8. RGB colorful light bar

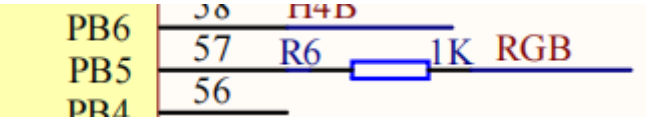
8.1. Purpose of the experiment

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

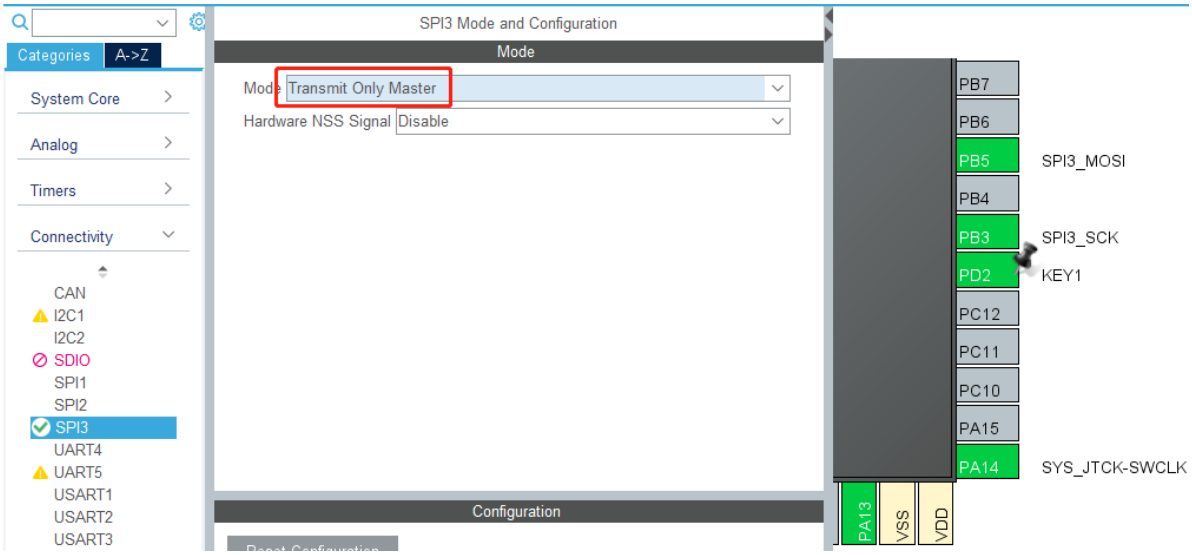
8.2. Configuration pin information

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

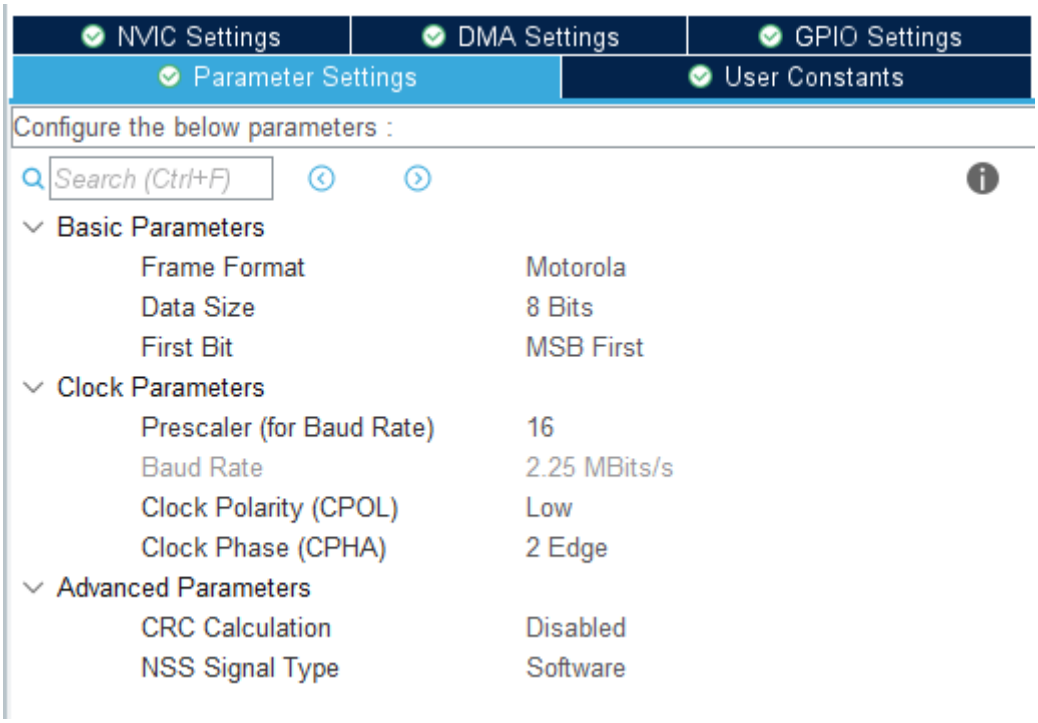
According to the schematic diagram, the pin connected to the RGB light bar is PB5. The driving method of the RGB light bar can use the timer PWM output or SPI output. The PB5 pin supports redefinition as the timer PWM or SPI output. Considering the subsequent timer conflict, the SPI communication method is used to drive the RGB light bar.



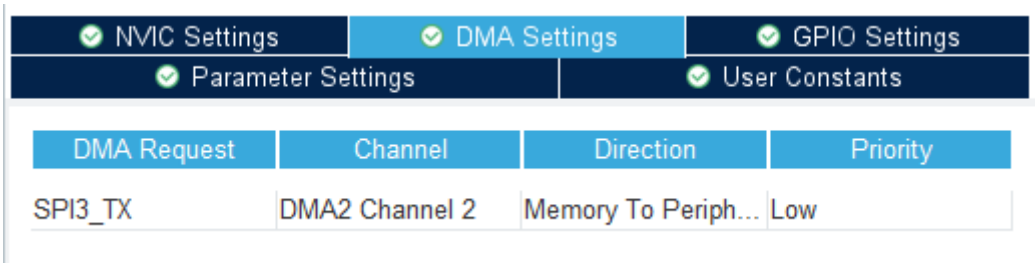
2. Set the mode of SPI3 to Transmit Only Master, so that the PB5 pin will be automatically set to SPI3_MOSI.



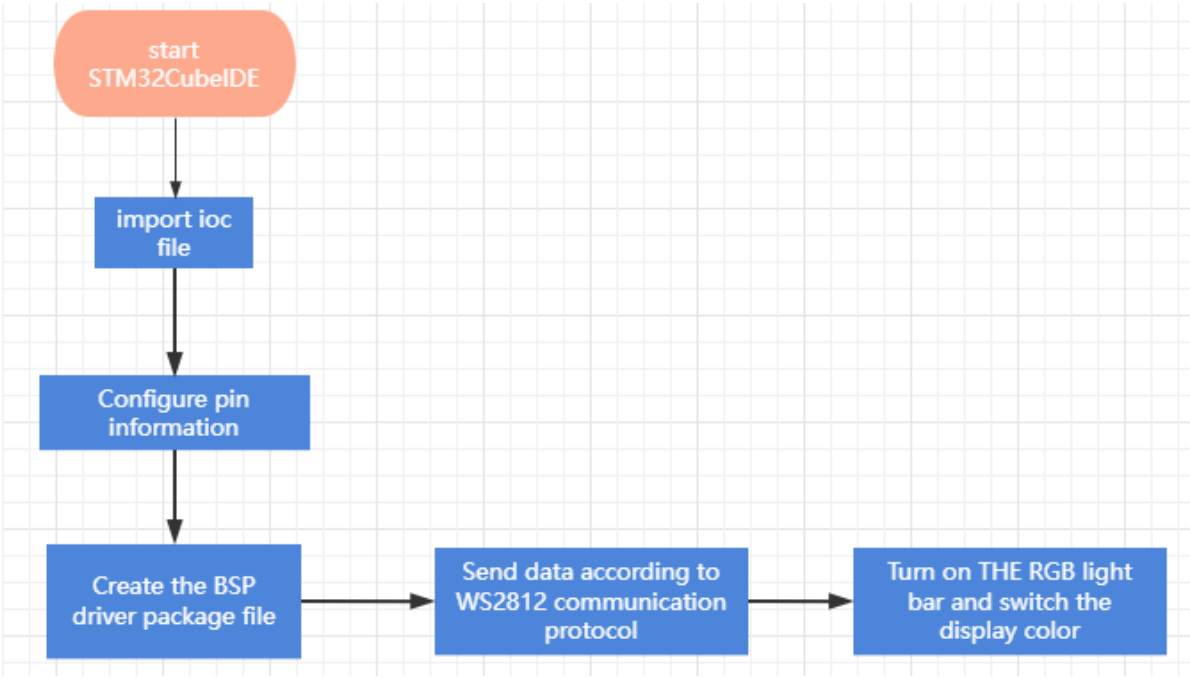
3. Modify the parameters of data sent by SPI3, refer to the following pictures for specific parameters.



4. Add DMA settings for SPI3_TX.



8.3. Analysis of the experimental flow chart



8.4. core code explanation

1. Create new bsp_rgb.h and bsp_rgb.c, and add the following content 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. Create the following content in the bsp_rgb.c file: According to the communication protocol of WS2812, use the three bits of SPI to simulate one bit of WS2812. When the value of the SPI output three bits is 0b110, it corresponds to the high level of WS2812, and when the SPI output three bits are 0b100, it corresponds to the low level of WS2812.

```
    __
// |  |_|   0bl10  high level
//  _
// | |__|   0bl00  low level
#define TIMING_ONE          0x06
#define TIMING_ZERO         0x04
```

3. The WS2812_Set_Color_One() function sets the color value of a single RGB lamp, index=[0, MAX_RGB-1], RGB=[0x00000000, 0x00FFFFFF], the arrangement of RGB values: red is at the front, 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 color value of the RGB light bar, index=[0, MAX_RGB-1] controls the color of the corresponding light beads, and index=255 controls the color of all light 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 light 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. The RGB_Update() function refreshes the color of the RGB light bar. After changing the color of the RGB lamp, you must call the RGB_Update() function to refresh the display, otherwise the product will not work.

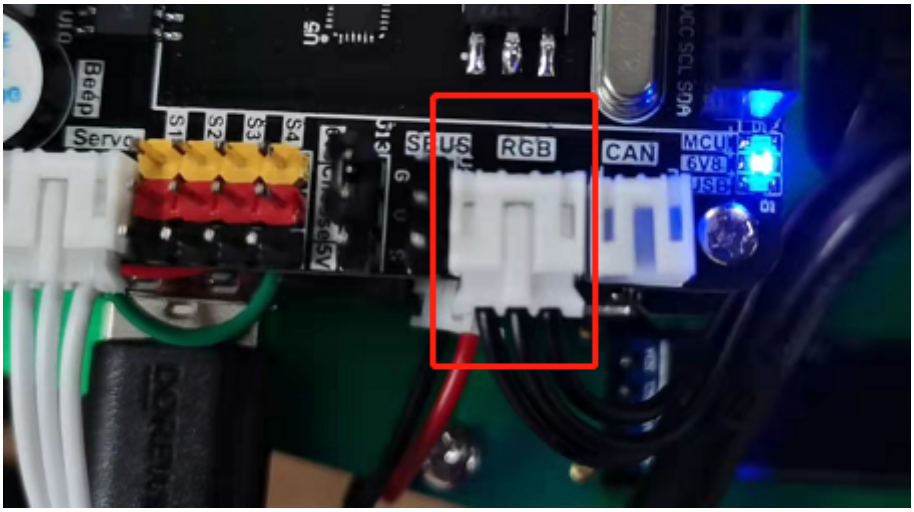
```
// 刷新RGB灯条颜色。下方函数调用修改RGB颜色后，必须调用此函数更新显示。
// Refresh RGB light bar color. This function must be called to update
void RGB_Update(void)
{
    WS2812_Send_Data((uint8_t*)&g_ws2812.Strip[0].Buff, 9*MAX_RGB);
}
```

7. Send data using SPI3 DMA.

```
// 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

Since the RGB light bar needs to be connected to the position of the RGB light bar on the expansion board, the interface has been set to prevent reverse connection, and you can find the interface and insert it in the correct direction.



8.6. Experimental effect

After the program is programmed, the LED light flashes once every 200 milliseconds, and the RGB light bar will display red, green, and blue in sequence, switching a color every 500 milliseconds.