Key（查询模式）实验

作者	fire
E-Mail	firestm32@foxmail.com
QQ	313303034
博客	firestm32.blog.chinaunix.net
硬件平台	野火 STM32 开发板
库版本	ST3.0.0

实验描述：PB0 连接到 key1，用扫描的方式查询是否有按键按下，key1 按下时，LED1 状态取反。

硬件连接：PB0 - key1

PB1 – key2

库文件 ：startup/start_stm32f10x_hd.c

CMSIS/core_cm3.c

CMSIS/system_stm32f10x.c

FWlib/stm32f10x_gpio.c

FWlib/stm32f10x_rcc.c

FWlib/misc.c

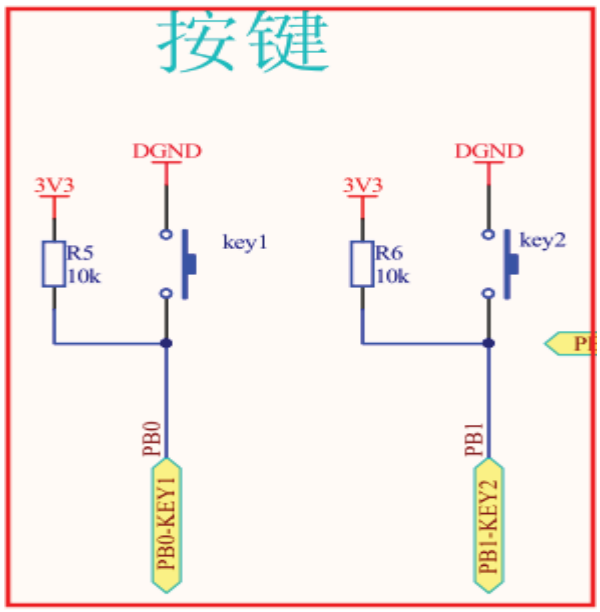
用户文件：USER/main.c

USER/stm32f10x_it.c

USER/led.c

USER/key.c

野火 **STM32** 开发板 按键 硬件原理图：



实验讲解->

我们从 main 函数开始分析：

```
1.  /**
2.    * @brief  Main program.
3.    * @param  None
4.    * @retval : None
5.    */
6.  int main(void)
7.  {
8.      /* config the sysclock to 72m */
9.    SystemInit();
10.
11.     /* config the led */
12.   LED_GPIO_Config();
13.     LED1( ON );
14.       /*config key*/
15.     Key_GPIO_Config();
16.
17.
18.   while(1)
19.   {
20.     if( Key_Scan(GPIOB,GPIO_Pin_0) == KEY_ON  )
21.     {
22.             /*LED1 反转*/
23.             GPIO_WriteBit(GPIOC, GPIO_Pin_3,
24.                       (BitAction)((1-
    GPIO_ReadOutputDataBit(GPIOC, GPIO_Pin_3))));
25.     }
26.   }
27. }
```

首先调用库函数 SystemInit()将我们的系统时钟配置为 72MHZ， LED_GPIO_Config()配置 LED 用到的 I/O。这两个函数的具体讲解可参考前面的教程。现在我们分析一下 Key_GPIO_Config() 这个函数。

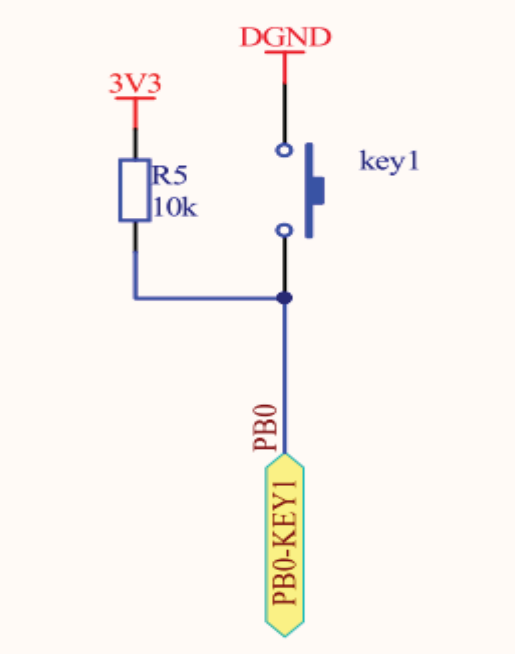
```
/*
 * 函数名：Key_GPIO_Config
 * 描述  ：配置按键用到的 I/O 口
 * 输入  ：无
 * 输出  ：无
 */
void Key_GPIO_Config(void)
{
  GPIO_InitTypeDef GPIO_InitStructure;

    /*开启按键端口（PB0）的时钟*/
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB,ENABLE);

    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_10MHz;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPU;

    GPIO_Init(GPIOB, &GPIO_InitStructure);
}
```

跟 LED 的 I/O 端口设置类似，只是 PB0 的模式设置为适合按键处理的上拉输入方式。在外围电路上我们将 PB0 接到了 key1 上：



当按键没有按下时，PB0 始终为高，当按键按下时 PB0 变为低，从而 PB0 上产生一个低电平跳变，在按键扫描函数中可以检测到电平的变化，通过消抖处理，对按键消息进行确认。

```
/*
 * 函数名：Key_Scan(GPIO_TypeDef* GPIOx,u16 GPIO_Pin)
 * 描述  ：检测是否有按键按下
 * 输入  ：GPIOx：x 可以是 A，B，C，D 或者 E
           GPIO_Pin：待读取的端口位
 * 输出  ：KEY_OFF(没按下按键)、KEY_ON（按下按键）
 */
u8 Key_Scan(GPIO_TypeDef* GPIOx,u16 GPIO_Pin)
{
        /*检测是否有按键按下 */
    if(GPIO_ReadInputDataBit(GPIOx,GPIO_Pin) == KEY_ON )
      {
         /*延时消抖*/
        Delay(10000);
            if(GPIO_ReadInputDataBit(GPIOx,GPIO_Pin) == KEY_ON )
                    {
                        /*等待按键释放 */
         while(GPIO_ReadInputDataBit(GPIOx,GPIO_Pin) == KEY_ON);
                        return  KEY_ON;
                    }
            else
                    return KEY_OFF;
        }
    else
        return KEY_OFF;
}
```

相信消抖的原理大家已经在学习其它单片机的时候已经了解，这里主要介绍一下 Key_Scan()的形参。这个函数看起来是不是很像 ST 官方的库函数？其实这是野火自己写的一个用户函数。^_^

这里利用了在 stm32f10x.h 文件中的数据 GPIO 类型定义。形参 GPIO_Pin 也是样实现的，所以如果在调用 Key_Scan()函数时，把实参改成 GPIOB，GPIO_Pin_1,就可以用 Key-2 来控制 LED1 啦，是不是很方便呢，利用官方的库，我们可以很方便地开发出这一类用户函数，这就是库的魅力呀！

```
typedef struct
{
  __IO uint32_t CRL;
  __IO uint32_t CRH;
  __IO uint32_t IDR;
  __IO uint32_t ODR;
  __IO uint32_t BSRR;
  __IO uint32_t BRR;
  __IO uint32_t LCKR;
} GPIO_TypeDef;
```

还有实现 LED 反转的代码

```
GPIO_WriteBit(GPIOC, GPIO_Pin_3,
                  (BitAction)((1-
GPIO_ReadOutputDataBit(GPIOC, GPIO_Pin_3))));
```

这是利用库函数来读出 LED1 端口状态再反转。其实还可以使用位带操作的方式，实现反转起来和使用 51 的端口一样简单直接比如：PA0=~PA0；

实验现象->

将野火 STM32 开发板供电(DC5V)，插上 JLINK，将编译好的程序下载到开发板，LED1 亮，按下按键时 LED1 灭，再按下按键时 LED1 灭，如此循环。

实验讲解完毕，野火祝大家学习愉快^_^。