 2, 11 = 3)

bbbb = quadrant bits quadrant bits (0b0001 is upper left, 0b0010 upper right, 0b1000 lower right, 0b0100 lower left

01pp bbbb Turn on selected quadrants on selected panels

pp = panel (00 = all, 01 = 1, 10 = 2, 11 = 3)

bbbb = quadrant bits quadrant bits (0b0001 is upper left, 0b0010 upper right, 0b1000 lower right, 0b0100 lower left

10pp qqcc Set intensity command of selected panel(s), quadrants(s) , and color(s) to last Intensity value (0-4095)

pp = panel (00 = all, 01 = 1, 10 = 2, 11 = 3)

qq = quadrant number 00 is upper left, etc.

cc = color (00 = BLU, 01 = RED, 10 = GRN, 11 = IR)

1100 iiii Set the lower 4 bits of the intensity value (or lowest 4 bits of digital pattern)

iiii = intensity value nibble

1101 iiii Set the middle 4 bits of the intensity value (or middle 4 bits of digital pattern)

iiii = intensity value nibble

1110 iiii Set the highest 4 bits of the intensity value (or highest 4 bits of digital pattern)

iiii = intensity value nibble

1111 00xx Enumerate

xx = 01 sent by main board, 10 sent by next panel, 11 sent by third panel

1111 0100 Set RX/TX switch to Series mode

1111 0101 Set RX/TX switch to Parallel mode

1111 1ddd Turn color DACs on or off on all panels (used for fast response, especially for pulsing

ddd = color DACs: bit 2 = red, bit 1 = green, bit 0 = blue

1111 0000 When in parallel mode, turns red marker off

1111 0000 When in parallel mode, turns green marker off

1111 0000 When in parallel mode, turns blue marker off

1111 0111 When in parallel mode, sets digital pin pattern from last pattern sent with Intensity command

1111 0001 When in parallel mode, turn digital outputs on

1111 0011 When in parallel mode, turn digital outputs off

1111 0101 When in parallel mode, turn digital marker off

# 