使用 GPIO 输入功能实现按键扫描，来控制 LED 灯的亮灭

一．需要配置的寄存器

1. 配置相关引脚时钟，包括与 LED 和按键 key 相连的 IO 口：RCC->APB2ENR
2. 配置相关引脚的工作模式（LED 强制推挽输出，key 如果没有外接上拉或者下拉电阻则在内部设置为上拉或下拉输入模式，如果 key 外接上拉或下拉电阻则设置为浮空输入模式） GPIOB->CRL (key 用的是 GIOB.0)
3. 获取按键 KEY 的值并进行判断

二．详细说明（按键使用的是 GPIOB 的 0 脚没有外接上拉或者下拉电阻，LED 是 GPIOC 的 1 脚）

第一步 配置 LED 和 key 的时钟

6.3.7 APB2 外设时钟使能寄存器(RCC_APB2ENR)

偏移地址：0x18

复位值：0x0000 0000

访问：字，半字和字节访问

通常无访问等待周期。但在APB2总线上的外设被访问时，将插入等待状态直到APB2的外设访问结束。

注： 当外设时钟没有启用时，软件不能读出外设寄存器的数值，返回的数值始终是0x0。

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADC3 EN	USART1 EN	TIM8 EN	SPI1 EN	TIM1 EN	ADC2 EN	ADC1 EN	IOPG EN	IOPF EN	IOPE EN	IOPD EN	IOPC EN	IOPB EN	IOPA EN	保留	AFIO EN
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw

这里配置时钟和配置 LED 是一样的把对应的位写上 1 即可（补充一个 LED 那个文档的一个问题:在 LED 那个笔记里说给某位写 1 要先清零再写入，感觉没有必要清零，使用的**位或**方法不管原来里面是 1 还是 0 或上 1 还是 1。给某位写 0 一样，只要和 0 **位与**就可以）

我们要给 GPIOC 和 GPIOB 时钟使能就位或一个数就可以要给 GPIOC 使能

```
RCC->APB2ENR |= 0x00000010;
```

给 GPIOB 使能

```
RCC->APB2ENR |=0X00000008;
```

也可以写在一起(不推荐，因为修改程序不方便，上面的哪个不用注释掉就好，写在一起需要修改数字容易出错)

```
RCC->APB2ENR |=0X00000018;
```

第二步 配置 IO 口的工作模式

需要配置两东西，LED 的相关配置可以使用第一篇文章配置好的，key 根据自己硬件分成两种情况

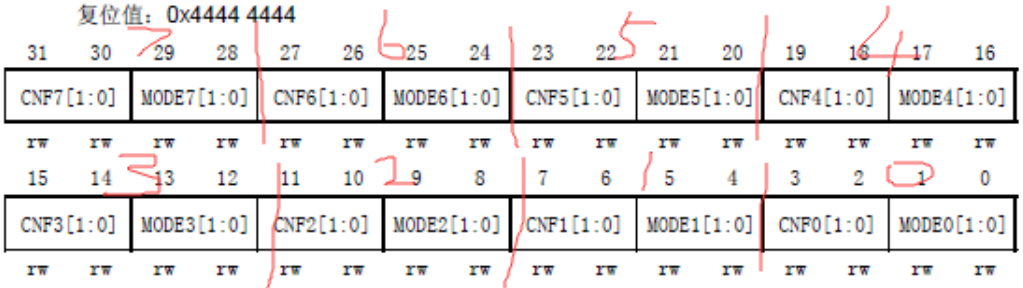
1.有外界上拉/下拉电阻配置成 浮空输入

2.无外部上拉/下拉电阻配置成 上拉/下拉输入模式

8.2.1 端口配置低寄存器(GPIOx_CRL) (x=A..E)

偏移地址：0x00

复位值：0x4444 4444



31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNF7[1:0]		MODE7[1:0]		CNF6[1:0]		MODE6[1:0]		CNF5[1:0]		MODE5[1:0]		CNF4[1:0]		MODE4[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNF3[1:0]		MODE3[1:0]		CNF2[1:0]		MODE2[1:0]		CNF1[1:0]		MODE1[1:0]		CNF0[1:0]		MODE0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位31:30 27:26 23:22 19:18 15:14 11:10 7:6 3:2	**CNFy[1:0]**：端口x配置位(y = 0...7) (Port x configuration bits) 软件通过这些位配置相应的I/O端口，请参考表17端口位配置表。 在输入模式(MODE[1:0]=00)： 00：模拟输入模式 01：浮空输入模式(复位后的状态) 10：上拉/下拉输入模式 11：保留 在输出模式(MODE[1:0]>00)： 00：通用推挽输出模式 01：通用开漏输出模式 10：复用功能推挽输出模式 11：复用功能开漏输出模式
位29:28 25:24 21:20 17:16 13:12 9:8, 5:4 1:0	**MODEy[1:0]**：端口x的模式位(y = 0...7) (Port x mode bits) 软件通过这些位配置相应的I/O端口，请参考表17端口位配置表。 00：输入模式(复位后的状态) 01：输出模式，最大速度10MHz 10：输出模式，最大速度2MHz 11：输出模式，最大速度50MHz

GPIOC.1 对应 LED 配置成强制推挽输出不再

```
GPIOC->CRL&=0XFFFFFF0F;
GPIOC->CRL|=0X00000030;// CNF1=00 ,MODE1=11 合在一起是 3
```

KEY 对应的引脚是 GPIOB.0 分成两种情况

1. 外接有上拉/下拉电阻

   外接有上拉/下拉电阻配置成浮空输入

   ```
   GPIOB->CRL&=0XFFFFFFF0;
   GPIOB->CRL|=0x00000004;//CNF0=01,MODE0=00 合在一起是 4
   ```

2. 外接没有上拉/下拉电阻

   KEY 对应引脚配置成上拉/下拉输入模式

   ```
   GPIOB->CRL&=0XFFFFFFF0;
   GPIOB->CRL|=0X00000008;//CNF0=10,MODE0=00 合在一起是 8
   ```

   到这里只是设置成了上拉/下拉输入模式。到底是上拉还是下拉要还看其他的寄存器

表17　端口位配置表

配置模式		CNF1	CNF0	MODE1	MODE0	PxODR寄存器
通用输出	推挽(Push-Pull)	0	0	01 10 11 见表18		0 或 1
	开漏(Open-Drain)		1			0 或 1
复用功能输出	推挽(Push-Pull)	1	0			不使用
	开漏(Open-Drain)		1			不使用
输入	模拟输入	0	0	00		不使用
	浮空输入		1			不使用
	下拉输入	1	0			0
	上拉输入					1

从上图可以看出设置上拉输入还是下拉输入需要除了设置 CNF 和 MODE 还要设置 ODR 这个寄存器。

8.2.4　端口输出数据寄存器(GPIOx_ODR) (x=A..E)

地址偏移：0Ch

复位值：0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ODR15	ODR14	ODR13	ODR12	ODR11	ODR10	ODR9	ODR8	ODR7	ODR6	ODR5	ODR4	ODR3	ODR2	ODR1	ODR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位31:16	保留，始终读为0。
位15:0	**ODRy[15:0]**：端口输出数据(y = 0…15) (Port output data) 这些位可读可写并只能以字(16位)的形式操作。 注：对GPIOx_BSRR(x = A…E)，可以分别地对各个ODR位进行独立的设置/清除。

这个寄存器是用来给 IO 口输出数据的。现在需要它来设置上拉下拉（我的理解是如果需要下拉就向对应的引脚写 0 让这个引脚默认是低电平，效果和在外部下拉电阻一样，想设置成上拉电阻就和向该位写 1 就等于把该位的电平默认设置为高电平）

所以要使用内部的上拉/下拉输入模式需要

GPIOB->ODR&=0xFFFFFFFe;//GPIOB.0 下拉

或者 GPIOB->ODR|=0X00000001;//GPIOB.0 上拉

综合一下我的板子上 GPIOB.0 连接的 key 没有上拉电阻所以我使用内部上拉输入，引脚配置如下

```
GPIOB->CRL&=0XFFFFFFF0;
GPIOB->CRL|=0X00000008;//上拉/下拉输入模式
GPIOB->ODR|=0X00000001;//上拉模式
```

第三步 检测按键状态控制 LED 亮灭

现在引脚已经配置好了，我们按下按键 STM32 可以检测到按键按下接下来就是需要读出按键的值来判断按键是按下还是抬起。

这里需要一个寄存器

8.2.3 端口输入数据寄存器(GPIOx_IDR) (x=A..E)

地址偏移：0x08

复位值：0x0000 XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IDR15	IDR14	IDR13	IDR12	IDR11	IDR10	IDR9	IDR8	IDR7	IDR6	IDR5	IDR4	IDR3	IDR2	IDR1	IDR0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

位31:16	保留，始终读为0。
位15:0	**IDRy[15:0]**：端口输入数据(y = 0…15) (Port input data) 这些位为只读并只能以字(16位)的形式读出。读出的值为对应I/O口的状态。

这个寄存器就是保存 IO 口输入数据寄存器 0~15 表示一组端口的 0~15 个引脚我们这里是 GPIOB.0 所以判断 0 号位的值就可以知道按键是否按下

If(GPIOB->IDR&0x0001==0)//把寄存器的值和 0x0001 位与判断最后一位是否为 0 是
//的话会判断条件为真就执行 if 里面的语句

GPIOC->BSRR|=0X00020000;//点亮 led

else GPIOC->BSRR|=OX00000002;//灭掉 LED

以上就是用按键控制 LED 的学的过程。

示例：LED ——GPI0C.1　　key——GPIOB.0

```
#include “stm32f103x.h”
Int main()
{
    RCC->APB2ENR|=0X00000010;
    RCC->APB2ENR|=0X00000008;

    GPIOB->CRL&=0XFFFFFFF0;
    GPIOB->CRL|=0x00000004;
    GPIOB->ODR|=0x00000001;

    GPIOC->CRL&=0XFFFFFF0F;
    GPIOC->CRL|=0X00000030;

    while(1)
    {

        if(GPIOB->IDR&0X00000001==0)
        {
            GPIOC->BSRR=0x00020000;
        }
        else GPIOC->BSRR=0x00000002;
    }}
```