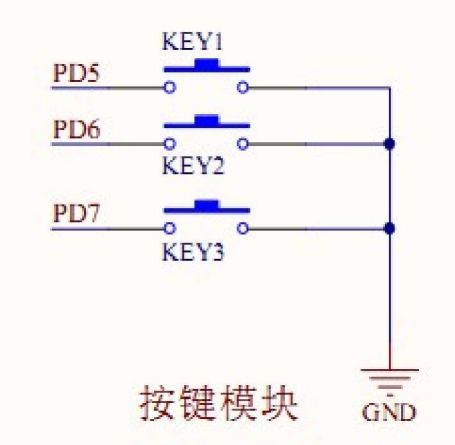
实验二　按键扫描

按键扫描实际上就是对 IO 口的输入捕捉操作，跟 LED 的刚好相反，LED 的对 IO 口输出的操作，因此在操作上很相似的。相信大家会点亮 LED 的话，那么按键扫描的话也很容易了。先看下风驰电子 STM8 开发板上的按键的硬件连接



由电路连接图来看，如果按键按下的话，IO 口读到的电平信号就是低电平。
先看看按键扫描要用到的内部资源

"stm8s.h"

"stm8s_clk.h"

"stm8s_clk.c"

"stm8s_gpio.h"

"stm8s_gpio.c"

首先，我们从主函数看起

```
int main(void)
{
  /* Infinite loop */

  /*设置内部时钟16M为主时钟*/

  CLK_HSIPrescalerConfig(CLK_PRESCALER_HSIDIV1);
  /*!<Set High speed internal clock  */

  Buttom_Init();
  LED_Init();
  SetLedOFF();
  while (1)
  {
    /*  添加你的代码  */
      if(!Key_Scan( Buttom1))Led_Reverse(led1);

  }
}
```

时钟和 LED 的初始话大家都清楚了，这里不多说了。下面主要来看下按键的初始化 Buttom_Init();

他的函数原型

```
void Buttom_Init(void)
{
    GPIO_Init(GPIOD,Buttom1|Buttom2|Buttom3,GPIO_MODE_IN_PU_NO_IT);
}
```

很简单，对吧，就是初始话一下 IO 口。上面的意思是对按键 Buttom1、Buttom2、Buttom3 的相应 IO 口初始话为上拉输入，没触发中断。

GPIO_Init(GPIOD,Buttom1|Buttom2|Buttom3,GPIO_MODE_IN_PU_NO_IT);函数原型

```
/**
  * @brief  Initializes the GPIOx according to the specified parameters.
  * @param  GPIOx : Select the GPIO peripheral number (x = A to I).
  * @param  GPIO_Pin : This parameter contains the pin number, it can be any value
  *         of the @ref GPIO_Pin_TypeDef enumeration.
  * @param  GPIO_Mode : This parameter can be a value of the
  *         @Ref GPIO_Mode_TypeDef enumeration.
  * @retval None
  */

void GPIO_Init(GPIO_TypeDef* GPIOx, GPIO_Pin_TypeDef GPIO_Pin, GPIO_Mode_TypeDef GPIO_Mode)
{
    /*-----------------------*/
    /* Check the parameters */
    /*-----------------------*/

    assert_param(IS_GPIO_MODE_OK(GPIO_Mode));
    assert_param(IS_GPIO_PIN_OK(GPIO_Pin));

  /* Reset corresponding bit to GPIO_Pin in CR2 register */
  GPIOx->CR2 &= (uint8_t)(~(GPIO_Pin));


    /*-----------------------------*/
    /* Input/Output mode selection */
    /*-----------------------------*/

    if ((((uint8_t)(GPIO_Mode)) & (uint8_t)0x80) != (uint8_t)0x00) /* Output mode */
    {
        if ((((uint8_t)(GPIO_Mode)) & (uint8_t)0x10) != (uint8_t)0x00) /* High level */
        {
            GPIOx->ODR |= (uint8_t)GPIO_Pin;
        }
        else /* Low level */
        {
            GPIOx->ODR &= (uint8_t)(~(GPIO_Pin));
        }
        /* Set Output mode */
        GPIOx->DDR |= (uint8_t)GPIO_Pin;
    }
    else /* Input mode */
    {
        /* Set Input mode */
        GPIOx->DDR &= (uint8_t)(~(GPIO_Pin));
    }
```

```
    /*-----------------------------------------------------------------------*/
    /* Pull-Up/Float (Input) or Push-Pull/Open-Drain (Output) modes selection */
    /*-----------------------------------------------------------------------*/

    if ((((uint8_t)(GPIO_Mode)) & (uint8_t)0x40) != (uint8_t)0x00) /* Pull-Up or Push-Pull */
    {
        GPIOx->CR1 |= (uint8_t)GPIO_Pin;
    }
    else /* Float or Open-Drain */
    {
        GPIOx->CR1 &= (uint8_t)(~(GPIO_Pin));
    }

    /*-----------------------------------------------------*/
    /* Interrupt (Input) or Slope (Output) modes selection */
    /*-----------------------------------------------------*/

    if ((((uint8_t)(GPIO_Mode)) & (uint8_t)0x20) != (uint8_t)0x00) /* Interrupt or Slow slope
    {
        GPIOx->CR2 |= (uint8_t)GPIO_Pin;
    }
    else /* No external interrupt or No slope control */
    {
        GPIOx->CR2 &= (uint8_t)(~(GPIO_Pin));
    }
}
```

按键的宏定义

```
#define Buttom1 GPIO_PIN_5
#define Buttom2 GPIO_PIN_6
#define Buttom3 GPIO_PIN_7
```

设置的模式的定义

```
typedef enum
{
  GPIO_MODE_IN_FL_NO_IT      = (uint8_t)0x00,  /*!< Input floating, no external interrupt */
  GPIO_MODE_IN_PU_NO_IT      = (uint8_t)0x40,  /*!< Input pull-up, no external interrupt */
  GPIO_MODE_IN_FL_IT         = (uint8_t)0x20,  /*!< Input floating, external interrupt */
  GPIO_MODE_IN_PU_IT         = (uint8_t)0x60,  /*!< Input pull-up, external interrupt */
  GPIO_MODE_OUT_OD_LOW_FAST  = (uint8_t)0xA0,  /*!< Output open-drain, low level, 10MHz */
  GPIO_MODE_OUT_PP_LOW_FAST  = (uint8_t)0xE0,  /*!< Output push-pull, low level, 10MHz */
  GPIO_MODE_OUT_OD_LOW_SLOW  = (uint8_t)0x80,  /*!< Output open-drain, low level, 2MHz */
  GPIO_MODE_OUT_PP_LOW_SLOW  = (uint8_t)0xC0,  /*!< Output push-pull, low level, 2MHz */
  GPIO_MODE_OUT_OD_HIZ_FAST  = (uint8_t)0xB0,  /*!< Output open-drain, high-impedance level,10MHz */
  GPIO_MODE_OUT_PP_HIGH_FAST = (uint8_t)0xF0,  /*!< Output push-pull, high level, 10MHz */
  GPIO_MODE_OUT_OD_HIZ_SLOW  = (uint8_t)0x90,  /*!< Output open-drain, high-impedance level, 2MHz */
  GPIO_MODE_OUT_PP_HIGH_SLOW = (uint8_t)0xD0   /*!< Output push-pull, high level, 2MHz */
}GPIO_Mode_TypeDef;
```

下面看看按键扫描的函数原型

```
BitStatus Key_Scan(GPIO_Pin_TypeDef Buttom)
{
    BitStatus ButtomStatus;
    ButtomStatus=Buttom_OFF;
    if(!GPIO_ReadInputPin(GPIOD,Buttom))
    {
        Delay(0x3ff);/*  消抖 */
        if(!GPIO_ReadInputPin(GPIOD,Buttom))
        {
          while(!GPIO_ReadInputPin(GPIOD,Buttom));/*松手释放*/
          ButtomStatus=Buttom_ON;
        }
    }
    return ButtomStatus;
}
```

这个函数是当有按键按下的话就返回 Buttom_ON,否则的话就返回 Buttom_OFF。
他们的宏定义

```
#define Buttom_ON    0
#define Buttom_OFF   1
```

下面在看看另外一个函数

```
void Led_Reverse(GPIO_Pin_TypeDef LedPins)
{
   GPIO_WriteReverse(GPIOD, LedPins);
}
```

这个函数直接是对 LED 的管脚的值直接取反。
GPIO_WriteReverse(GPIOD, LedPins);函数原型

```
/**
  * @brief  Writes reverse level to the specified GPIO pins.
  * @note   The port must be configured in output mode.
  * @param  GPIOx : Select the GPIO peripheral number (x = A to I).
  * @param  PortPins : Specifies the pins to be reversed to the port output.
  *         data register.
  * @retval None
  */
void GPIO_WriteReverse(GPIO_TypeDef* GPIOx, GPIO_Pin_TypeDef PortPins)
{
    GPIOx->ODR ^= (uint8_t)PortPins;
}
```

所以 Led_Reverse(led1);是和 51 单片机里面的 led1=~led1 是一样的。也就是对该管脚的值取反。所以在主函数里面的

```
/*  添加你的代码  */
  if(!Key_Scan( Buttom1))Led_Reverse(led1);
```

的意思说 Buttom1 按键按一下灯 LED1 就会亮，再按一下，灯就会灭，不断循环。
实验现象：
当你把例程下载到风驰电子 STM8 开发板上，你就会看到每当你按一下 Buttom1(Key1)，灯 LED1 就会亮，再按一下就会灭，如此循环。

风驰电子祝您学习愉快～～～!!!!!