9. Timer interrupt to control PWM servo

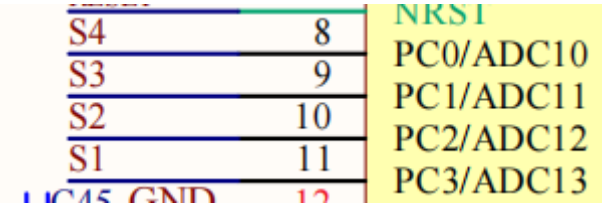
9.1. Purpose of the experiment

Use the basic timer interrupt function of STM32 to simulate the output PWM signal and control the PWM servo.

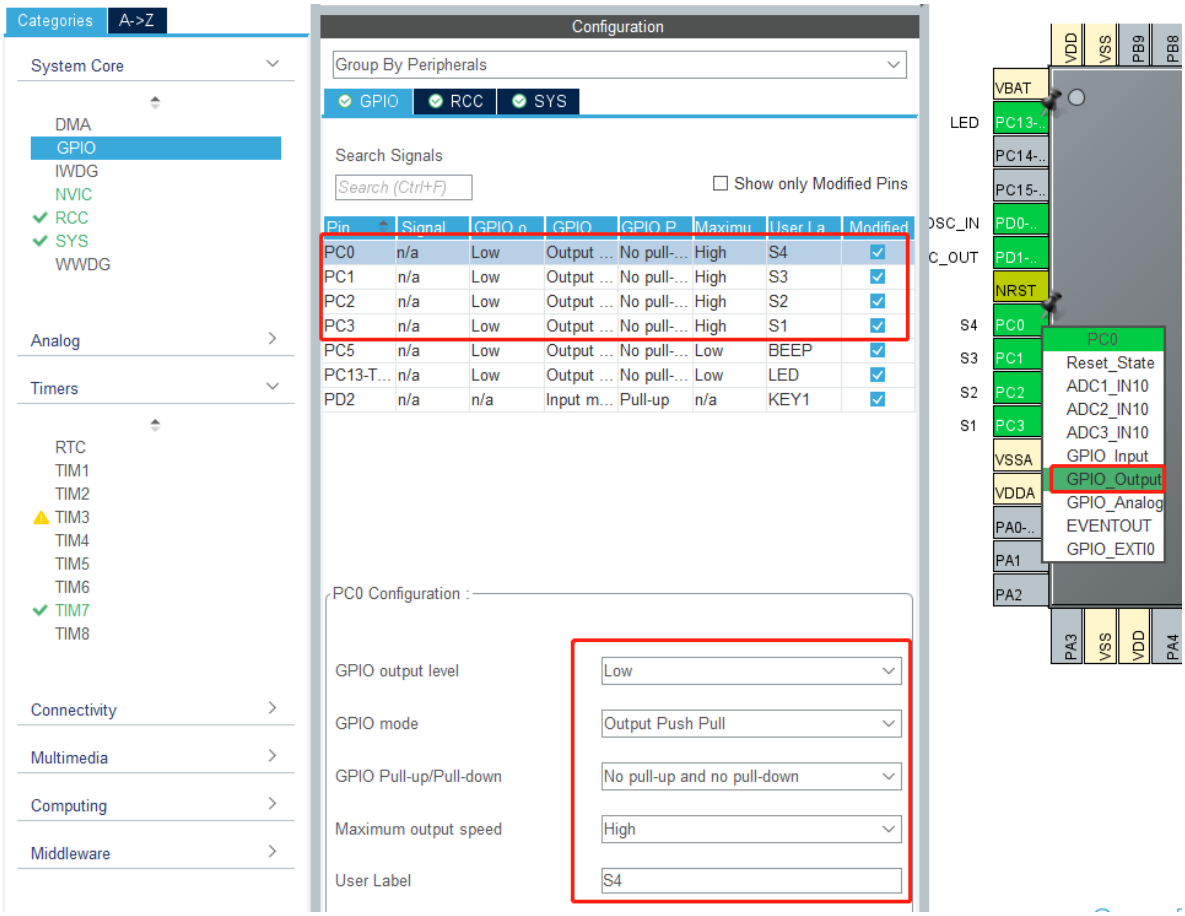
9.2, configuration pin information

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

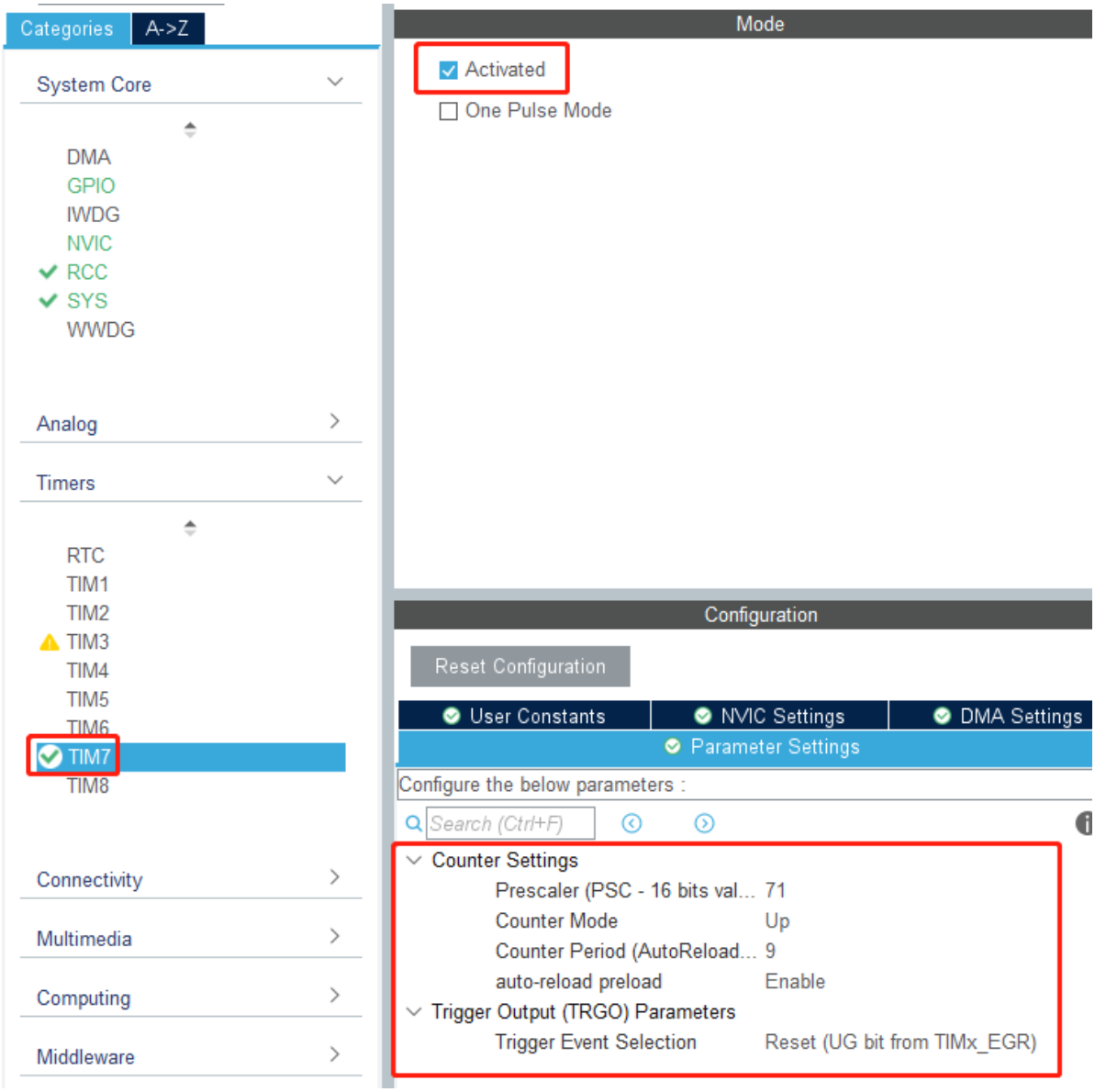
According to the schematic diagram, the servos S1 S2 S3 S4 are connected to the PC3 PC2 PC1 PC0 pins of the STM32 respectively.



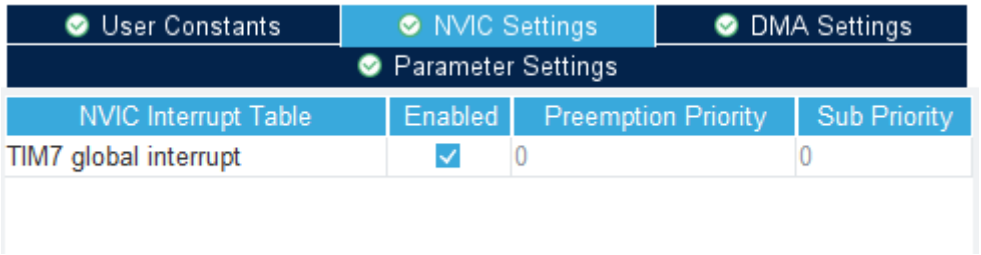
Set the PC0 PC1 PC2 PC3 pins as output mode, the specific parameters are shown in the following figure:



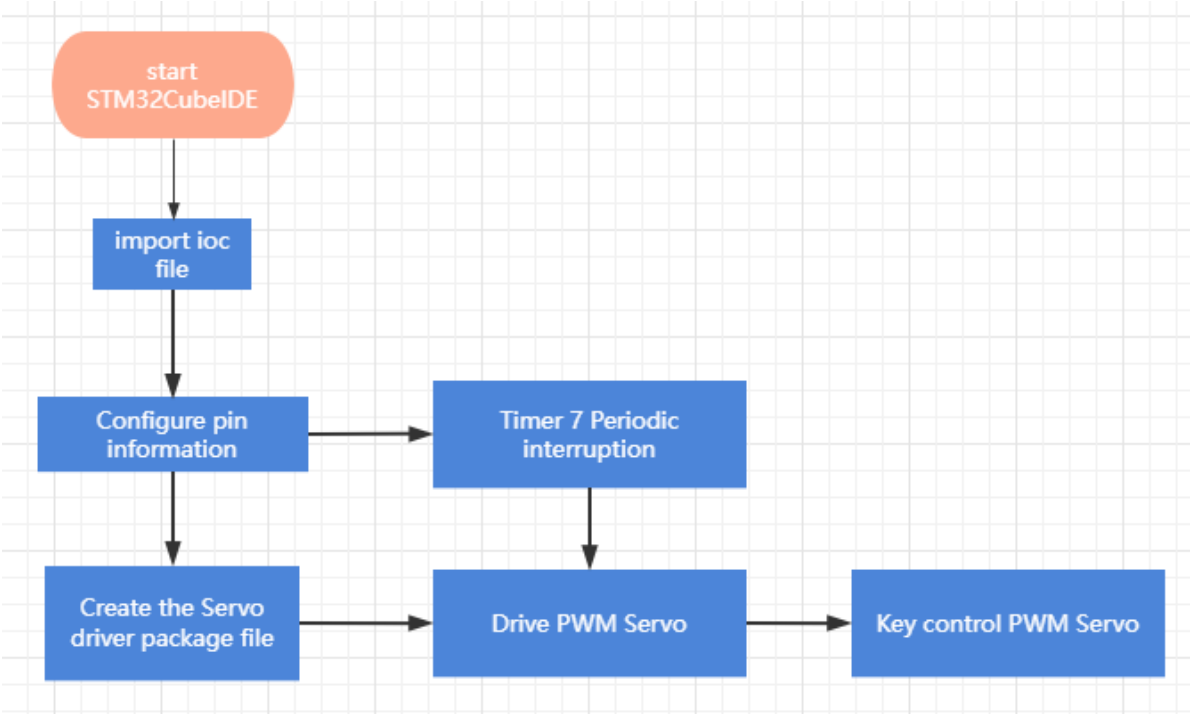
2. Next, we need to configure timer 7. The specific configuration parameters are shown in the figure below.



Turn on the timer global interrupt setting.



9.3. Analysis of the experimental flow chart



9.4. core code explanation

1. Create new bsp_pwmServo.h and bsp_pwmServo.c, and add the following content in pwmServo.h:

```
#define SERVO_1_HIGH()   HAL_GPIO_WritePin(S1_GPIO_Port, S1_Pin, GPIO_PIN_SET)
#define SERVO_1_LOW()    HAL_GPIO_WritePin(S1_GPIO_Port, S1_Pin, GPIO_PIN_RESET)

#define SERVO_2_HIGH()   HAL_GPIO_WritePin(S2_GPIO_Port, S2_Pin, GPIO_PIN_SET)
#define SERVO_2_LOW()    HAL_GPIO_WritePin(S2_GPIO_Port, S2_Pin, GPIO_PIN_RESET)

#define SERVO_3_HIGH()   HAL_GPIO_WritePin(S3_GPIO_Port, S3_Pin, GPIO_PIN_SET)
#define SERVO_3_LOW()    HAL_GPIO_WritePin(S3_GPIO_Port, S3_Pin, GPIO_PIN_RESET)

#define SERVO_4_HIGH()   HAL_GPIO_WritePin(S4_GPIO_Port, S4_Pin, GPIO_PIN_SET)
#define SERVO_4_LOW()    HAL_GPIO_WritePin(S4_GPIO_Port, S4_Pin, GPIO_PIN_RESET)


void PwmServo_Init(void);
void PwmServo_Set_Angle(uint8_t index, uint8_t angle);
void PwmServo_Set_Angle_All(uint8_t angle_s1, uint8_t angle_s2, uint8_t angle_s3, uint8_t angle_s4);
void PwmServo_Handle(void);
```

Among them, SERVO_X_HIGH() means output high level, SERVO_X_LOW() means output low level.

2. Create the following content in the bsp_pwmServo.c file:

The PwmServo_Init() function initializes the PWM position to 90 degrees.

```
// Initialize the steering gear  舵机初始化
void PwmServo_Init(void)
{
    for (int i = 0; i < MAX_PWM_SERVO; i++)
    {
        g_pwm_angle[i] = 90;
        g_angle_num[i] = PwmServo_Angle_To_Pulse(g_pwm_angle[i]);
    }
}
```

The PwmServo_Angle_To_Pulse() function converts the angle into a PWM duty cycle value.

```
// 角度转化为脉冲数, angle= [0, 180]
// The Angle is converted to the number of pulses, angle= [0, 180]
static uint16_t PwmServo_Angle_To_Pulse(uint8_t angle)
{
    uint16_t pulse = (angle * 11 + 500) / 10;
    return pulse;
}
```

3. The PwmServo_Set_Angle() function sets the pwm servo angle, index=0~3, and angle is 0-180.

```
// 设置pwm舵机角度，index=0~MAX_PWM_SERVO-1，angle为0-180
// Set the PWM servo Angle, index=0~MAX_PWM_SERVO, Angle to 0-180
void PwmServo_Set_Angle(uint8_t index, uint8_t angle)
{
    if (index >= MAX_PWM_SERVO)
        return;
    if (angle > 180)
        return;
    g_pwm_angle[index] = angle;
    g_angle_num[index] = PwmServo_Angle_To_Pulse(angle);
}
```

4. The PwmServo_Set_Angle_All() function sets the angle of all pwm servos, angle_s1 corresponds to the angle of S1, the range is 1-180, and the other three parameters correspond to the angle values of S2, S3 and S4 respectively.

```
// 设置全部pwm舵机的角度
// Set the Angle of all PWM steering gear
void PwmServo_Set_Angle_All(uint8_t angle_s1, uint8_t angle_s2, uint8_t angle_s3, uint8_t angle_s4)
{
    if (angle_s1 <= 180)
    {
        g_pwm_angle[0] = angle_s1;
        g_angle_num[0] = PwmServo_Angle_To_Pulse(angle_s1);
    }

    if (angle_s2 <= 180)
    {
        g_pwm_angle[1] = angle_s2;
        g_angle_num[1] = PwmServo_Angle_To_Pulse(angle_s2);
    }

    if (angle_s3 <= 180)
    {
        g_pwm_angle[2] = angle_s3;
        g_angle_num[2] = PwmServo_Angle_To_Pulse(angle_s3);
    }

    if (angle_s4 <= 180)
    {
        g_pwm_angle[3] = angle_s4;
        g_angle_num[3] = PwmServo_Angle_To_Pulse(angle_s4);
    }
}
```

5. The PwmServo_Handle() function needs to be called in the interrupt of the timer to simulate the output PWM signal and control the servo according to the angle value of the servo set above.

```
// PWM舵机控制，在定时器中调用，模拟输出PWM信号
// PWM steering gear control, in the timer call, analog output PWM signal
void PwmServo_Handle(void)
{
    g_pwm_pulse++;

#ifdef USE_SERVO_J1
    if (g_pwm_pulse <= g_angle_num[0])
        SERVO_1_HIGH();
    else
        SERVO_1_LOW();
#endif

#ifdef USE_SERVO_J2
    if (g_pwm_pulse <= g_angle_num[1])
        SERVO_2_HIGH();
    else
        SERVO_2_LOW();
#endif

#ifdef USE_SERVO_J3
    if (g_pwm_pulse <= g_angle_num[2])
        SERVO_3_HIGH();
    else
        SERVO_3_LOW();
#endif

#ifdef USE_SERVO_J4
    if (g_pwm_pulse <= g_angle_num[3])
        SERVO_4_HIGH();
    else
        SERVO_4_LOW();
#endif

    if (g_pwm_pulse >= 2000)
        g_pwm_pulse = 0;
}
```

6. Create a new HAL_TIM_PeriodElapsedCallback() function. The name of this function cannot be changed, otherwise the function will be found. The PwmServo_Handle() function is called by the timer 7 interrupt to generate the PWM signal.

```
// Timer interrupts the callback function  定时器中断回调函数
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
    if (htim->Instance == htim7.Instance)
    {
        PwmServo_Handle();
    }
}
```

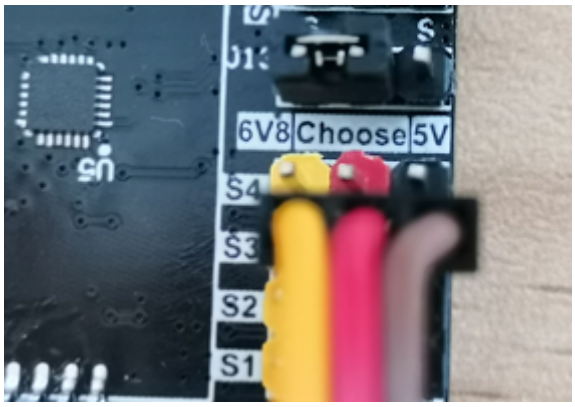
7. Add the following content to the Bsp_Loop() function and press the button to control the PWM servo.

```
// main.c中循环调用此函数，避免多次修改main.c文件。
// This function is called in a loop in main.c to avoid multiple modifications to the main.c file
void Bsp_Loop(void)
{
    static uint8_t key_state = 0;
    // Detect button down events   检测按键按下事件
    if (Key1_State(KEY_MODE_ONE_TIME))
    {
        Beep_On_Time(50);
        if (key_state)
        {
            key_state = 0;
            PwmServo_Set_Angle_All(50, 50, 50, 50);
        }
        else
        {
            key_state = 1;
            PwmServo_Set_Angle_All(150, 150, 150, 150);
        }
    }

    Bsp_Led_Show_State_Handle();
    // The buzzer automatically shuts down when times out   蜂鸣器超时自动关闭
    Beep_Timeout_Close_Handle();
    HAL_Delay(10);
}
```

9.5. Hardware connection

Since the PWM servos have different voltage drive values, the expansion board has added a voltage switching function. According to the jumper cap on the expansion board, the PWM output voltage can be modified to 5V or 6.8V. To use the PWM servo, the corresponding voltage must be selected with a jumper cap to avoid burning the servo. The PWM servos cannot be controlled without the jumper caps inserted. The pins of the PWM servo are: yellow->signal, red->power positive, black->power negative.



Since the power of the PWM servo is relatively large, the expansion board should not be powered by USB 5V directly, but must be powered by DC 12V.

9.6. Experimental effect

After the program is programmed, the LED light flashes every 200 milliseconds. Press the button multiple times, the PWM servo will go back and forth between 50 degrees and 150 degrees.