3.7PWM 呼吸灯实验

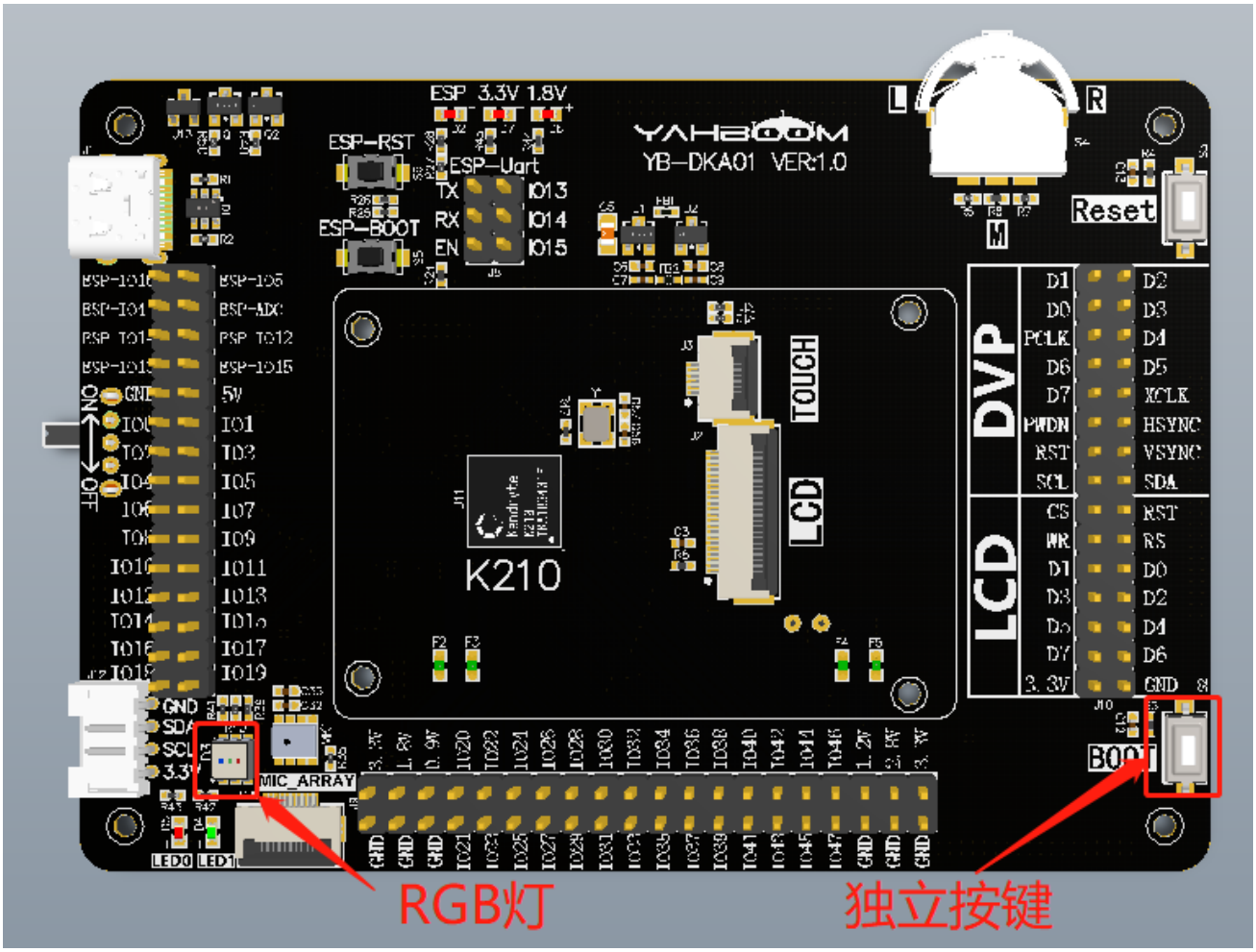
一、实验目的

本节课主要学习 K210 的 PWM 功能。

二、实验准备

1.实验元件

独立按键 BOOT、RGB 灯



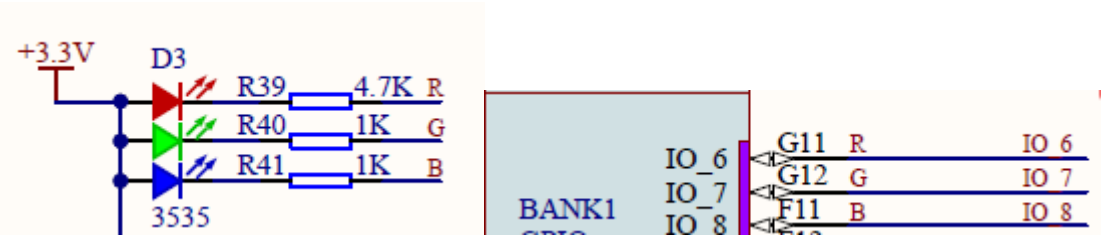
2.元件特性

K210 芯片定时器总共有 3 个，每个定时器有 4 路通道。每个定时器可以设置触发间隔，和定时器中断处理函数。定时器还可以设置为 PWM 输出的功能，但是

如果设置 PWM 输出，则不可以使用定时的功能。

3.硬件连接

K210 开发板出厂默认已经焊接好 RGB 灯。RGB 灯 R 连接的是 IO6，G 连接的是 IO7，B 连接的是 IO8。



4.SDK 中对应 API 功能

对应的头文件 pwm.h

脉冲宽度调制器 PWM 用于控制脉冲输出的占空比，其本质是一个定时器，所以注意设置 PWM 号与通道时，不要与 TIMER 定时器冲突。

为用户提供以下接口：

- pwm_init
- pwm_set_frequency
- pwm_set_enable

三、实验原理

PWM 控制的是脉冲输出的占空比，占空比是指在一个脉冲循环内，通电时间相对于总时间所占的比例。比如说，一个 RGB 灯在它一个工作周期中有一半时间被接通了，那么它的占空比就是 50%，同时亮度也只有 50%。如果加在该工作元件上的信号电压为 3V，则实际的工作电压平均值或电压有效值就是 1.5V。

四、实验过程

1.首先根据上面的硬件连接引脚图，K210 的硬件引脚和软件功能使用的是FPIOA 映射关系。这里映射的是定时器 1 的通道 0（开关 1）。

```
/*****************************HARDWARE-PIN**
// 硬件IO口，与原理图对应
#define PIN_RGB_R             (6)
#define PIN_RGB_G             (7)
#define PIN_RGB_B             (8)
```

```
void hardware_init(void)
{
    fpioa_set_function(PIN_RGB_R, FUNC_TIMER1_TOGGLE1);
}
```

2.第二步需要初始化外部中断服务，并且使能全局中断。如果没有这一步操作，系统的中断就不会运行，所以也不会调用中断回调函数。

```
/* 外部中断初始化 */
plic_init();
/* 使能全局中断 */
sysctl_enable_irq();
```

3.初始化定时器，这里使用的是定时器 0 通道 0，超时时间为 10 毫秒，定时器中断回调函数为 timer_timeout_cb，参数为空 NULL。

```
void init_timer(void) {
    /* 定时器初始化 */
    timer_init(TIMER_DEVICE_0);
    /* 设置定时器超时时间，单位为ns */
    timer_set_interval(TIMER_DEVICE_0, TIMER_CHANNEL_0, 10 * 1e6);
    /* 设置定时器中断回调 */
    timer_irq_register(TIMER_DEVICE_0, TIMER_CHANNEL_0, 0, 1, timer_timeout_cb, NULL);
    /* 使能定时器 */
    timer_set_enable(TIMER_DEVICE_0, TIMER_CHANNEL_0, 1);
}
```

4.初始化 PWM，设置 PWM 为定时器 1 通道 0，频率为 200KHz，占空比为 0.5 的方波。

```c
void init_pwm(void)
{
    /* 初始化PWM */
    pwm_init(PWM_DEVICE_1);
    /* 设置PWM频率为200KHZ，占空比为0.5的方波 */
    pwm_set_frequency(PWM_DEVICE_1, PWM_CHANNEL_0, 200000, 0.5);
    /* 使能 PWM 输出 */
    pwm_set_enable(PWM_DEVICE_1, PWM_CHANNEL_0, 1);
}
```

5.在定时器中断中设置 PWM 的占空比，根据 duty_cycle 的不同值，来修改 PWM 的占空比，从而改变 RGB 灯的亮度。

```c
int timer_timeout_cb(void *ctx) {
    static double duty_cycle = 0.01;
    /* 0为渐增，1为渐减 */
    static int flag = 0;

    /* 传入cycle的不同值，调节PWM的占用比，也就是调节灯的亮度 */
    pwm_set_frequency(PWM_DEVICE_1, PWM_CHANNEL_0, 200000, duty_cycle);

    /* 修改cycle的值，让其在区间（0,1）内渐增和渐减 */
    flag ? (duty_cycle -= 0.01): (duty_cycle += 0.01);
    if(duty_cycle > 1.0)
    {
        duty_cycle = 1.0;
        flag = 1;
    }
    else if (duty_cycle < 0.0)
    {
        duty_cycle = 0.0;
        flag = 0;
    }
    return 0;
}
```

6.最后是一个 while(1)循环，这个是必须的，否则系统就会退出，不再运行。

```c
int main(void)
{
    /* 硬件引脚初始化 */
    hardware_init();

    /* 系统中断初始化和使能 */
    plic_init();
    sysctl_enable_irq();

    /* 初始化定时器 */
    init_timer();

    /* 初始化PWM */
    init_pwm();

    while(1);

    return 0;
}
```

7.编译调试，烧录运行

把本课程资料中的 pwm 复制到 SDK 中的 src 目录下，然后进入 build 目录，运行以下命令编译。

cmake .. -DPROJ=pwm -G "MinGW Makefiles"

make

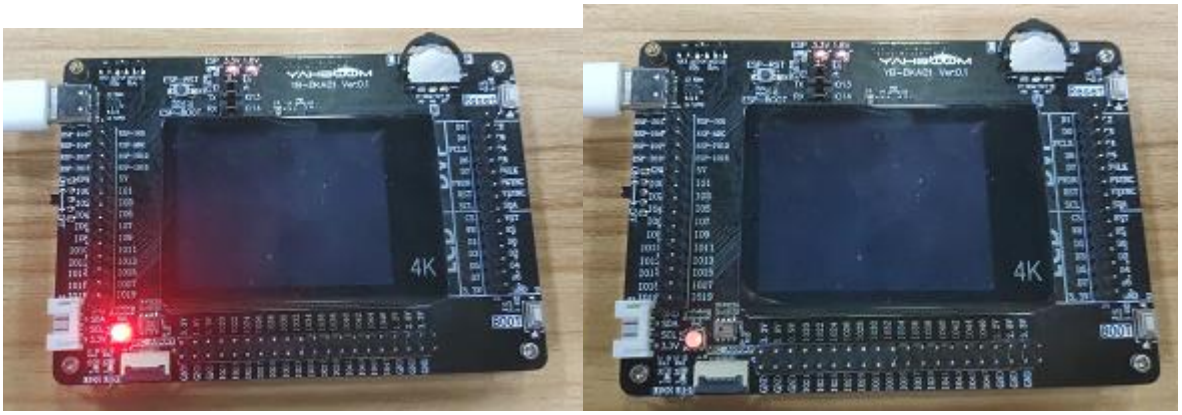
```
Scanning dependencies of target button
[ 97%] Building C object CMakeFiles/button.dir/src/button/main.c.obj
[100%] Linking C executable button
Generating .bin file ...
[100%] Built target button
PS C:\K210\SDK\kendryte-standalone-sdk-develop\build>
```

编译完成后，在 build 文件夹下会生成 pwm.bin 文件。

使用 type-C 数据线连接电脑与 K210 开发板，打开 kflash，选择对应的设备，再将程序固件烧录到 K210 开发板上。

五、实验现象

RGB 灯亮红色，而且显示呼吸灯效果，颜色从最亮逐渐变暗，再从熄灭状态中逐渐变亮，一直循环。



六、实验总结

1.PWM 的内部实现是基于定时器的定时功能。

2.控制 PWM 的两个重要因素是频率和占空比。

3.PWM 输出修改的是占空比，也就是通电时间占总时间的百分比，改变了输出的有效值，从而改变了 RGB 灯的亮度。

附：API

对应的头文件 pwm.h

pwm_init

描述

初始化 PWM。

函数原型

```
void pwm_init(pwm_device_number_t pwm_number)
```

参数

参数名称	描述	输入输出
pwm_number	pwm 号	输入

返回值

无。

pwm_set_frequency

描述

设置频率及占空比。

函数原型

```
double pwm_set_frequency(pwm_device_number_t pwm_number, pwm_channel_number_t channel, double frequency, double duty)
```

参数

参数名称	描述	输入输出
pwm_number	PWM 号	输入
channel	PWM 通道号	输入
frequency	PWM 输出频率	输入
duty	占空比	输入

返回值

实际输出频率。

pwm_set_enable

描述

使能禁用 PWM。

函数原型

```
void pwm_set_enable(pwm_device_number_t pwm_number, uint32_t channel, int
enable)
```

参数

参数名称	描述	输入输出
pwm_number	PWM 号	输入
channel	PWM 通道号	输入
enable	使能禁用 PWM 0：禁用 1：使能	输入

返回值

无。

举例

```
/* pwm0 channel 1 输出 200KHZ 占空比为 0.5 的方波 */
/* 设置 IO13 作为 PWM 的输出管脚 */
fpioa_set_function(13, FUNC_TIMER0_TOGGLE2);
pwm_init(PWM_DEVICE_0);
pwm_set_frequency(PWM_DEVICE_0, PWM_CHANNEL_1, 200000, 0.5);
pwm_set_enable(PWM_DEVICE_0, PWM_CHANNEL_1, 1);
```

数据类型

- pwm_device_number_t：pwm 号。
- pwm_channel_number_t：pwm 通道号。

pwm_device_number_t

描述

pwm 号。

定义

```
typedef enum _pwm_device_number
{
    PWM_DEVICE_0,
    PWM_DEVICE_1,
    PWM_DEVICE_2,
```

```
    PWM_DEVICE_MAX,
} pwm_device_number_t;
```

成员

成员名称	描述
PWM_DEVICE_0	PWM0
PWM_DEVICE_1	PWM1
PWM_DEVICE_2	PWM2

pwm_channel_number_t

描述

pwm 通道号。

定义

```
typedef enum _pwm_channel_number
{
    PWM_CHANNEL_0,
    PWM_CHANNEL_1,
    PWM_CHANNEL_2,
    PWM_CHANNEL_3,
    PWM_CHANNEL_MAX,
} pwm_channel_number_t;
```

成员

成员名称	描述
PWM_CHANNEL_0	PWM 通道 0
PWM_CHANNEL_1	PWM 通道 1
PWM_CHANNEL_2	PWM 通道 2
PWM_CHANNEL_3	PWM 通道 3