KMK FW - RGB/Underglow/NeoPixel

Want your keyboard to shine? Add some lights!

CircuitPython

This does require the NeoPixel library from Adafruit. It is part of the Adafruit CircuitPython Bundle. Simply put this in the "root" of your CircuitPython device. If unsure, it's the folder with `main.py` in it, and should be the first folder you see when you open the device.

Currently we support the following addressable LEDs:

- WS2811, WS2812, WS2812B, WS2812C, etc.
- SK6812, SK6812MINI, SK6805

Color Selection

KMK uses Hue, Saturation, and Value to select colors rather than RGB. The color wheel below demonstrates how this works.

Changing the **Hue** cycles around the circle. Changing the **Saturation** moves between the inner and outer sections of the wheel, affecting the intensity of the color. Changing the **Value** sets the overall brightness.

Enabling the extension

The only required values that you need to give the RGB extension would be the board pin for the data line, and the number of pixels/LED's. If using a split keyboard, this number is per side, and not the total of both sides.

```
import board
from kmk.extensions.RGB import RGB

rgb = RGB(pixel_pin=board.GP14, num_pixels=27)
keyboard.extensions.append(rgb)
```

[Keycodes]

Key	Aliases	Description
`KC.RGB_TOG`		Toggles RGB

Key	Aliases	Description
KC.RGB_HUI		Increase Hue
KC.RGB_HUD		Decrease Hue
KC.RGB_SAI		Increase Saturation
KC.RGB_SAD		Decrease Saturation
KC.RGB_VAI		Increase Value
KC.RGB_VAD		Decrease Value
KC.RGB_ANI		Increase animation speed
KC.RGB_AND		Decrease animation speed
KC.RGB_MODE_PLAIN	RGB_M_P	Static RGB
KC.RGB_MODE_BREATHE	RGB_M_B	Breathing animation
KC.RGB_MODE_RAINBOW	RGB_M_R	Rainbow animation
KC.RGB_MODE_BREATHE_RAINBOW	RGB_M_BR	Breathing rainbow animation
KC.RGB_MODE_KNIGHT	RGB_M_K	Knight Rider animation
KC.RGB_MODE_SWIRL	RGB_M_S	Swirl animation

Configuration

Define	Default	Description
rgb.num_pixels		The number of LEDs connected
rgb.rgb_order	(1, 0, 2)	The order of the pixels R G B, and optionally white. Example(1, 0, 2, 3)
rgb.hue_step	10	The number of steps to cycle through the hue by
rgb.sat_step	17	The number of steps to change the saturation by
rgb.val_step	17	The number of steps to change the brightness by
rgb.hue_default	0	The default hue when the keyboard boots
rgb.sat_default	255	The default saturation when the keyboard boots
rgb.val_default	255	The default value (brightness) when the keyboard boots
rgb.val_limit	255	The maximum brightness level

Built-in Animation Configuration

Define	Default	Description
rgb.breathe_center	1.5	Used to calculate the curve for the breathing animation. Anywhere from 1.0 - 2.7 is valid
rgb.knight_effect_length	4	The number of LEDs to light up for the "Knight" animation

Functions

If you want to create your own animations, or for example, change the lighting in a macro, or a layer switch, here are some functions that are available.

Function	Description
rgb.set_hsv_fill(hue, sat, val)	Fills all LED's with HSV values

Function	Description
`rgb.set_hsv(hue, sat, val, index)`	Sets a single LED with HSV value
`rgb.set_rgb_fill((r, g, b))`	Fills all LED's with RGB(W) values
`rgb.set_rgb((r, g, b), index)`	Set's a single LED with RGB(W) values
`rgb.increase_hue(step)`	Increases hue by a given step
`rgb.decrease_hue(step)`	Decreases hue by a given step
`rgb.increase_sat(step)`	Increases saturation by a given step
`rgb.decrease_sat(step)`	Decreases saturation by a given step
`rgb.increase_val(step)`	Increases value (brightness) by a given step
`rgb.decrease_val(step)`	Decreases value (brightness) by a given step
`rgb.increase_ani()`	Increases animation speed by 1. Maximum 10
`rgb.decrease_ani()`	Decreases animation speed by 1. Minimum 10
`rgb.off()`	Turns all LED's off
`rgb.show()`	Displays all stored configuration for LED's

Direct variable access

Define	Default	Description
`rgb.hue`	`0`	Sets the hue from 0-255
`rgb.sat`	`255`	Sets the saturation from 0-255
`rgb.val`	`255`	Sets the brightness from 0-255
`rgb.reverse_animation`	`False`	If true, some animations will run in reverse. Can be safely used in user animations
`rgb.animation_mode`	`static`	This can be changed to any modes included, or to something custom for user animations. Any string is valid
`rgb.animation_speed`	`1`	Increases animation speed of most animations. Recommended 1-5, Maximum 10.

```
from kmk.extensions.rgb import AnimationModes
rgb = RGB(pixel_pin=rgb_pixel_pin,
        num_pixels=27
        val_limit=100,
        hue_default=0,
        sat_default=100,
        rgb_order=(1, 0, 2),  # GRB WS2812
        val_default=100,
        hue_step=5,
        sat_step=5,
        val_step=5,
        animation_speed=1,
        breathe_center=1,  # 1.0-2.7
        knight_effect_length=3,
        animation_mode=AnimationModes.STATIC,
```

```
        reverse_animation=False,
        refresh_rate=60,
        )
```

Hardware Modification

To add RGB LED's to boards that don't support them directly, you will have to add a 3 wires. The power wire will run on 3.3v or 5v (depending on the LED), ground, and data pins will need added to an unused pin on your microcontroller unless your keyboard has specific solder points for them. With those 3 wires connected, set the `pixel_pin` as described above, and you are ready to use your RGB LED's/NeoPixel.

Troubleshooting

Incorrect colors

If your colors are incorrect, check the pixel order of your specific LED's. Here are some common ones. * WS2811, WS2812, WS2812B, WS2812C are all GRB (1, 0, 2) * SK6812, SK6812MINI, SK6805 are all GRB (1, 0, 2) * NeoPixels will vary depending on which one you buy. It will be listed on the product page.

Lights don't turn on

Make sure that your board supports LED backlight by checking for a line with `PIXEL_PIN`. If it does not, you can add it to your keymap. If you added the LED's yourself, you will also need to set `num_pixels` to the number of installed LED's in total.

Alternate LED chipsets

Not all RGB LEDs are compatible with NeoPixels. To support these, the RGB extension accepts an instance of a `Pixelbuf`-compatible object as an optional parameter. If supplied, `pixel_pin` is ignored and the supplied `Pixelbuf` is used instead of creating a NeoPixel object. The RGB extension will figure out LED count from the pixel buffer length if not passed explicitly.

This works easily with APA102 ("DotStar") LEDs, but for most other RGB LED chipsets you will need to provide a wrapper to match the expected interface.

A simple example using APA102:

```
import adafruit_dotstar
from kmk.extensions.RGB import RGB
from kb import rgb_pixel_pin  # This can be imported or defined manually

_LED_COUNT=12
pixels = adafruit_dotstar.DotStar(board.SCK, board.MOSI, _LED_COUNT)
```

```
rgb = RGB(pixel_pin=None, pixels=pixels)
keyboard.extensions.append(rgb)
```

Multiple PixelBuffer

Similar to alternate drivers, the RGB module supports passing multiple `Pixelbuf` objects as an iterable.

```
from kmk.extensions.RGB import RGB

pixels = (
    Neopixel(...),
    DotStar(...),
    CustomPixelBuf(...)
)

rgb = RGB(pixel_pin=None, pixels=pixels)
keyboard.extensions.append(rgb)
```