实验 1　　GPIO 实验

一、实验目的：

1、掌握 GPIO 的使用方法。

2、熟悉 IO 复用配置方法，熟悉中断系统应用。

3、熟悉 CCS 软件编程调试方法。

二、实验内容

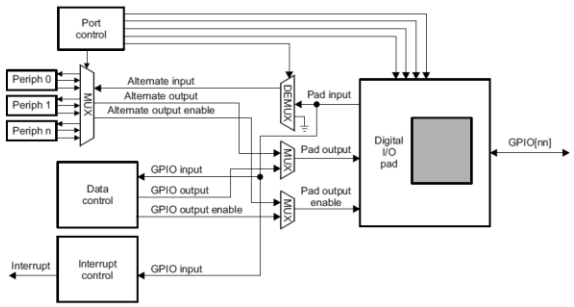
1、通过 GPIO 输出，控制 LED 状态

2、通过 GPIO 输入模式，读取按键状态，控制 LED 显示状态

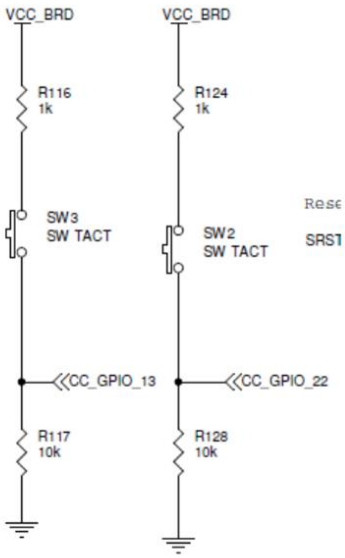
3、通过中断方式，实现按键控制 LED 亮灭

三、硬件方框图和电路图

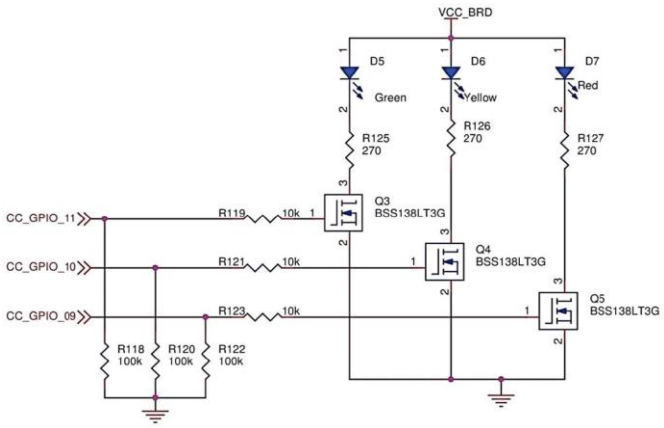
可控 LED 闪烁灯电路图



按键电路图



LED 电路图



四、实验原理

1、通过 GPIO 控制 LED 显示程序，参考 CC3200SDK 的 Blinky 工程。

在管脚复用设置中设定时钟，引脚映射成 GPIO 模式并设置为端口输出

```
PinMuxConfig(void)
{
    MAP_PRCMPeripheralClkEnable(PRCM_GPIOA1, PRCM_RUN_MODE_CLK);
    MAP_PinTypeGPIO(PIN_64, PIN_MODE_0, false);
    MAP_GPIODirModeSet(GPIOA1_BASE, 0x2, GPIO_DIR_MODE_OUT);
    MAP_PinTypeGPIO(PIN_01, PIN_MODE_0, false);
    MAP_GPIODirModeSet(GPIOA1_BASE, 0x4, GPIO_DIR_MODE_OUT);
    MAP_PinTypeGPIO(PIN_02, PIN_MODE_0, false);
    MAP_GPIODirModeSet(GPIOA1_BASE, 0x8, GPIO_DIR_MODE_OUT);
}
```

调用 gpio_if.c 中的 LED 配置

GPIO_IF_LedConfigure (*LED1*|*LED2*|*LED3*);

Main 中设置 LED 灯亮灭以及显示时间

```
void LEDBlinkyRoutine()
{
    GPIO_IF_LedOff(MCU_ALL_LED_IND);
    while(1)
    {
        MAP_UtilsDelay(8000000);
        GPIO_IF_LedOn(MCU_RED_LED_GPIO);
        MAP_UtilsDelay(8000000);
        GPIO_IF_LedOff(MCU_RED_LED_GPIO);
        MAP_UtilsDelay(8000000);
        GPIO_IF_LedOn(MCU_ORANGE_LED_GPIO);
        MAP_UtilsDelay(8000000);
        GPIO_IF_LedOff(MCU_ORANGE_LED_GPIO);
        MAP_UtilsDelay(8000000);
        GPIO_IF_LedOn(MCU_GREEN_LED_GPIO);
        MAP_UtilsDelay(8000000);
        GPIO_IF_LedOff(MCU_GREEN_LED_GPIO);
    }
}
```

2、按键中断程序参考 DY-IoT-PB_KEY 工程

在管脚复用设置中设定时钟，引脚映射成 GPIO 模式并设置为端口输出/输出

```
void PinMuxConfig(void)
{
    PRCMPeripheralClkEnable(PRCM_GPIOA0, PRCM_RUN_MODE_CLK);
    PRCMPeripheralClkEnable(PRCM_GPIOA2, PRCM_RUN_MODE_CLK);
    PRCMPeripheralClkEnable(PRCM_GPIOA3, PRCM_RUN_MODE_CLK);
    PinTypeGPIO(PIN_55, PIN_MODE_0, false);
    GPIODirModeSet(GPIOA0_BASE, 0x2, GPIO_DIR_MODE_IN);
    PinTypeGPIO(PIN_15, PIN_MODE_0, false);
    GPIODirModeSet(GPIOA2_BASE, 0x40, GPIO_DIR_MODE_IN);
    PinTypeGPIO(PIN_16, PIN_MODE_0, false);
    GPIODirModeSet(GPIOA2_BASE, 0x80, GPIO_DIR_MODE_IN);
    PinTypeGPIO(PIN_17, PIN_MODE_0, false);
    GPIODirModeSet(GPIOA3_BASE, 0x1, GPIO_DIR_MODE_IN);
    PinTypeGPIO(PIN_18, PIN_MODE_0, false);
    GPIODirModeSet(GPIOA3_BASE, 0x10, GPIO_DIR_MODE_OUT);
    GPIODirModeSet(GPIOA3_BASE,GPIO_PIN_2,GPIO_DIR_MODE_OUT);
}
```

在 main 中对按 GPIO 按键中断初始化

```
void Keyinit(void)
{
    GPIOIntTypeSet(GPIOA2_BASE,GPIO_PIN_7,GPIO_RISING_EDGE);
    IntPrioritySet(INT_GPIOA2, INT_PRIORITY_LVL_1);
    GPIOIntRegister(GPIOA2_BASE,Key3Handler);
    GPIOIntClear(GPIOA2_BASE,GPIO_INT_PIN_7);
    IntPendClear(INT_GPIOA2);
    IntEnable(INT_GPIOA2);
    GPIOIntEnable(GPIOA2_BASE,GPIO_INT_PIN_7);
    GPIOIntTypeSet(GPIOA3_BASE,GPIO_PIN_0,GPIO_RISING_EDGE);
    IntPrioritySet(INT_GPIOA3, INT_PRIORITY_LVL_1);
    GPIOIntRegister(GPIOA3_BASE,Key4Handler);
    GPIOIntClear(GPIOA3_BASE,GPIO_INT_PIN_0);
    IntPendClear(INT_GPIOA3);
    IntEnable(INT_GPIOA3);
    GPIOIntEnable(GPIOA3_BASE,GPIO_INT_PIN_0);
}
```

利用 if 判断循环查询两个按键情况，进入 if 后对 flag 归零并运行指定操作

```
while(1)
    {
       if(key3flag==1)
       {
           key3flag=0;
           GPIOPinWrite(GPIOA3_BASE,GPIO_PIN_4 | GPIO_PIN_3 | GPI
O_PIN_2,GPIO_PIN_4 | GPIO_PIN_3 | GPIO_PIN_2); //RGB LED ON }
       if(key4flag==1)
       {
           key4flag=0;
           GPIOPinWrite(GPIOA3_BASE,GPIO_PIN_4 | GPIO_PIN_3 | GPI
O_PIN_2,~(GPIO_PIN_4 | GPIO_PIN_3 | GPIO_PIN_2)); //RGB LED OFF }
    }
```

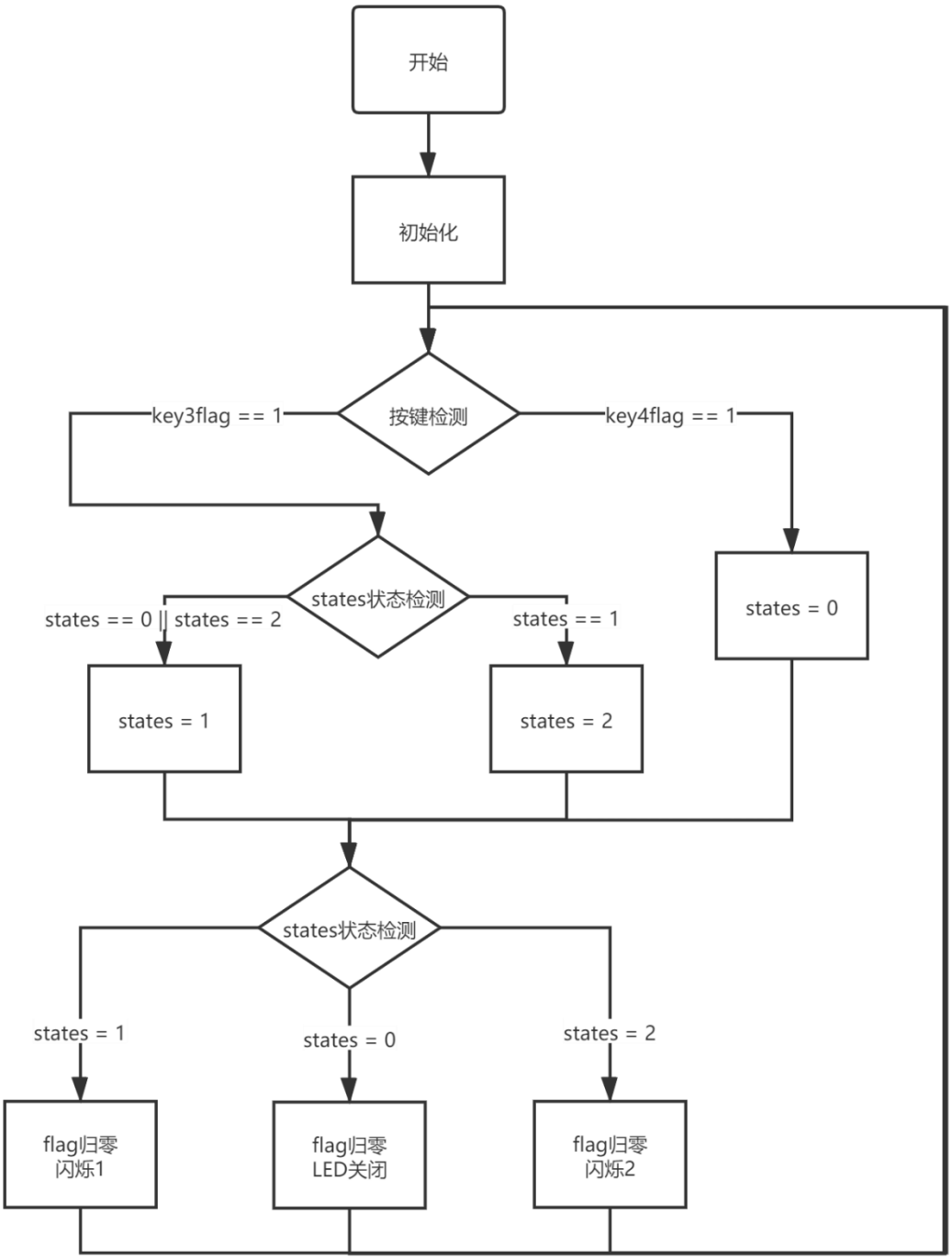
四、程序流程图和核心语句

核心语句：

```
unsigned char key3flag, key4flag;
static void
BoardInit(void)
{
#ifndef USE_TIRTOS
#if defined(ccs)
    MAP_IntVTableBaseSet((unsigned long)&g_pfnVectors[0]);
#endif
#if defined(ewarm)
    MAP_IntVTableBaseSet((unsigned long)&__vector_table);
#endif
#endif
    MAP_IntMasterEnable();
    MAP_IntEnable(FAULT_SYSTICK);

    PRCMCC3200MCUInit();
}

void Key3Handler(void)
{
    unsigned long ulPinState = GPIOIntStatus(GPIOA2_BASE, 1);
    if (ulPinState & GPIO_INT_PIN_7)
    {
```

```
        GPIOIntClear(GPIOA2_BASE, GPIO_INT_PIN_7);
        key3flag = 1;
    }
}

void Key4Handler(void)
{
    unsigned long ulPinState = GPIOIntStatus(GPIOA3_BASE, 1);
    if (ulPinState & GPIO_INT_PIN_0)
    {
        GPIOIntClear(GPIOA3_BASE, GPIO_INT_PIN_0);
        key4flag = 1;
    }
}

void Keyinit(void)
{
    GPIOIntTypeSet(GPIOA2_BASE, GPIO_PIN_7, GPIO_RISING_EDGE);
    IntPrioritySet(INT_GPIOA2, INT_PRIORITY_LVL_1);
    GPIOIntRegister(GPIOA2_BASE, Key3Handler);
    GPIOIntClear(GPIOA2_BASE, GPIO_INT_PIN_7);
    IntPendClear(INT_GPIOA2);
    IntEnable(INT_GPIOA2);
    GPIOIntEnable(GPIOA2_BASE, GPIO_INT_PIN_7);

    GPIOIntTypeSet(GPIOA3_BASE, GPIO_PIN_0, GPIO_RISING_EDGE);
    IntPrioritySet(INT_GPIOA3, INT_PRIORITY_LVL_1);
    GPIOIntRegister(GPIOA3_BASE, Key4Handler);
    GPIOIntClear(GPIOA3_BASE, GPIO_INT_PIN_0);
    IntPendClear(INT_GPIOA3);
    IntEnable(INT_GPIOA3);
    GPIOIntEnable(GPIOA3_BASE, GPIO_INT_PIN_0);
}
void main(void)
{
    BoardInit();
    PinMuxConfig();
    Keyinit();
    GPIO_IF_LedConfigure(LED1 | LED2 | LED3);
    int states = 0;

    while (1)
    {
        if (key3flag == 1)
```

```
        {
            key3flag = 0;
            if (states == 0 || states == 2)
                states = 1;
            if (states == 1)
                states = 2;
        }
        if (key4flag == 1)
        {
            key4flag = 0;
            states = 0;
        }
        if (states == 0)
            GPIO_IF_LedOff(MCU_ALL_LED_IND);
        if (states == 1)
        {
            GPIO_IF_LedOn(MCU_RED_LED_GPIO);
            MAP_UtilsDelay(800000);
            GPIO_IF_LedOff(MCU_RED_LED_GPIO);
            MAP_UtilsDelay(800000);
            GPIO_IF_LedOn(MCU_ORANGE_LED_GPIO);
            MAP_UtilsDelay(800000);
            GPIO_IF_LedOff(MCU_ORANGE_LED_GPIO);
            MAP_UtilsDelay(800000);
            GPIO_IF_LedOn(MCU_GREEN_LED_GPIO);
            MAP_UtilsDelay(800000);
            GPIO_IF_LedOff(MCU_GREEN_LED_GPIO);
            MAP_UtilsDelay(800000);
            GPIO_IF_LedOn(MCU_ORANGE_LED_GPIO);
            MAP_UtilsDelay(800000);
            GPIO_IF_LedOff(MCU_ORANGE_LED_GPIO);
            MAP_UtilsDelay(800000);
        }
        if (states == 2)
        {
            GPIO_IF_LedOn(MCU_ALL_LED_IND);
            MAP_UtilsDelay(500000);
            GPIO_IF_LedOff(MCU_ALL_LED_IND);
            MAP_UtilsDelay(500000);
        }
    }
    return 0;
}
```



五、设计过程中遇到的问题和解决方法

①如何更改 pinmux 的设置

使用 TI Pin Mux Tool 软件，仅选中三个 LED 灯和两个按键的 PIN 口，再生成导出 pinmux.c 和 pinmux.h 文件

②电路版按键问题

在刚开始做实验的时候，有一部分电路板的 sw2 和 sw3 按键会出问题，通过更换其他同学的电路板，发现自己的板子确实有问题。

六、思考问题解答、收获和建议

1、GPIO 的基本操作有哪些?

①GPIO_Mode_AIN 模拟输入

②GPIO_Mode_IN_FLOATING 浮空输入

③GPIO_Mode_IPD 下拉输入

④GPIO_Mode_IPU 上拉输入

⑤GPIO_Mode_Out_OD 开漏输出

⑥GPIO_Mode_Out_PP 推挽输出

⑦GPIO_Mode_AF_OD 复用开漏输出

⑧GPIO_Mode_AF_PP 复用推挽输

2、CC3200 外设中断应用需要怎么编程设计?

①设置引脚为 GPIO。

②使能时钟。

③设置引脚电流的强度，上拉或者下拉等等。

④设置 GPIO 为输入模式。

⑤设置成边沿触发。

⑥编写中断处理函数：1、判断哪个 GPIO 触发，2、清除中断标志。

⑦使能中断触发。

CC3200 有 4 组 GPIO，每组的 GPIO 共用一个中断处理函数，需要在中断函数查询是哪个 GPIO 触发中断的。

中断配置需要的库函数：

GPIOIntTypeSet(GPIOA1_BASE,GPIO_PIN_5,GPIO_RISING_EDGE);

配置 GPIO A1 组的 pin 5 设置成上升边沿触发中断，类似的还有下拉。

GPIOIntRegister(GPIOA1_BASE,GPIO13_handle);

配置 GPIOA1 组的中断处理函数 GPIO13_handle.

GPIOIntEnable(GPIOA1_BASE,GPIO_INT_PIN_5);

使能 GPIOA1 的 pin5 触发中断。

中断函数需要调用的函数。

GPIOIntStatus(GPIOA1_BASE,true);

查询 GPIOA1 的中断状态寄存器：注意查询的是组内的 GPIO 的中断状态。获得的值跟 GPIO_INT_PIN_5 相与，就可以获取是不是 pin5 触发中断的。

GPIOIntClear(GPIOA1_BASE,GPIO_INT_PIN_5);

清除 GPIOA1 的 pin5 的中断标志。