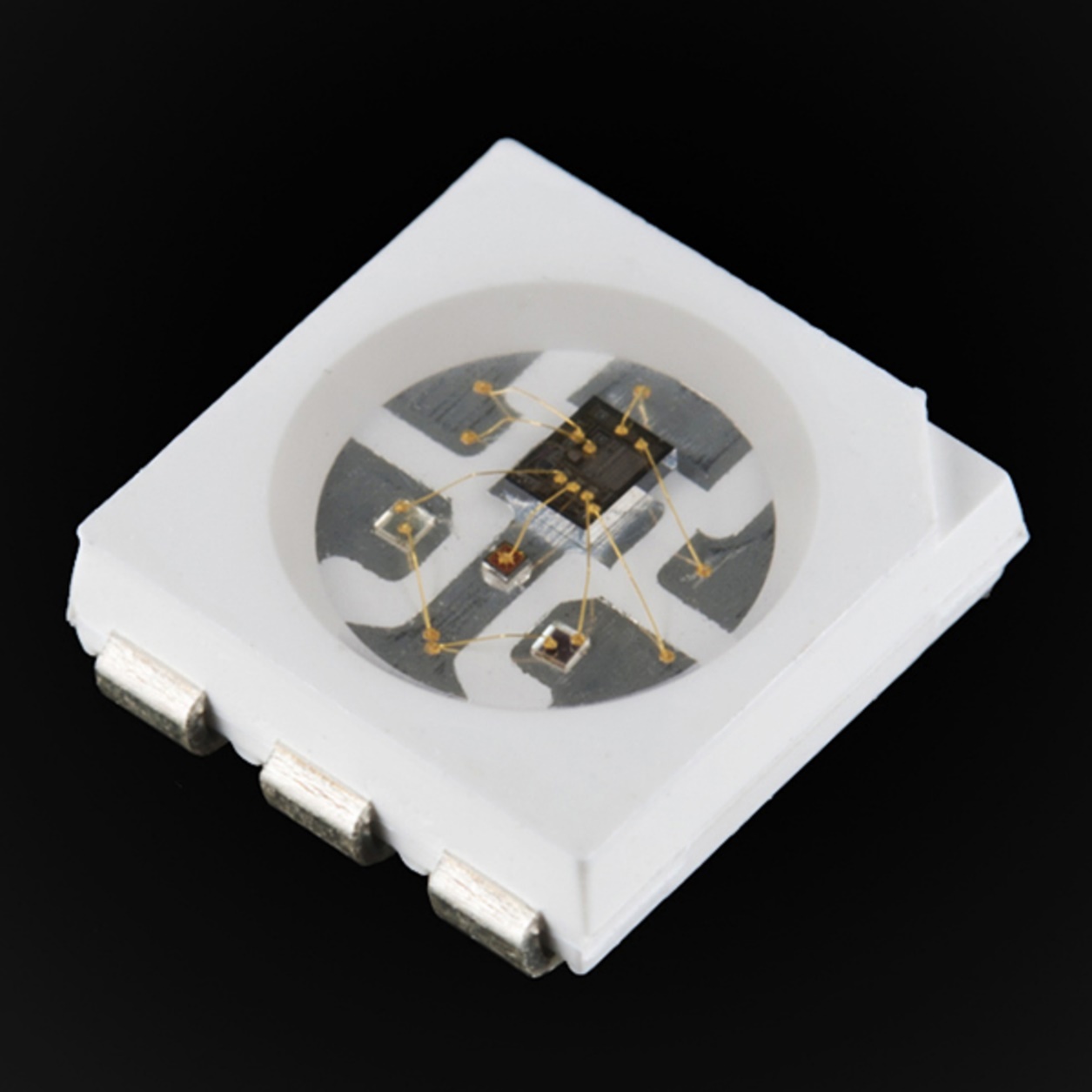
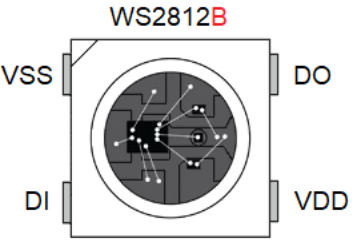
WS2812

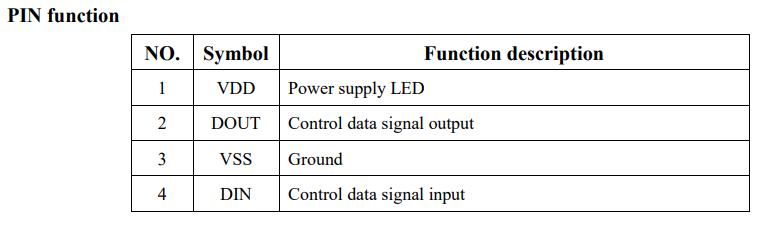
مقدمه

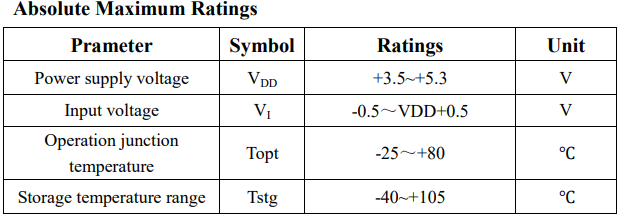
این LED را در اصل میتوان ترکیبی از سه LED به رنگ های قرمز، سبز و آبی دانست، در واقع یک LED RGB میباشد. به همراه یک واحد کنترل که وظیفه دریافت دیتا از ورودی و سپس نمایش رنگ مورد نظر با استفاده از ترکیب رنگ ها را دارد.



اطلاعات کلی

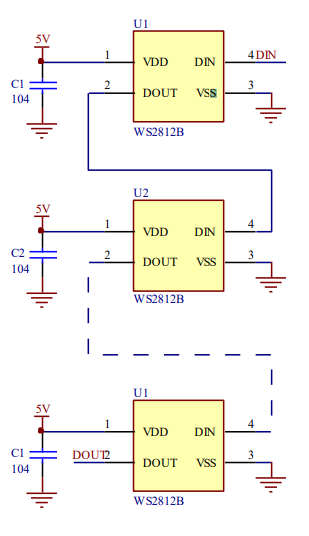






نحوه اتصال

LED ها را میتوان به صورت سری به شکل زیر به یکدیگر متصل کرد

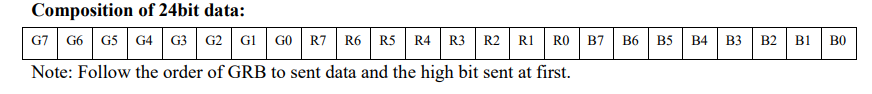


ارسال داده

برای مشخص کردن رنگ هر LED باید برای آن 3 بایت دیتا ارسال نماییم، که هر بایت میزان شدت یکی از رنگ ها (RGB, Red, Green, Blue) را مشخص مینماید.

فرمت رنگ

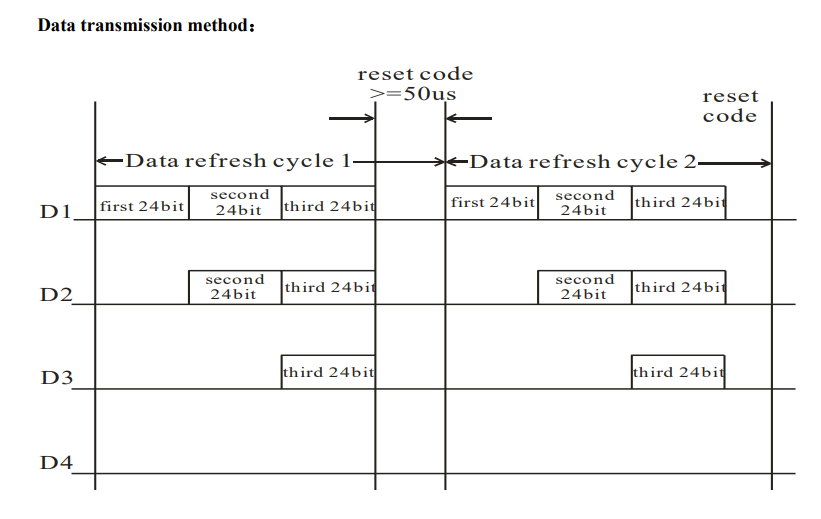
نحوه چیدمان دیتا برای مشخص کردن رنگ نیز به صورت زیر میباشد



ترتیب ارسال

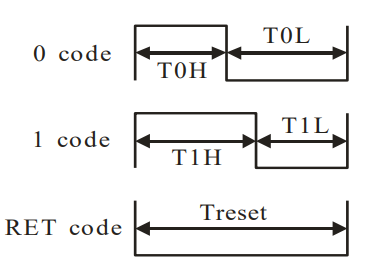
برای مثال، اگر بخواهیم رنگ 3 عدد LED را مشخص نماییم، باید 3 عدد رنگ کامل (24 بیت) را ارسال نماییم؛ با توجه به اینکه LED ها به صورت سری بسته شده اند، اولین رنگی که ارسال میشود برای اولین LED و آخرین رنگی که ارسال میشود برای آخرین LED میباشد.

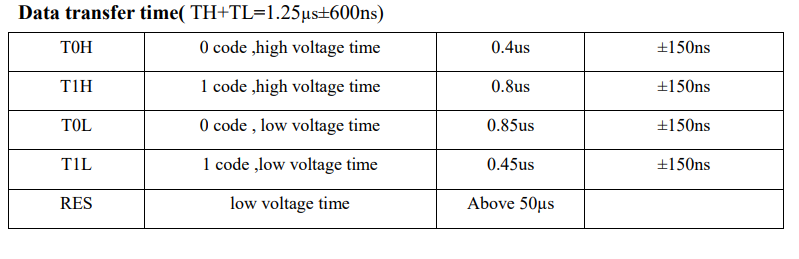
نیاز است، پس از ارسال رنگ ها، خط ارسالی را به مدت حداقل 50us، 0 نماییم تا LED ها رنگ های دریافت شده را نمایش دهند.



مودولاسیون 0 و 1

برای ارسال کردن بیت با معنای 1 و بیت به معنای 0 باید پالس ارسالی را به شکل زیر تنظیم نماییم





همان طور که واضح است، مودولاسیون به صورت PWM میباشد و تنها duty cycle در هر Period تغییر کرده است.

طراحی درایور

مقدمه

همان طور که گفته شد، برای ارسال اطلاعات به یک PWM Generator نیاز داریم، به این صورت واضح است که باید از یک تایمر استفاده نماییم برای مثال در این پروژه خاص ما از Timer 1 Channel 1 استفاده میکنیم.

محاسبات

برای ارسال یک بیت، طول داده مورد نظر ما 1.25us میباشد، و با توجه به اینکه فرکانس کاری ما 40MHz است، محاسباتperiod Counter تایمر را به صورت زیر داریم

برای تنظیم کردن آن، هم میتوانیم از نرم افزار CubeMX استفاده نماییم، یا پراپرتی زیر را مستقیما تغییر دهیم

htim1.Init.Period = 50 - 1;

برای ارسال 0 و 1 نیز محاسبات عرض پالس به شکل زیر است

نحوه ارسال دیتا بر روی PWM

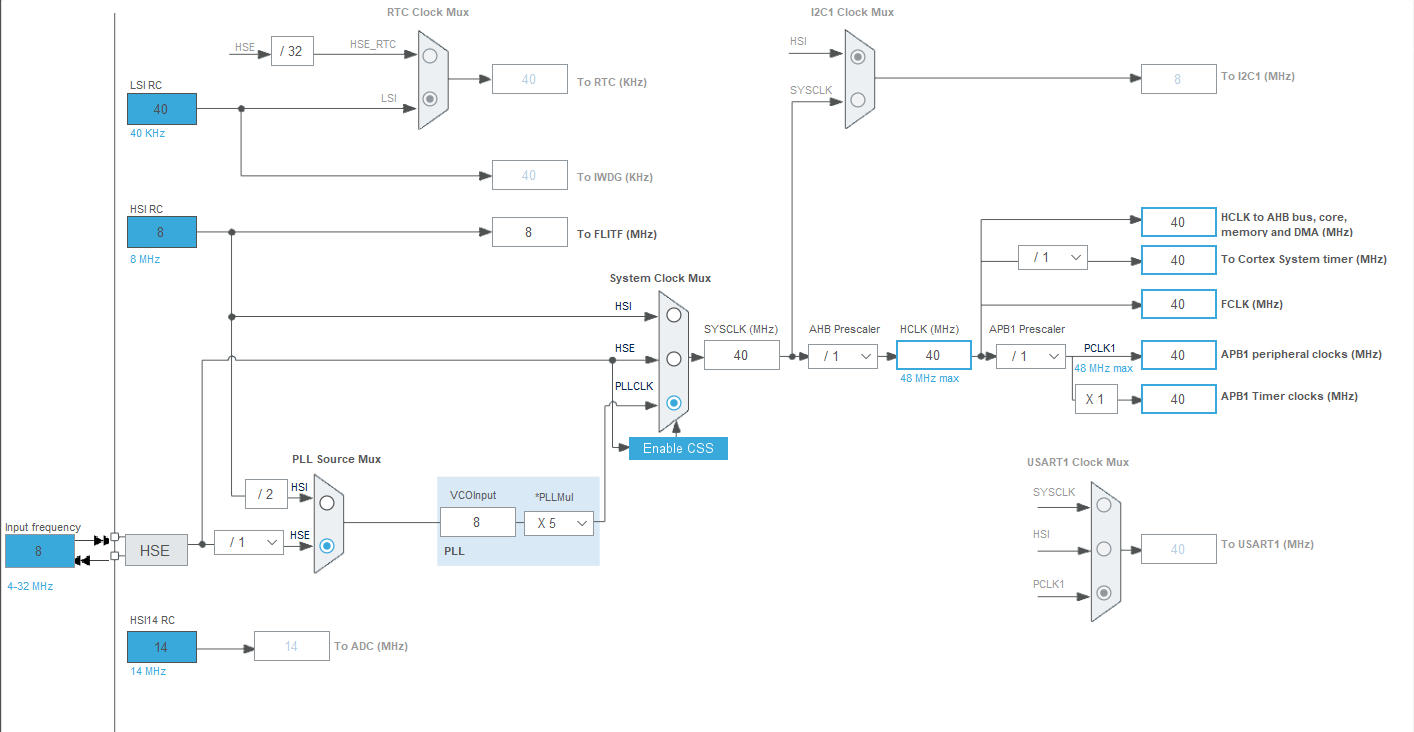
برای ارسال دیتا میتوانیم از واحد DMA استفاده نماییم، به این صورت که با ایجاد کردن یک buffer و ذخیره کردن دیتای مورد نظر در آن، واحد DMA را فراخوانی کرده، و دیتای مورد نظر را ارسال میکنیم

نحوه ایجاد تاخیر

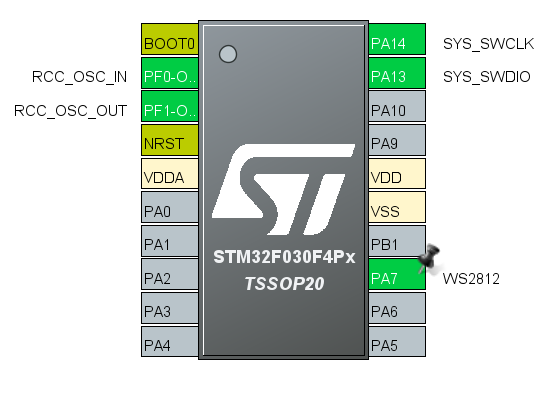
با توجه به اینکه حداقل تاخیر مورد نیاز، برای اینکه رنگ ها در LED ها ست شود، برابر با 50us است، میتوانیم با توجه به اینکه طول موج PWM ما برابر با 1.25us است، موج خروجی را به اندازه ای صفر نگه داریم، که حداقل 50us برای LED ما حاصل شود. برای مثال میتوانیم برای اینکار، 40 بایت داده صفر به خروجی ارسال کنیم، که برای اطمینان از صحت عملکرد، 50 بایت صفر ارسال خواهیم کرد.

تنظیمات در محیط CubeMX

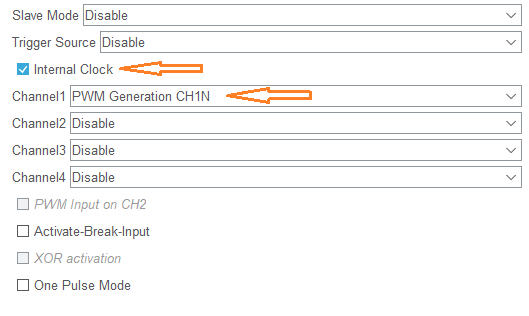
تنظیم کلاک کاری



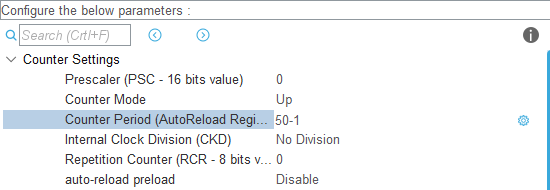
پایه خروجی



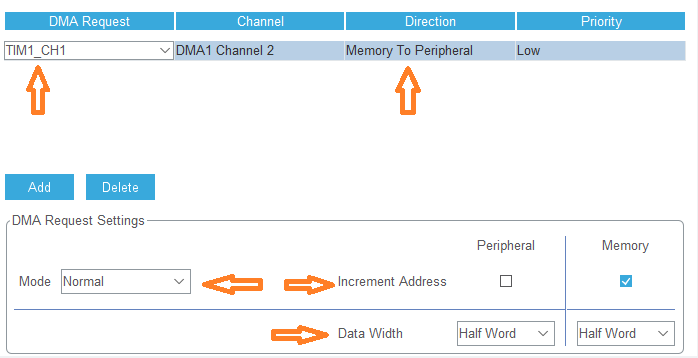
تنظیم Timer



تنظیم Counter Period



تنظیمات DMA



برنامه نویسی

مقدمه

برای مشاهده کامل درایور میتوانید به صفحه [GitHub](https://github.com/MasoudHeidary) من مراجعه نمایید.

Header

/\*

 \* WS2812.h

 \*

 \*  Created on: Apr 9, 2022

 \*      Author: Masoud Heidary

 \*      gitHub: https://github.com/MasoudHeidary/

 \*     License: MIT

 \*/

#ifndef INC\_WS2812\_H\_

#define INC\_WS2812\_H\_

/\* Includes ------------------------------------------------------------------\*/

#include <stdlib.h>

#include <stdbool.h>

#include "main.h"

/\* defines -------------------------------------------------------------------\*/

#define \_\_WS2812\_LEDsCount  3

در این قسمت تعداد LED ها توسط کاربر مشخص میشود.

#define \_\_WS2812\_DelayBufLen    50

میزان طول بافر که برای ایجاد کردن تاخیر مورد استفاده قرار میگیرد.

// (LEDs number + 2) \* (color size \_ 24bit \_ 3byte)

#define \_\_WS2812\_TimerBufLen    \_\_WS2812\_LEDsCount \* 24 + \_\_WS2812\_DelayBufLen

میزان طول بافر برای ذخیر کرده دیتا PWM میباشد، به این صورت که با ازای هر LED، 24 نیاز به ارسال بایت داده میباشد، که به عبارتی 24 عدد پالس مورد نیاز است

#define \_\_WS2812\_ZeroTime   16  // 0.4us \* 40MHz

#define \_\_WS2812\_OneTime    32  // 0.8us \* 40MHz

داده ای که برای مودلاسیون 0 و 1 مورد نیاز است تا به خروجی ارسال شود

/\* variables -----------------------------------------------------------------\*/

bool \_\_WS2812\_DMAIsReady;

زمانی که DMA ما شروع به ارسال داده مینماید، این متغیر را برابر false قرار خواهیم داد، و زمانی که ارسال داده توسط آن پایان یافت، تابع وقفه به صورت اتومات فراخوانی خواهد شد، در آن قسمت این متغیر را برابر true قرار خواهیم داد.

typedef struct {

    uint8\_t red;

    uint8\_t green;

    uint8\_t blue;

} WS2812\_colorStruct;

این ساختمان برای مشخص کردن رنگ میباشد.

// for setting colors for each LED

WS2812\_colorStruct \_\_WS2812\_ColorsBuf [\_\_WS2812\_LEDsCount];

این بافر برای ذخیره کردن رنگ های LED ها مورد استفاده قرار میگیرد.

// for timer to generate PWM

uint16\_t \_\_WS2812\_TimerBuf [\_\_WS2812\_TimerBufLen];

رنگ ها که در بافر \_\_WS2812\_ColorsBuf ذخیره شده است، برای اینکه قابل استفاده باشد باید به پالس های 0 و 1 تبدیل شود، این بافر برای ذخیره کردن این دیتا ها مورد استفاده قرار خواهد گرفت

\* توضیحات توابع در قسمت های خود داده خواهد شد

/\* public functions ----------------------------------------------------------\*/

// general

void WS2812\_init(void);

void WS2812\_DMACallBack(void);

// update led colors

void WS2812\_refresh(TIM\_HandleTypeDef, uint32\_t);

// set color

void WS2812\_setAll(WS2812\_colorStruct);

void WS2812\_setOne(WS2812\_colorStruct, int);

void WS2812\_clearAll(void);

void WS2812\_shiftNext(WS2812\_colorStruct);

void WS2812\_shiftPrevious(WS2812\_colorStruct color);

void WS2812\_rotateNext(void);

void WS2812\_rotatePrevious(void);

/\* private functions ----------------------------------------------------------\*/

void \_\_WS2812\_generateBuffer(void);

/\* ----------------------------------------------------------------------------\*/

#endif /\* INC\_WS2812\_H\_ \*/

Source

/\*

 \* WS2812.c

 \*

 \*  Created on: Apr 9, 2022

 \*      Author: Masoud Heidary

 \*      rief: <https://github.com/MasoudHeidary/>

 \*     License: MIT

 \*/

#include “WS2812.h”

/\*\*

 \* @breif initialize WS2812 Driver

 \*

 \* @param None

 \*

 \* @retval None

 \*/

void WS2812\_init(void) {

    \_\_WS2812\_DMAIsReady = true;

}

این تابع متغیر \_\_WS2812\_DMAIsReady را true میکند تا درایور متوجه شود که DMA قابل استفاده است.

/\*\*

 \* @breif use in DMA callback

 \*

 \* @param None

 \*

 \* @retval None

 \*/

void WS2812\_DMACallBack(void) {

    \_\_WS2812\_DMAIsReady = true;

}

این تابع متغیر \_\_WS2812\_DMAIsReady را true میکند، تا درایور متوجه شود که DMA قابل استفاده است، این تابع را در وقفه مربوط به تایمر قرار خواهیم داد.

/\*\*

 \* @breif generate PWM pulses from color buffer

 \*

 \* @param None

 \*

 \* @retval None

 \*/

void \_\_WS2812\_generateBuffer(void) {

    // buffer position

    long pos = 0;

    for(int i=0; i< \_\_WS2812\_LEDsCount; i++) {

        WS2812\_colorStruct color = \_\_WS2812\_ColorsBuf[i];

در این حلقه در رنگ های LED ها پیمایش میکنیم و موج PWM مورد نیاز را تولید میکنیم.

        // green bit:0-7

        \_\_WS2812\_TimerBuf[pos++] =

                ((color.green & 0x80) != 0) ? \_\_WS2812\_OneTime : \_\_WS2812\_ZeroTime;

        \_\_WS2812\_TimerBuf[pos++] =

                ((color.green & 0x40) != 0) ? \_\_WS2812\_OneTime : \_\_WS2812\_ZeroTime;

        \_\_WS2812\_TimerBuf[pos++] =

                ((color.green & 0x20) != 0) ? \_\_WS2812\_OneTime : \_\_WS2812\_ZeroTime;

        \_\_WS2812\_TimerBuf[pos++] =

                ((color.green & 0x10) != 0) ? \_\_WS2812\_OneTime : \_\_WS2812\_ZeroTime;

        \_\_WS2812\_TimerBuf[pos++] =

                ((color.green & 0x08) != 0) ? \_\_WS2812\_OneTime : \_\_WS2812\_ZeroTime;

        \_\_WS2812\_TimerBuf[pos++] =

                ((color.green & 0x04) != 0) ? \_\_WS2812\_OneTime : \_\_WS2812\_ZeroTime;

        \_\_WS2812\_TimerBuf[pos++] =

                ((color.green & 0x02) != 0) ? \_\_WS2812\_OneTime : \_\_WS2812\_ZeroTime;

        \_\_WS2812\_TimerBuf[pos++] =

                ((color.green & 0x01) != 0) ? \_\_WS2812\_OneTime : \_\_WS2812\_ZeroTime;

        // red bit:8-15

        \_\_WS2812\_TimerBuf[pos++] =

                ((color.red & 0x80) != 0) ? \_\_WS2812\_OneTime : \_\_WS2812\_ZeroTime;

        \_\_WS2812\_TimerBuf[pos++] =

                ((color.red & 0x40) != 0) ? \_\_WS2812\_OneTime : \_\_WS2812\_ZeroTime;

        \_\_WS2812\_TimerBuf[pos++] =

                ((color.red & 0x20) != 0) ? \_\_WS2812\_OneTime : \_\_WS2812\_ZeroTime;

        \_\_WS2812\_TimerBuf[pos++] =

                ((color.red & 0x10) != 0) ? \_\_WS2812\_OneTime : \_\_WS2812\_ZeroTime;

        \_\_WS2812\_TimerBuf[pos++] =

                ((color.red & 0x08) != 0) ? \_\_WS2812\_OneTime : \_\_WS2812\_ZeroTime;

        \_\_WS2812\_TimerBuf[pos++] =

                ((color.red & 0x04) != 0) ? \_\_WS2812\_OneTime : \_\_WS2812\_ZeroTime;

        \_\_WS2812\_TimerBuf[pos++] =

                ((color.red & 0x02) != 0) ? \_\_WS2812\_OneTime : \_\_WS2812\_ZeroTime;

        \_\_WS2812\_TimerBuf[pos++] =

                ((color.red & 0x01) != 0) ? \_\_WS2812\_OneTime : \_\_WS2812\_ZeroTime;

        // blue 16-23

        \_\_WS2812\_TimerBuf[pos++] =

                ((color.blue & 0x80) != 0) ? \_\_WS2812\_OneTime : \_\_WS2812\_ZeroTime;

        \_\_WS2812\_TimerBuf[pos++] =

                ((color.blue & 0x40) != 0) ? \_\_WS2812\_OneTime : \_\_WS2812\_ZeroTime;

        \_\_WS2812\_TimerBuf[pos++] =

                ((color.blue & 0x20) != 0) ? \_\_WS2812\_OneTime : \_\_WS2812\_ZeroTime;

        \_\_WS2812\_TimerBuf[pos++] =

                ((color.blue & 0x10) != 0) ? \_\_WS2812\_OneTime : \_\_WS2812\_ZeroTime;

        \_\_WS2812\_TimerBuf[pos++] =

                ((color.blue & 0x08) != 0) ? \_\_WS2812\_OneTime : \_\_WS2812\_ZeroTime;

        \_\_WS2812\_TimerBuf[pos++] =

                ((color.blue & 0x04) != 0) ? \_\_WS2812\_OneTime : \_\_WS2812\_ZeroTime;

        \_\_WS2812\_TimerBuf[pos++] =

                ((color.blue & 0x02) != 0) ? \_\_WS2812\_OneTime : \_\_WS2812\_ZeroTime;

        \_\_WS2812\_TimerBuf[pos++] =

                ((color.blue & 0x01) != 0) ? \_\_WS2812\_OneTime : \_\_WS2812\_ZeroTime;

    }

    // delay

    for(int i=0; i<\_\_WS2812\_DelayBufLen; i++)

        \_\_WS2812\_TimerBuf[pos++] = 0;

این قسمت برای ایجاد کردن وقفه مورد استفاده است، به این صورت که با دادن عرض پالس صفر، خروجی را صفر میکند.

}

/\*\*

 \* @breif generate PWM signal and set colors in LEDs

 \*

 \* @param timer TIM\_HandleTypeDef timer to generate PWM

 \* @param channel uint32\_t timer channel to generate PWM

 \*

 \* @retval None

 \*/

void WS2812\_refresh(TIM\_HandleTypeDef timer, uint32\_t channel) {

    \_\_WS2812\_generateBuffer();

    // wait until DMA being ready

    while (!\_\_WS2812\_DMAIsReady);

    \_\_WS2812\_DMAIsReady = false;

    HAL\_TIM\_PWM\_Start\_DMA(&timer, channel, (uint32\_t \*) \_\_WS2812\_TimerBuf, \_\_WS2812\_TimerBufLen);

}

این تابع ابتدا تا فراخوانی تابع \_\_WS2812\_generateBuffer عرض پالس های مورد نیاز خود را تولید میکند، سپس عرض پالس های تولید شده را با استفاده از DMA برای timer ارسال میکند تا موج خروجی میکرو کنترل تشکیل شود.

/\*\*

 \* @breif set color to all LEDs

 \*

 \* @param color WS2812\_colorStruct

 \*

 \* @retval None

 \*/

void WS2812\_setAll(WS2812\_colorStruct color) {

    for(int i=0; I < \_\_WS2812\_LEDsCount; i++) {

        \_\_WS2812\_ColorsBuf[i] = color;

    }

}

یک رنگ را به عنوان ورودی دریافت کرده، سپس تمام رنگ های LED ها را به آن تغییر میدهد.

/\*\*

 \* @breif turn off all the LEDs

 \*

 \* @param None

 \*

 \* @retval None

 \*/

void WS2812\_clearAll(void) {

    for(int i=0; I < \_\_WS2812\_LEDsCount; i++) {

        \_\_WS2812\_ColorsBuf[i] = (WS2812\_colorStruct){0, 0, 0};

    }

}

تمامی LED ها را خاموش میکند.

/\*\*

 \* @breif set color the led\_number'th LED

 \*

 \* @param color WS2812\_colorStruct to set

 \* @param led\_number - start from 0

 \*

 \* @retval None

 \*/

void WS2812\_setOne(WS2812\_colorStruct color, int led\_number) {

    if(led\_number < \_\_WS2812\_LEDsCount)

        \_\_WS2812\_ColorsBuf[led\_number] = color;

}

با دریافت یک رنگ و شماره LED، رنگ آن LED را به رنگ دریافتی تغییر میدهد.

/\*\*

 \* @breif shift colors

 \* D0, D1, D2 -> color, D0, D1

 \*

 \* @param color WS2812\_colorStruct insert in shift

 \*

 \* @retval None

 \*/

void WS2812\_shiftNext(WS2812\_colorStruct color) {

    for (int i = \_\_WS2812\_LEDsCount - 1; i > 0; i--)

        \_\_WS2812\_ColorsBuf[i] = \_\_WS2812\_ColorsBuf[i - 1];

    WS2812\_setOne(color, 0);

}

رنگ های LED ها به یکی به سمت بعدی شیفت داده، سپس رنگی که دریافت کرده را به عنوان رنگ اولین LED قرار میدهد.

/\*\*

 \* @breif shift colors

 \* D0, D1, D2 -> D1, D2, color

 \*

 \* @param color WS2812\_colorStruct insert in shift

 \*

 \* @retval None

 \*/

void WS2812\_shiftPrevious(WS2812\_colorStruct color) {

    for (int i = 0; i < \_\_WS2812\_LEDsCount - 1; i++)

        \_\_WS2812\_ColorsBuf[i] = \_\_WS2812\_ColorsBuf[i+1];

    WS2812\_setOne(color, \_\_WS2812\_LEDsCount - 1);

}

/\*\*

 \* @breif rotate colors

 \* 0, 1, 2 -> 2, 0, 1

 \*

 \* @param None

 \*

 \* @retval None

 \*/

void WS2812\_rotateNext(void) {

    WS2812\_colorStruct color = \_\_WS2812\_ColorsBuf[\_\_WS2812\_LEDsCount - 1];

    WS2812\_shiftNext(color);

}

/\*\*

 \* @breif rotate colors

 \* 0, 1, 2 -> 1, 2, 0

 \*

 \* @param None

 \*

 \* @retval None

 \*/

void WS2812\_rotatePrevious(void) {

    WS2812\_colorStruct color = \_\_WS2812\_ColorsBuf[0];

    WS2812\_shiftPrevious(color);

}

نحوه استفاده از درایور نوشته شده