

RoutinesRGB
v1.9.8

Generated by Doxygen 1.8.8

Sun Jul 3 2016 16:02:53

Contents

1	Module Index	1
1.1	Modules	1
2	Class Index	3
2.1	Class List	3
3	File Index	5
3.1	File List	5
4	Module Documentation	7
4.1	Getters and Setters	7
4.1.1	Detailed Description	7
4.1.2	Function Documentation	7
4.1.2.1	barSize	7
4.1.2.2	blinkSpeed	7
4.1.2.3	blue	8
4.1.2.4	brightness	8
4.1.2.5	color	8
4.1.2.6	customColorCount	8
4.1.2.7	fadeSpeed	8
4.1.2.8	green	8
4.1.2.9	mainColor	8
4.1.2.10	red	8
4.1.2.11	setColor	8
4.1.2.12	setCustomColorCount	8
4.1.2.13	setMainColor	9
4.2	Single Color Routines	10
4.2.1	Detailed Description	10
4.2.2	Function Documentation	10
4.2.2.1	singleBlink	10
4.2.2.2	singleFade	10
4.2.2.3	singleGlimmer	10

4.2.2.4	singleSawtoothFade	11
4.2.2.5	singleSolid	11
4.2.2.6	singleWave	11
4.3	Multi Colors Routines	12
4.3.1	Detailed Description	12
4.3.2	Function Documentation	12
4.3.2.1	multiBarsMoving	12
4.3.2.2	multiBarsSolid	12
4.3.2.3	multiFade	12
4.3.2.4	multiGlimmer	13
4.3.2.5	multiRandomIndividual	13
4.3.2.6	multiRandomSolid	13
4.4	These methods can be called after a routine is chosen	14
5	Class Documentation	15
5.1	RoutinesRGB Class Reference	15
5.1.1	Detailed Description	16
5.1.2	Constructor & Destructor Documentation	16
5.1.2.1	RoutinesRGB	16
5.1.3	Member Function Documentation	17
5.1.3.1	applyBrightness	17
5.1.3.2	drawColor	17
5.1.3.3	resetToDefaults	17
6	File Documentation	19
6.1	ColorPresets.h File Reference	19
6.1.1	Detailed Description	19
6.2	LightingProtocols.h File Reference	19
6.2.1	Detailed Description	20
6.2.2	Enumeration Type Documentation	20
6.2.2.1	EColorGroup	20
6.2.2.2	ELightingRoutine	21
6.2.2.3	EPacketHeader	22
Index		23

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

Getters and Setters	7
Single Color Routines	10
Multi Colors Routines	12
These methods can be called after a routine is chosen	14

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

[RoutinesRGB](#)

An Arduino library that provides a set of RGB lighting routines for compatible LED array hardware [15](#)

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

ColorPresets.h	19
LightingProtocols.h	19
RoutinesRGB.h	??

Chapter 4

Module Documentation

4.1 Getters and Setters

Functions

- bool [RoutinesRGB::setMainColor](#) (byte r, byte g, byte b)
- void [RoutinesRGB::setColor](#) (uint16_t colorIndex, byte r, byte g, byte b)
- void [RoutinesRGB::setCustomColorCount](#) (uint8_t count)
- uint8_t [RoutinesRGB::customColorCount](#) ()
- void [RoutinesRGB::brightness](#) (uint8_t brightness)
- void [RoutinesRGB::fadeSpeed](#) (uint8_t fadeSpeed)
- void [RoutinesRGB::blinkSpeed](#) (uint8_t blinkSpeed)
- void [RoutinesRGB::barSize](#) (uint8_t barSize)
- Color [RoutinesRGB::mainColor](#) ()
- Color [RoutinesRGB::color](#) (uint16_t i)
- uint8_t [RoutinesRGB::red](#) (uint16_t i)
- uint8_t [RoutinesRGB::green](#) (uint16_t i)
- uint8_t [RoutinesRGB::blue](#) (uint16_t i)

4.1.1 Detailed Description

These are the getters and setters for [RoutinesRGB](#) that are used to control the settings and the colors.

4.1.2 Function Documentation

4.1.2.1 void [RoutinesRGB::barSize](#) (uint8_t *barSize*)

Sets the size of bars in routines that use them. Bars are groups of LEDs that all display the same color. The routines SingleWave, MultiBarsSolid, and MultiBarsMoving use them.

a number greater than 0 and less than the number of LEDs being used.

4.1.2.2 void [RoutinesRGB::blinkSpeed](#) (uint8_t *blinkSpeed*)

Sets how many updates to wait before changing the light state in the blink routine and in routines that switch between solid colors.

Parameters

<i>blinkSpeed</i>	a value between 1 and 255 representing how fast to blink. A value of 1 will make it blink on every frame, which may be too fast when used with other routines.
-------------------	--

4.1.2.3 `uint8_t RoutinesRGB::blue (uint16_t i)`

Retrieve the b value at a given index in the buffer.

4.1.2.4 `void RoutinesRGB::brightness (uint8_t brightness)`

Set the brightness between 0 and 100. 0 is off, 100 is full power.

4.1.2.5 `RoutinesRGB::Color RoutinesRGB::color (uint16_t i)`

Retrieve the color at the given index.

4.1.2.6 `uint8_t RoutinesRGB::customColorCount ()`

Retrieve the amount of colors that are used from the custom array.

4.1.2.7 `void RoutinesRGB::fadeSpeed (uint8_t fadeSpeed)`

Sets the speed of routines that fade between colors between 1 and 100. A fade speed of 1 is the slowest possible fade.

4.1.2.8 `uint8_t RoutinesRGB::green (uint16_t i)`

Retrieve the g value at a given index in the buffer.

4.1.2.9 `RoutinesRGB::Color RoutinesRGB::mainColor ()`

Retrieve the main color, which is used for single color routines.

4.1.2.10 `uint8_t RoutinesRGB::red (uint16_t i)`

Retrieve the r value at a given index in the buffer.

4.1.2.11 `void RoutinesRGB::setColor (uint16_t colorIndex, byte r, byte g, byte b)`

Set the color in the custom color array at the provided index. colorIndex must be less than the size of the custom color array or else it won't have any effect.

4.1.2.12 `void RoutinesRGB::setCustomColorCount (uint8_t count)`

Sets the amount of colors used in custom multi color routines. The value given must be less than the size of the custom array or else it will be set to use the entire array.

4.1.2.13 `bool RoutinesRGB::setMainColor (byte r, byte g, byte b)`

Sets the color used for single color routines. This is automatically called by each routine. Returns false if the new main color matches the previous main color.

Returns

true if a new color is set, false if the input matches the current color.

4.2 Single Color Routines

Functions

- void `RoutinesRGB::singleSolid` (uint8_t red, uint8_t green, uint8_t blue)
- void `RoutinesRGB::singleBlink` (uint8_t red, uint8_t green, uint8_t blue)
- void `RoutinesRGB::singleWave` (uint8_t red, uint8_t green, uint8_t blue)
- void `RoutinesRGB::singleGlimmer` (uint8_t red, uint8_t green, uint8_t blue, uint8_t percent=20)
- void `RoutinesRGB::singleFade` (uint8_t red, uint8_t green, uint8_t blue, bool isSine)
- void `RoutinesRGB::singleSawtoothFade` (uint8_t red, uint8_t green, uint8_t blue, bool fadeIn)

4.2.1 Detailed Description

These routines each take a R, G, and B value as parameters to generate a color. This color is the only color used by the routine.

All routines except singleSolid should be called repeatedly on a loop for their full effect. The speed of the loop determines how fast the LEDs update.

4.2.2 Function Documentation

4.2.2.1 void `RoutinesRGB::singleBlink` (uint8_t *red*, uint8_t *green*, uint8_t *blue*)

Switches between ON and OFF states using the provided color.

Parameters

<i>red</i>	strength of red LED, between 0 and 255
<i>green</i>	strength of green LED, between 0 and 255
<i>blue</i>	strength of blue LED, between 0 and 255

4.2.2.2 void `RoutinesRGB::singleFade` (uint8_t *red*, uint8_t *green*, uint8_t *blue*, bool *isSine*)

Fades the LEDs in and out based on the provided color. Can fade in two ways: linear and sine. If isSine is set to false, the interval between each update is constant. If isSine is true, a sine wave is used to generate the intervals, resulting in lights that stay on near their full brightness for longer.

Parameters

<i>red</i>	strength of red LED, between 0 and 255
<i>green</i>	strength of green LED, between 0 and 255
<i>blue</i>	strength of blue LED, between 0 and 255
<i>isSine</i>	if true, a sine wave is used, if false, constant intervals are used.

4.2.2.3 void `RoutinesRGB::singleGlimmer` (uint8_t *red*, uint8_t *green*, uint8_t *blue*, uint8_t *percent* = 20)

Set every LED to the provided color. A subset of the LEDs based on the percent parameter will be less bright than the rest of the LEDs.

Parameters

<i>red</i>	strength of red LED, between 0 and 255
<i>green</i>	strength of green LED, between 0 and 255
<i>blue</i>	strength of blue LED, between 0 and 255
<i>percent</i>	determines how many LEDs will be slightly dimmer than the rest, between 0 and 100

4.2.2.4 void RoutinesRGB::singleSawtoothFade (uint8_t *red*, uint8_t *green*, uint8_t *blue*, bool *fadeIn*)

If *fadeIn* is true, the LEDs start with a brightness value of 0 and each update raises the brightness by a constant value. When it reaches maximum brightness, it resets the brightness back to 0 and repeats the fade in. If *fadeIn* is set to false, it does the opposite; it starts at full brightness and then fades to darkness.

Parameters

<i>red</i>	strength of red LED, between 0 and 255
<i>green</i>	strength of green LED, between 0 and 255
<i>blue</i>	strength of blue LED, between 0 and 255
<i>fadeIn</i>	if true, it fades from darkness to maximum brightness, if false, it fades from maximum brightness to darkness.

4.2.2.5 void RoutinesRGB::singleSolid (uint8_t *red*, uint8_t *green*, uint8_t *blue*)

Set every LED to the provided color.

Parameters

<i>red</i>	strength of red LED, between 0 and 255
<i>green</i>	strength of green LED, between 0 and 255
<i>blue</i>	strength of blue LED, between 0 and 255

4.2.2.6 void RoutinesRGB::singleWave (uint8_t *red*, uint8_t *green*, uint8_t *blue*)

Uses the provided color and generates groups of the color in increasing levels of brightness. On each update, the LEDs move one index to the right. This creates a wave/scrolling effect.

Parameters

<i>red</i>	strength of red LED, between 0 and 255
<i>green</i>	strength of green LED, between 0 and 255
<i>blue</i>	strength of blue LED, between 0 and 255

4.3 Multi Colors Routines

Functions

- void `RoutinesRGB::multiGlimmer` (`EColorGroup` colorGroup, uint8_t percent=20)
- void `RoutinesRGB::multiFade` (`EColorGroup` colorGroup)
- void `RoutinesRGB::multiRandomIndividual` (`EColorGroup` colorGroup)
- void `RoutinesRGB::multiRandomSolid` (`EColorGroup` colorGroup)
- void `RoutinesRGB::multiBarsSolid` (`EColorGroup` colorGroup, uint8_t barSizeSetting)
- void `RoutinesRGB::multiBarsMoving` (`EColorGroup` colorGroup, uint8_t barSizeSetting)

4.3.1 Detailed Description

These routines use multiple colors. They all take the parameter of `colorGroup` which is used to determine which set of colors to use. The custom color array is `eCustom`, all other values for `colorGroup` come from groups of preset colors. Go to the project's github for a full list of the colorGroups and their corresponding values.

All routines except `multiBarsSolid` should be called repeatedly on a loop for their full effect. The speed of the loop determines how fast the LEDs update.

4.3.2 Function Documentation

4.3.2.1 void `RoutinesRGB::multiBarsMoving` (`EColorGroup` colorGroup, uint8_t barSizeSetting)

Provides a similar effect as `multiBarSolid`, but the alternating patches move up one LED index on each frame update to create a "scrolling" effect.

Parameters

<i>colorGroup</i>	the color group to use for the routine. <code>eCustom</code> is the custom array, all other values are preset groups.
<i>barSize</i>	how many LEDs before switching to the other bar.

4.3.2.2 void `RoutinesRGB::multiBarsSolid` (`EColorGroup` colorGroup, uint8_t barSizeSetting)

Uses the chosen color group to set the LEDs in alternating patches with a size of `barSize`.

Parameters

<i>colorGroup</i>	the color group to use for the routine. <code>eCustom</code> is the custom array, all other values are preset groups.
<i>barSize</i>	how many LEDs before switching to the other bar.

4.3.2.3 void `RoutinesRGB::multiFade` (`EColorGroup` colorGroup)

Fades between all the colors used by the color group.

Parameters

<i>colorGroup</i>	the color group to use for the routine. <code>eCustom</code> is the custom array, all other values are preset groups.
-------------------	---

4.3.2.4 void RoutinesRGB::multiGlimmer (EColorGroup *colorGroup*, uint8_t *percent* = 20)

This method uses its percent parameter to dim LEDs randomly, similar to the standard glimmer mode. It also uses the percent to randomly change the color of select LEDs to a color in the chosen color group. The base color is the first from the chosen color group.

Parameters

<i>colorGroup</i>	the color group to use for the routine. eCustom is the custom array, all other values are preset groups.
<i>percent</i>	percent of LEDs that will get the glimmer applied, between 0 and 100

4.3.2.5 void RoutinesRGB::multiRandomIndividual (EColorGroup *colorGroup*)

sets each individual LED as a random color from the chosen color group.

Parameters

<i>colorGroup</i>	the color array to use for the routine. eCustom is the custom array, all other values are colorGroup arrays.
-------------------	--

4.3.2.6 void RoutinesRGB::multiRandomSolid (EColorGroup *colorGroup*)

A random color is chosen from the chosen color group and applied to each LED.

Parameters

<i>colorGroup</i>	the color group to use for the routine. eCustom is the custom array, all other values are preset groups.
-------------------	--

4.4 These methods can be called after a routine is chosen

but before the routines get displayed to the LEDs. They add special effects to the routines.

Chapter 5

Class Documentation

5.1 RoutinesRGB Class Reference

An Arduino library that provides a set of RGB lighting routines for compatible LED array hardware.

```
#include <RoutinesRGB.h>
```

Public Member Functions

- [RoutinesRGB](#) (uint16_t ledCount)
- void [resetToDefaults](#) ()
- bool [setMainColor](#) (byte r, byte g, byte b)
- void [setColor](#) (uint16_t colorIndex, byte r, byte g, byte b)
- void [setCustomColorCount](#) (uint8_t count)
- uint8_t [customColorCount](#) ()
- void [brightness](#) (uint8_t brightness)
- void [fadeSpeed](#) (uint8_t fadeSpeed)
- void [blinkSpeed](#) (uint8_t blinkSpeed)
- void [barSize](#) (uint8_t barSize)
- Color [mainColor](#) ()
- Color [color](#) (uint16_t i)
- uint8_t [red](#) (uint16_t i)
- uint8_t [green](#) (uint16_t i)
- uint8_t [blue](#) (uint16_t i)
- void [singleSolid](#) (uint8_t [red](#), uint8_t [green](#), uint8_t [blue](#))
- void [singleBlink](#) (uint8_t [red](#), uint8_t [green](#), uint8_t [blue](#))
- void [singleWave](#) (uint8_t [red](#), uint8_t [green](#), uint8_t [blue](#))
- void [singleGlimmer](#) (uint8_t [red](#), uint8_t [green](#), uint8_t [blue](#), uint8_t percent=20)
- void [singleFade](#) (uint8_t [red](#), uint8_t [green](#), uint8_t [blue](#), bool isSine)
- void [singleSawtoothFade](#) (uint8_t [red](#), uint8_t [green](#), uint8_t [blue](#), bool fadeIn)
- void [multiGlimmer](#) (EColorGroup colorGroup, uint8_t percent=20)
- void [multiFade](#) (EColorGroup colorGroup)
- void [multiRandomIndividual](#) (EColorGroup colorGroup)
- void [multiRandomSolid](#) (EColorGroup colorGroup)
- void [multiBarsSolid](#) (EColorGroup colorGroup, uint8_t barSizeSetting)
- void [multiBarsMoving](#) (EColorGroup colorGroup, uint8_t barSizeSetting)
- void [applyBrightness](#) ()
- bool [drawColor](#) (uint16_t i, uint8_t [red](#), uint8_t [green](#), uint8_t [blue](#))

5.1.1 Detailed Description

An Arduino library that provides a set of RGB lighting routines for compatible LED array hardware.

Version

v1.9.8

Date

July 3, 2016

Author

Tim Seemann

Copyright

MIT License

This library has been tested with SeeedStudio Rainbowduinos, quite a few of the Adafruit Neopixels products, and a standard RGB LED. Sample code is provided in the git repo for all tested hardware in the samples folder of the git repository.

If you are starting a project from scratch, first you'll need to make a global object in the arduino sketch:

```
RoutinesRGB routines = RoutinesRGB(LED_COUNT);
```

where `LED_COUNT` is the number of LEDs in your array.

The library produces lighting routines based on the functions used and stores the routine in its internal buffers. These buffers can then be accessed by getters and displayed on the LED hardware. For routines that change over time, this process should be repeated on a loop. For example, here is how you would make a red blinking light with the library and a Neopixels board:

First, call this function to store the routine in the library's internal buffers:

```
routines.singleBlink(255, 0, 0);
```

Then, update the LED array with the values from the library's RGB buffer. The way to do this will vary from hardware to hardware, but for a NeoPixels sample, it would look something like this:

```
routines.applyBrightness(); // Optional. sets how bright the LEDs shine, based off of the
                             brightness setting.
for (int x = 0; x < LED_COUNT; x++) {
    pixels.setPixelColor(x, pixels.Color(routines.red(x),
                                         routines.green(x),
                                         routines.blue(x)));
}
pixels.show();
```

By this point, the LEDs should be showing red. To achieve the blink effect, put both of these in your `loop()` function and then put a delay between updates. This delay will be used to determine how fast the LED's blink.

5.1.2 Constructor & Destructor Documentation

5.1.2.1 RoutinesRGB::RoutinesRGB (uint16_t ledCount)

Required constructor. The library should be stored in global memory and allocated only once at startup.

It will allocate `4 * ledCount` bytes.

Parameters

<i>ledCount</i>	number of individual RGB LEDs.
-----------------	--------------------------------

5.1.3 Member Function Documentation

5.1.3.1 void RoutinesRGB::applyBrightness ()

This function takes the [brightness\(\)](#) value given to the routines object and applies it to every LED. Relatively speaking, this is a pretty expensive operation so it is left optional.

5.1.3.2 bool RoutinesRGB::drawColor (uint16_t *i*, uint8_t *red*, uint8_t *green*, uint8_t *blue*)

Attempts to draw the color provided on the index provided.

Parameters

<i>i</i>	the index of the LED that you want to change. Must be less than the total amount of LEDs or else it will return false.
<i>red</i>	the new red value of the LED, between 0 and 255.
<i>green</i>	the new green value of the LED, between 0 and 255.
<i>blue</i>	the new blue value of the LED, between 0 and 255.

Returns

true if index exists and the color was drawn, false otherwise.

5.1.3.3 void RoutinesRGB::resetToDefaults ()

Resets all internal values to the original values.

Chapter 6

File Documentation

6.1 ColorPresets.h File Reference

```
#include <avr/pgmspace.h>
```

6.1.1 Detailed Description

Version

v1.9.8

Date

July 3, 2016

Author

Tim Seemann

Copyright

MIT License

These color presets are stored in program memory and loaded into a buffer when accessed. This makes the presets read-only, but in return, it allows them to take a much smaller hit on SRAM usage.

6.2 LightingProtocols.h File Reference

Enumerations

- enum `ELightingRoutine` {
 `eOff`, `eSingleSolid`, `eSingleBlink`, `eSingleWave`,
 `eSingleGlimmer`, `eSingleLinearFade`, `eSingleSineFade`, `eSingleSawtoothFadeIn`,
 `eSingleSawtoothFadeOut`, `eMultiGlimmer`, `eMultiFade`, `eMultiRandomSolid`,
 `eMultiRandomIndividual`, `eMultiBarsSolid`, `eMultiBarsMoving` }
- enum `EColorGroup` {
 `eCustom`, `eWater`, `eFrozen`, `eSnow`,
 `eCool`, `eWarm`, `eFire`, `eEvil`,
 `eCorrosive`, `ePoison`, `eRose`, `ePinkGreen`,
 `eRedWhiteBlue`, `eRGB`, `eCMY`, `eSixColor`,
 `eSevenColor`, `eAll` }

- enum `EPacketHeader` {
`eModeChange`, `eMainColorChange`, `eCustomArrayColorChange`, `eBrightnessChange`,
`eSpeedChange`, `eCustomColorCountChange`, `eIdleTimeoutChange`, `eResetSettingsToDefaults` }

6.2.1 Detailed Description

Version

v1.9.8

Date

July 3, 2016

Author

Tim Seemann

Copyright

MIT License

This file defines the protocols used for the Arduino libraries and the GUI.

A slightly modified version of this file exists in the Qt GUI project. None of the modifications change the naming, documentation, or order of the protocols. Instead, the changes allow the GUI version to use the strongly typed enums that were made available in C++11.

6.2.2 Enumeration Type Documentation

6.2.2.1 enum `EColorGroup`

used during multi color routines to determine which colors to use in the routine. `eCustom` uses the custom color array, `eAll` generates its colors randomly. All other values use presets based around overall themes.

Enumerator

`eCustom` 0

Use the custom color array instead of a preset group.

`eWater` 1

Shades of blue with some teal.

`eFrozen` 2

Shades of teal with some blue, white, and light purple.

`eSnow` 3

Shades of white with some blue and teal.

`eCool` 4

Based on the cool colors: blue, green, and purple.

`eWarm` 5

Based on the warm colors: red, orange, and yellow.

`eFire` 6

Similar to the warm set, but with an emphasis on oranges to give it a fire-like glow.

`eEvil` 7

Mostly red, with some other, evil highlights.

`eCorrosive` 8

Greens and whites, similar to radioactive goo from a 90s kids cartoon.

ePoison 9

A purple-based theme. Similar to poison vials from a 90s kids cartoon.

eRose 10

Shades of pink, red, and white.

ePinkGreen 11

The colors of watermelon candy. bright pinks and bright green.

eRedWhiteBlue 12

Bruce Springsteen's favorite color scheme, good ol' red, white, and blue.

eRGB 13

red, green, and blue.

eCMY 14

Cyan, magenta, yellow.

eSixColor 15

Red, yellow, green, cyan, blue, magenta.

eSevenColor 16

Red, yellow, green, cyan, blue, magenta, white.

eAll 17

Rather than using using preset colors, it uses all possible colors.

6.2.2.2 enum ELightingRoutine

Each routine makes the LEDs shine in different ways. There are two main types of routines: Single Color Routines use a single color while Multi Color Routines rely on an EColorGroup.

Enumerator**eOff 0**

Turns off the LEDs.

eSingleSolid 1

Shows a single color at a fixed brightness.

eSingleBlink 2

Alternates between showing a single color at a fixed brightness and turning the LEDs completely off.

eSingleWave 3

Linear fade of the brightness of the LEDs.

eSingleGlimmer 4

Randomly dims some of the LEDs to give a glimmer effect.

eSingleLinearFade 5

Linear fade of the brightness of the LEDs.

eSingleSineFade 6

Uses a sine function to fade in and out. This makes it spend more time near the extremes of full brightness and very dim light, and less time in the mid range. of the LEDs.

eSingleSawtoothFadeIn 7

fades in starting at 0 brightness and increases at a constant rate. Once it reaches full brightness, it resets back to zero and repeats.

eSingleSawtoothFadeOut 8

fades out starting at 0 brightness and decreases at a constant rate. Once it reaches 0, it resets back to full brightness and repeats.

eMultiGlimmer 9

Uses the first color of the array as the base color and uses the other colors for a glimmer effect.

eMultiFade 10

Fades slowly between each color in the array.

eMultiRandomSolid 11

Chooses a random color from the array and lights all LEDs to match that color.

eMultiRandomIndividual 12

Chooses a random color from the array for each individual LED.

eMultiBarsSolid 13

Draws the colors of the array in alternating groups of equal size.

eMultiBarsMoving 14

Draws the colors of the array in alternating groups of equal size. On each update, it moves those groups one index to the right, creating a scrolling effect.

6.2.2.3 enum EPacketHeader

Message headers for packets coming over serial.

Enumerator

eModeChange 0

Takes one int parameter that gets cast to ELightingMode.

eMainColorChange 1

Takes 3 parameters, a 0-255 representation of Red, Green, and Blue.

eCustomArrayColorChange 2

Takes four parameters, three parameters, the LED, a 0-255 representation of Red, Green, and Blue.

eBrightnessChange 3

Takes one parameter, sets the brightness between 0 and 100.

eSpeedChange 4

Takes one parameter, sets the delay value 1 - 23767.

eCustomColorCountChange 5

Change the number of colors used in a custom array routine.

eIdleTimeoutChange 6

Set to 0 to turn off, set to any other number minutes until idle timeout happens.

eResetSettingsToDefaults 7

Resets all values inside of [RoutinesRGB](#) back to their default values. Useful for soft resetting the LED hardware.

Index

- blue
 - Getters and Setters, [8](#)
- brightness
 - Getters and Setters, [8](#)
- color
 - Getters and Setters, [8](#)
- eAll
 - LightingProtocols.h, [21](#)
- eBrightnessChange
 - LightingProtocols.h, [22](#)
- eCMY
 - LightingProtocols.h, [21](#)
- eCool
 - LightingProtocols.h, [20](#)
- eCorrosive
 - LightingProtocols.h, [20](#)
- eCustom
 - LightingProtocols.h, [20](#)
- eCustomArrayColorChange
 - LightingProtocols.h, [22](#)
- eCustomColorCountChange
 - LightingProtocols.h, [22](#)
- eEvil
 - LightingProtocols.h, [20](#)
- eFire
 - LightingProtocols.h, [20](#)
- eFrozen
 - LightingProtocols.h, [20](#)
- eIdleTimeoutChange
 - LightingProtocols.h, [22](#)
- eMainColorChange
 - LightingProtocols.h, [22](#)
- eModeChange
 - LightingProtocols.h, [22](#)
- eMultiBarsMoving
 - LightingProtocols.h, [22](#)
- eMultiBarsSolid
 - LightingProtocols.h, [22](#)
- eMultiFade
 - LightingProtocols.h, [21](#)
- eMultiGlimmer
 - LightingProtocols.h, [21](#)
- eMultiRandomIndividual
 - LightingProtocols.h, [22](#)
- eMultiRandomSolid
 - LightingProtocols.h, [21](#)
- eOff
 - LightingProtocols.h, [21](#)

- ePinkGreen
 - LightingProtocols.h, [21](#)
- ePoison
 - LightingProtocols.h, [20](#)
- eRGB
 - LightingProtocols.h, [21](#)
- eRedWhiteBlue
 - LightingProtocols.h, [21](#)
- eResetSettingsToDefaults
 - LightingProtocols.h, [22](#)
- eRose
 - LightingProtocols.h, [21](#)
- eSevenColor
 - LightingProtocols.h, [21](#)
- eSingleBlink
 - LightingProtocols.h, [21](#)
- eSingleGlimmer
 - LightingProtocols.h, [21](#)
- eSingleLinearFade
 - LightingProtocols.h, [21](#)
- eSingleSawtoothFadeIn
 - LightingProtocols.h, [21](#)
- eSingleSawtoothFadeOut
 - LightingProtocols.h, [21](#)
- eSingleSineFade
 - LightingProtocols.h, [21](#)
- eSingleSolid
 - LightingProtocols.h, [21](#)
- eSingleWave
 - LightingProtocols.h, [21](#)
- eSixColor
 - LightingProtocols.h, [21](#)
- eSnow
 - LightingProtocols.h, [20](#)
- eSpeedChange
 - LightingProtocols.h, [22](#)
- eWarm
 - LightingProtocols.h, [20](#)
- eWater
 - LightingProtocols.h, [20](#)
- Getters and Setters, [7](#)
 - blue, [8](#)
 - brightness, [8](#)
 - color, [8](#)
 - green, [8](#)
 - red, [8](#)
- green
 - Getters and Setters, [8](#)

LightingProtocols.h

- eAll, [21](#)
- eBrightnessChange, [22](#)
- eCMY, [21](#)
- eCool, [20](#)
- eCorrosive, [20](#)
- eCustom, [20](#)
- eCustomArrayColorChange, [22](#)
- eCustomColorCountChange, [22](#)
- eEvil, [20](#)
- eFire, [20](#)
- eFrozen, [20](#)
- eIdleTimeoutChange, [22](#)
- eMainColorChange, [22](#)
- eModeChange, [22](#)
- eMultiBarsMoving, [22](#)
- eMultiBarsSolid, [22](#)
- eMultiFade, [21](#)
- eMultiGlimmer, [21](#)
- eMultiRandomIndividual, [22](#)
- eMultiRandomSolid, [21](#)
- eOff, [21](#)
- ePinkGreen, [21](#)
- ePoison, [20](#)
- eRGB, [21](#)
- eRedWhiteBlue, [21](#)
- eResetSettingsToDefaults, [22](#)
- eRose, [21](#)
- eSevenColor, [21](#)
- eSingleBlink, [21](#)
- eSingleGlimmer, [21](#)
- eSingleLinearFade, [21](#)
- eSingleSawtoothFadeIn, [21](#)
- eSingleSawtoothFadeOut, [21](#)
- eSingleSineFade, [21](#)
- eSingleSolid, [21](#)
- eSingleWave, [21](#)
- eSixColor, [21](#)
- eSnow, [20](#)
- eSpeedChange, [22](#)
- eWarm, [20](#)
- eWater, [20](#)

Multi Colors Routines, [12](#)

red

- Getters and Setters, [8](#)

Single Color Routines, [10](#)

These methods can be called after a routine is chosen,

[14](#)