8. RGB colorful light bar

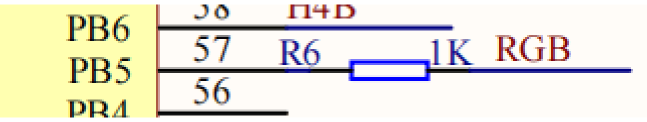
8.1. Purpose of Experiment

Use the SPI communication of STM32 to simulate the communication protocol of WS2812B module to drive the RGB light bar to show the effect.

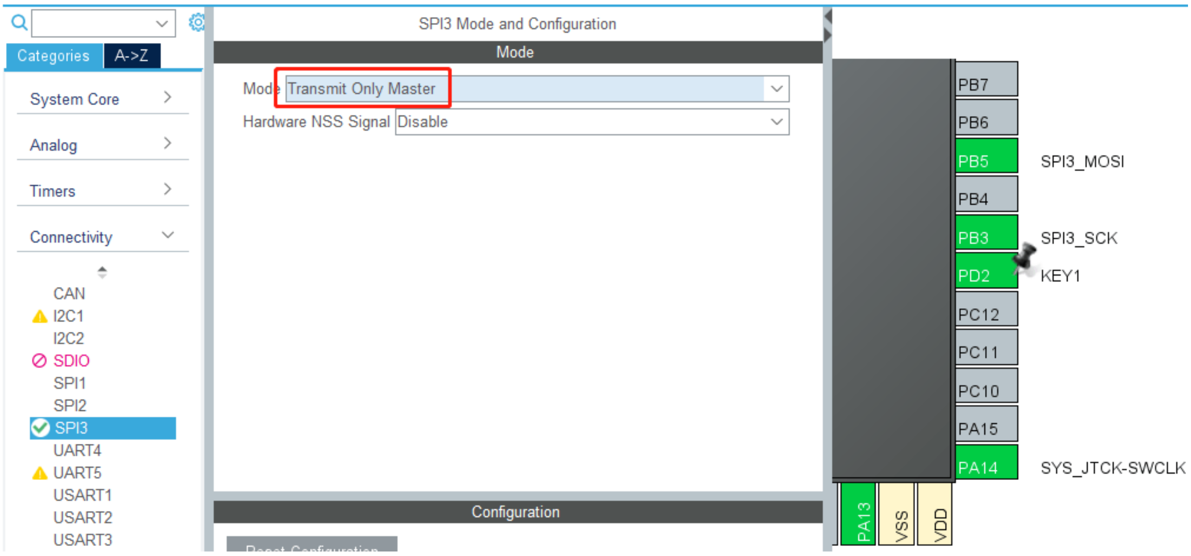
8.2 Configuring pin information

1. Import the ioc file from Beep's project and name it RGB_Strip.

According to the schematic diagram, we can see that the pin connected to the RGB strip is PB5. The RGB strip can be driven by timer PWM output or SPI output, and the PB5 pin supports redefining it as timer PWM or SPI output, considering the subsequent timer conflict problem, so we adopt the SPI communication method to drive the RGB strip here.



2. Set the mode of SPI3 to Transmit Only Master so that the PB5 pin is automatically set to SPI3_MOSI.

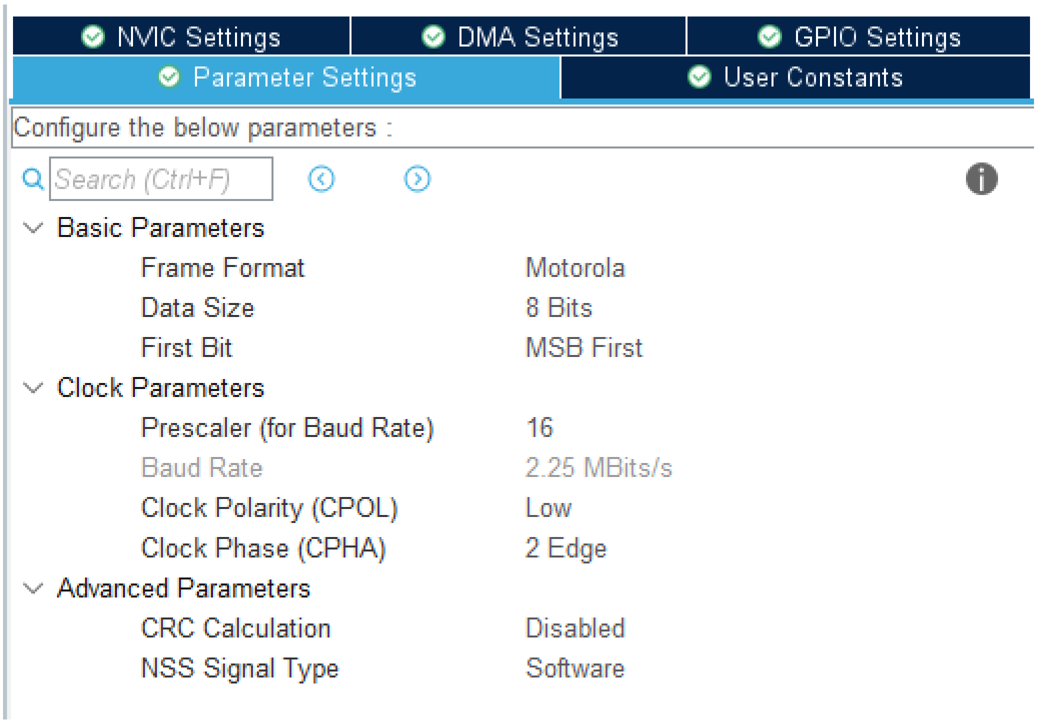

SPI3 Mode and Configuration

Mode

Mode Transmit Only Master

Hardware NSS Signal Disable

3. Modify the parameters of SPI3 sending data, refer to the following picture for specific parameters.



Configure the below parameters :

Basic Parameters

Frame Format	Motorola
Data Size	8 Bits
First Bit	MSB First

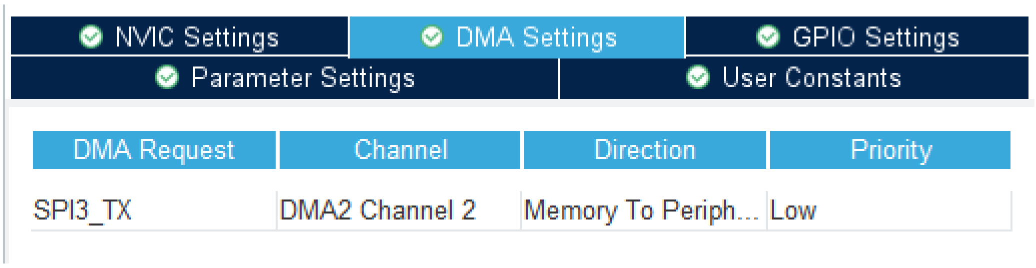
Clock Parameters

Prescaler (for Baud Rate)	16
Baud Rate	2.25 MBits/s
Clock Polarity (CPOL)	Low
Clock Phase (CPHA)	2 Edge

Advanced Parameters

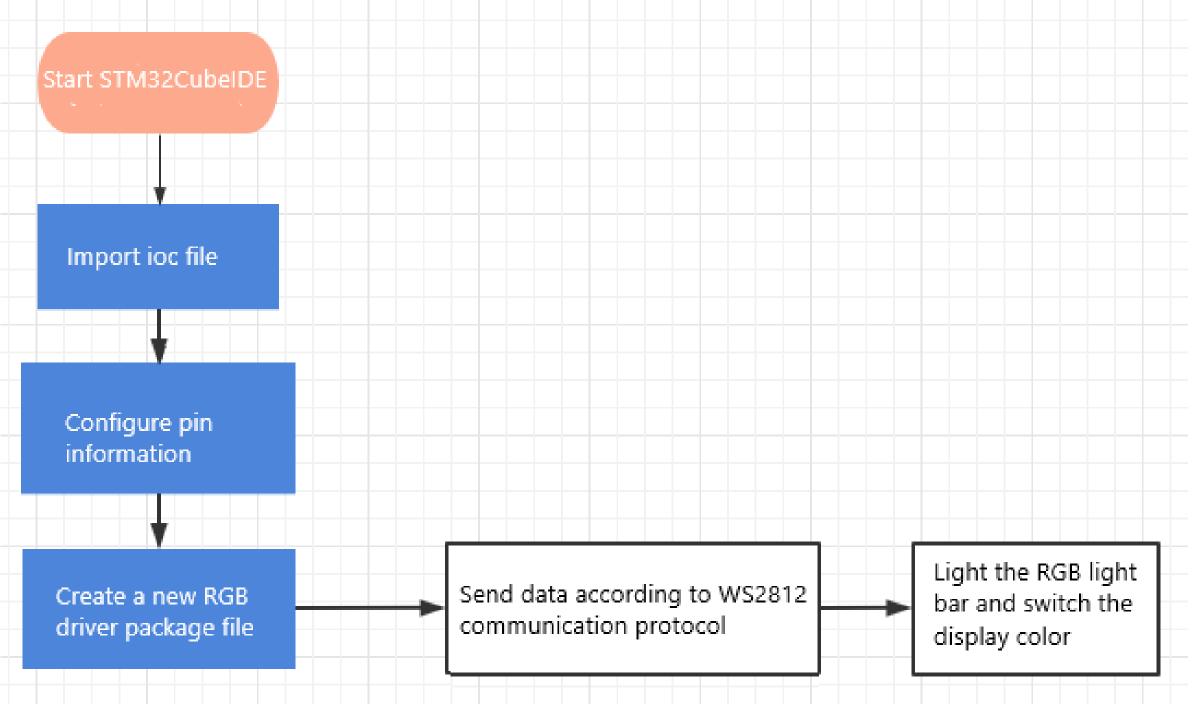
CRC Calculation	Disabled
NSS Signal Type	Software

4. Add DMA settings for SPI3_TX.



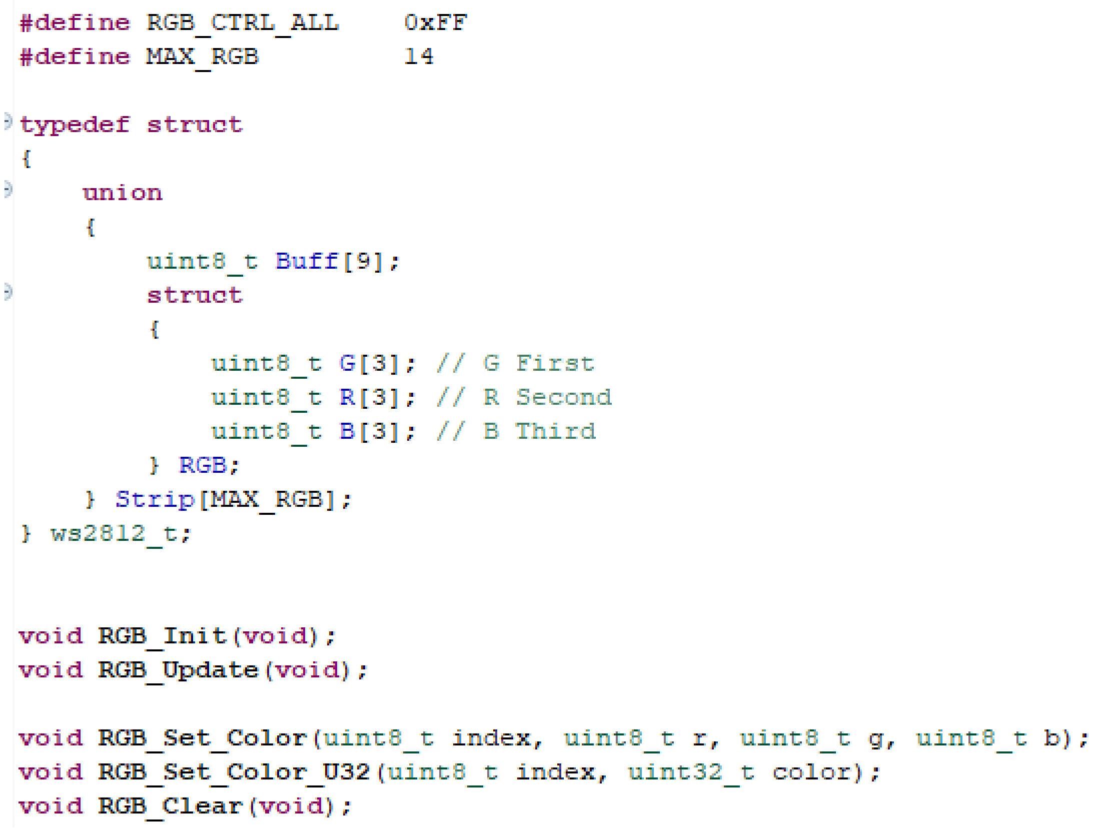
DMA Request	Channel	Direction	Priority
SPI3_TX	DMA2 Channel 2	Memory To Periph...	Low

8.3. Experimental flow chart analysis


Start STM32CubeIDE
Import ioc file
Configure pin information
Create a new RGB driver package file
Send data according to WS2812 communication protocol
Light the RGB light bar and switch the display color

8.4. Core Code Explanation

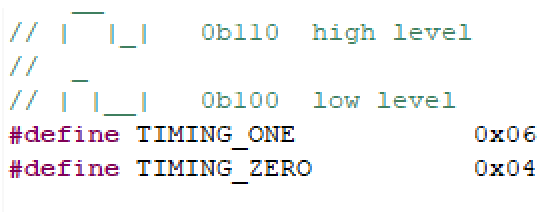
1. Create a new bsp_rgb.h and bsp_rgb.c, and add the following to bsp_rgb.h:



```
#define RGB_CTRL_ALL    0xFF
#define MAX_RGB         14

typedef struct
{
    union
    {
        uint8_t Buff[9];
        struct
        {
            uint8_t G[3]; // G First
            uint8_t R[3]; // R Second
            uint8_t B[3]; // B Third
        } RGB;
    } Strip[MAX_RGB];
} ws2812_t;


void RGB_Init(void);
void RGB_Update(void);

void RGB_Set_Color(uint8_t index, uint8_t r, uint8_t g, uint8_t b);
void RGB_Set_Color_U32(uint8_t index, uint32_t color);
void RGB_Clear(void);
```

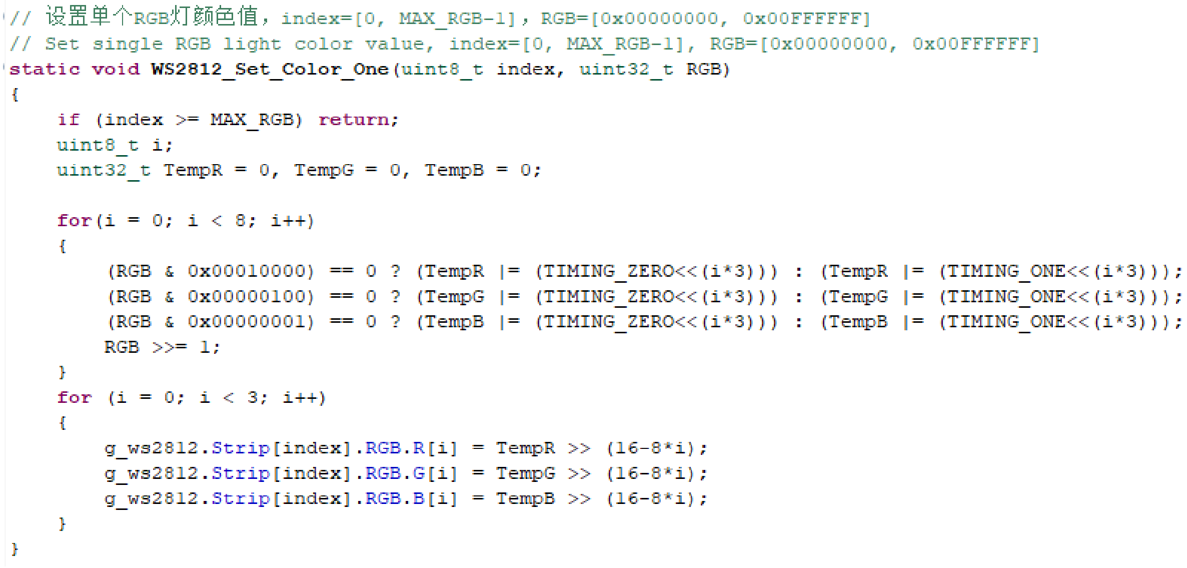
Among them, the ws2812_t structure is used to store the cache data of the light bar.

2. in bsp_rgb.c file in the following new content: according to the WS2812 communication protocol, the SPI three bits to simulate the WS2812 a. SPI output three bits of the value of 0b110, corresponding to the high level of the WS2812, SPI output three bits of the value of 0b100, corresponding to the low level of the WS2812.



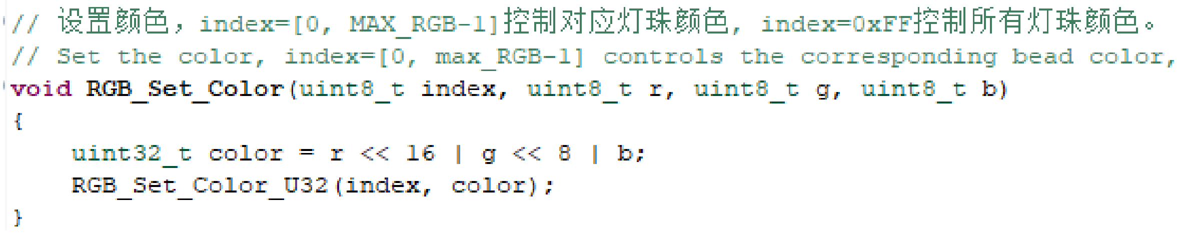
```
    __
// |  |_|   0b110  high level
//  _
// | |__|   0b100  low level
#define TIMING_ONE          0x06
#define TIMING_ZERO         0x04
```

3.WS2812_Set_Color_One() function sets a single RGB lamp color value, index=[0, MAX_RGB-1], RGB=[0x00000000, 0x00FFFFFF], the arrangement of the RGB value: red is at the top, green in the middle, and blue at the end.

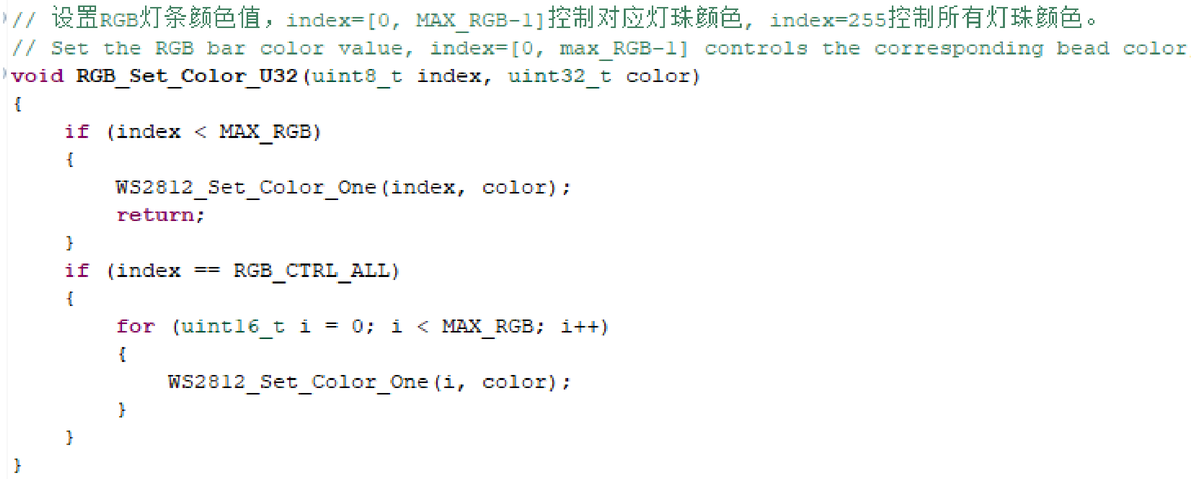


```
// 设置单个RGB灯颜色值，index=[0, MAX_RGB-1]，RGB=[0x00000000, 0x00FFFFFF]
// Set single RGB light color value, index=[0, MAX_RGB-1], RGB=[0x00000000, 0x00FFFFFF]
static void WS2812_Set_Color_One(uint8_t index, uint32_t RGB)
{
    if (index >= MAX_RGB) return;
    uint8_t i;
    uint32_t TempR = 0, TempG = 0, TempB = 0;

    for(i = 0; i < 8; i++)
    {
        (RGB & 0x00010000) == 0 ? (TempR |= (TIMING_ZERO<<(i*3))) : (TempR |= (TIMING_ONE<<(i*3)));
        (RGB & 0x00000100) == 0 ? (TempG |= (TIMING_ZERO<<(i*3))) : (TempG |= (TIMING_ONE<<(i*3)));
        (RGB & 0x00000001) == 0 ? (TempB |= (TIMING_ZERO<<(i*3))) : (TempB |= (TIMING_ONE<<(i*3)));
        RGB >>= 1;
    }
    for (i = 0; i < 3; i++)
    {
        g_ws2812.Strip[index].RGB.R[i] = TempR >> (16-8*i);
        g_ws2812.Strip[index].RGB.G[i] = TempG >> (16-8*i);
        g_ws2812.Strip[index].RGB.B[i] = TempB >> (16-8*i);
    }
}
```

4. RGB_Set_Color(index, r, g, b) and RGB_Set_Color_U32(index, color) set the RGB color value of the light bar, index=[0, MAX_RGB-1] control the corresponding lamp bead color, index=255 control the color of all lamp beads.

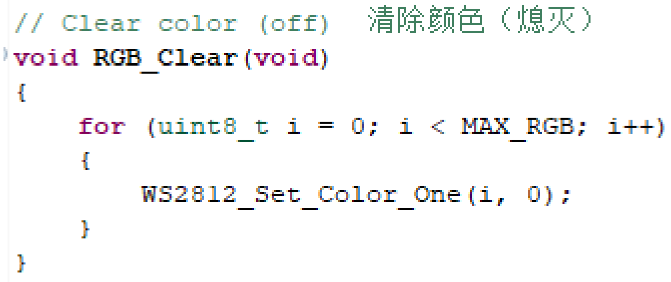


```
// 设置颜色，index=[0, MAX_RGB-1]控制对应灯珠颜色, index=0xFF控制所有灯珠颜色。
// Set the color, index=[0, max_RGB-1] controls the corresponding bead color,
void RGB_Set_Color(uint8_t index, uint8_t r, uint8_t g, uint8_t b)
{
    uint32_t color = r << 16 | g << 8 | b;
    RGB_Set_Color_U32(index, color);
}
```



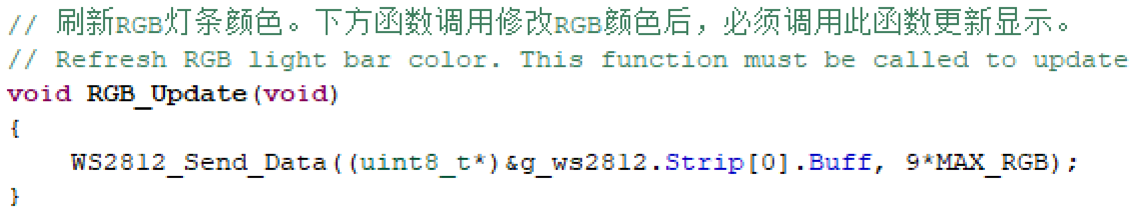
```
// 设置RGB灯条颜色值，index=[0, MAX_RGB-1]控制对应灯珠颜色, index=255控制所有灯珠颜色。
// Set the RGB bar color value, index=[0, max_RGB-1] controls the corresponding bead color
void RGB_Set_Color_U32(uint8_t index, uint32_t color)
{
    if (index < MAX_RGB)
    {
        WS2812_Set_Color_One(index, color);
        return;
    }
    if (index == RGB_CTRL_ALL)
    {
        for (uint16_t i = 0; i < MAX_RGB; i++)
        {
            WS2812_Set_Color_One(i, color);
        }
    }
}
```

5.RGB_Clear() clears the RGB lamp cache.



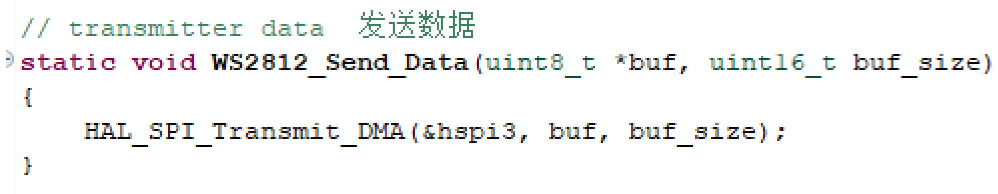
```
// Clear color (off)  清除颜色（熄灭）
void RGB_Clear(void)
{
    for (uint8_t i = 0; i < MAX_RGB; i++)
    {
        WS2812_Set_Color_One(i, 0);
    }
}
```

6. RGB_Update() function to refresh the RGB light bar color. RGB lights modify the color must be called after the RGB_Update() function to refresh the display, otherwise there will be no product effect.



```
// 刷新RGB灯条颜色。下方函数调用修改RGB颜色后，必须调用此函数更新显示。
// Refresh RGB light bar color. This function must be called to update
void RGB_Update(void)
{
    WS2812_Send_Data((uint8_t*)&g_ws2812.Strip[0].Buff, 9*MAX_RGB);
}
```

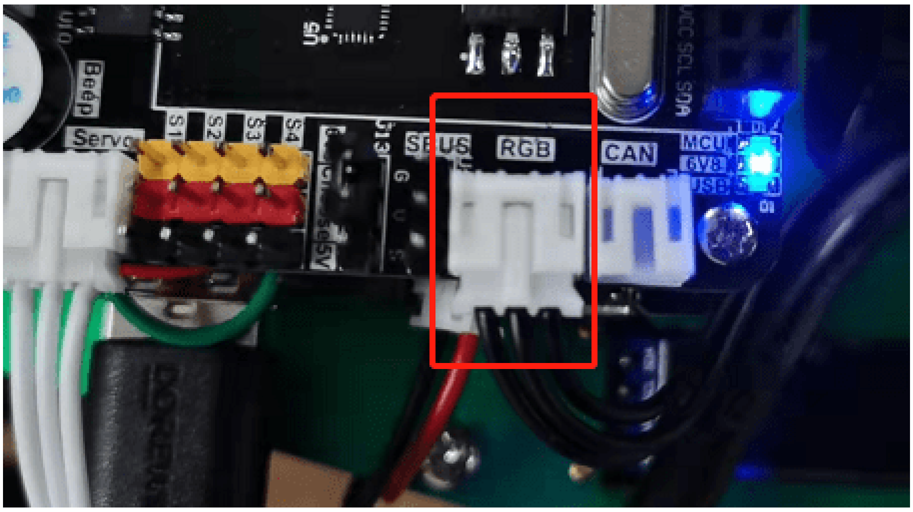
7. Send data using SPI3 DMA send.



```
// transmitter data  发送数据
static void WS2812_Send_Data(uint8_t *buf, uint16_t buf_size)
{
    HAL_SPI_Transmit_DMA(&hspi3, buf, buf_size);
}
```

8.5 Hardware Connection

As the RGB light bar needs to be connected to the location of the RGB light bar on the expansion board, the interface has been set up to prevent reverse connection, find the interface to see the direction of insertion can be.



8.6. Experimental Effect

After burning the program, the LED flashes every 200 milliseconds, and the RGB strip will display red, green, and blue in sequence, switching one color every 500 milliseconds.