

实验 3 - GPIO 输入按键检测

1. 实验目的

掌握 NRF52832 的 GPIO 的配置方式和输入检测。

2. 实验内容

配置 NRF52832 的 GPIO P0.17~ P0.20 为输出驱动 LED 指示灯 D1~D4。

配置 NRF52832 的 GPIO P0.13~ P0.16 为输入检测按键 S1~S4 的状态。

程序运行后，按下开发板上的按键 S1~S4，对应的指示灯 D1~D4 会点亮，释放按键后，指示灯熄灭。

3. 实验设备

硬件	
1.	IK-52832DK 开发板
2.	USB MINI 数据线
3.	JLINK 仿真器
4.	JTAG-SWD 转接板、排线
软件	
1.	win7/win8.1 系统
2.	MDK5.18A 集成开发环境

4. 实验原理

4.1. 电路原理

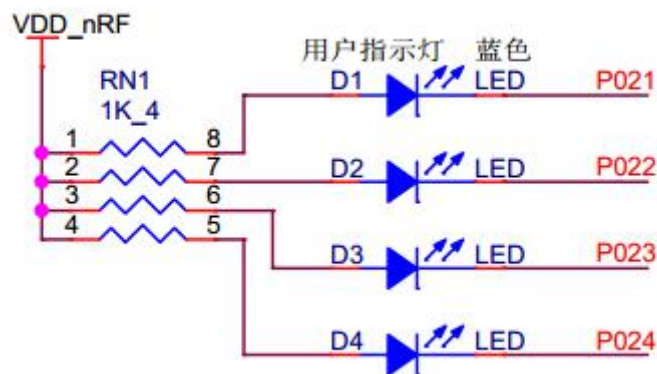


图 1：指示灯驱动电路

开发板上配置的四个用户指示灯 D1、D2、D3、D4，分别有 GPIO P0.17、P0.18、P0.19 和 P0.20 控制，当 GPIO 输出高电平时，LED 两端电压相等，LED 上没有电流流过，LED 处于熄灭状态，当 GPIO 输出低电平时，LED 两端存在正向压差，电流流过 LED，LED 被点亮。

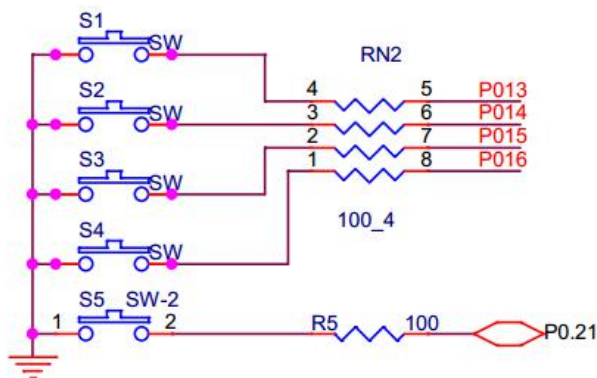


图 2：按键输入 (S1~S4) 检测电路

开发板上配置了 4 个用户按键 S1、S2、S3、S4，分别连接到 GPIO P0.13~P0.16。GPIO 用作输入时，需要通过软件打开 GPIO 的上拉电阻，用于确定 IO 口状态，在按键释放时保证 IO 口状态为高电平。串接的电阻用于：

- 保护 IO，若 IO 口不小心被配置成了输出，按下按键可能会损坏 IO，串接电阻后，即使出现这种情况，也不会损坏 IO。
- 降低按键时产生的抖动峰值电压。

4.2. GPIO 配置

1. 配置 P0.21~P0.24 为输出：

本实验中，GPIO P0.21 用于驱动指示灯 D1 的亮灭，所以需要将 P0.17~P0.20 配置为输出。之后通过操作寄存器 OUTSET 和 OUTCLR 控制指示灯的亮灭。

- 通过 PIN_CNF[n] 寄存器配置 P0.17~P0.20 为输出。
- 通过向 OUTSET 寄存器相应的位写 1 让 P0.17~P0.20 输出高电平，熄灭指示灯。
- 通过向 OUTCLR 寄存器相应的位写 1 让 P0.17~P0.20 输出低电平，点亮指示灯。

各个寄存器的功能如下：

- PIN_CNF[n]：GPIO 配置寄存器，其中的 n 代表硬件的引脚 0~31。每一个 PIN 单独对应一个该寄存器，其主要用来配置引脚的方向、上拉下拉、驱动配置、以及检测配置等。
- OUTSET：0~31 位对应于管脚 P0.00~P0.31，如要某个管脚输出高电平，向对应的位写“1”即可，写“0”无效。
- OUTCLR：0~31 位对应于管脚 P0.00~P0.31，如要某个管脚输出低电平，向对应的位写“1”即可，写“0”无效。

在本实验中，我们使用另外一个 GPIO 配置函数 `nrf_gpio_range_cfg_output` 来配置 GPIO 为输出，`nrf_gpio_range_cfg_output` 适合用来配置管脚号连续的多个 GPIO。程序如下：

`nrf_gpio_range_cfg_output(LED_START, LED_STOP);` //配置 P0.17~P0.20 为输出
其中 LED_START 和 LED_STOP 的定义在“pca10040.h”文件中，如下图：

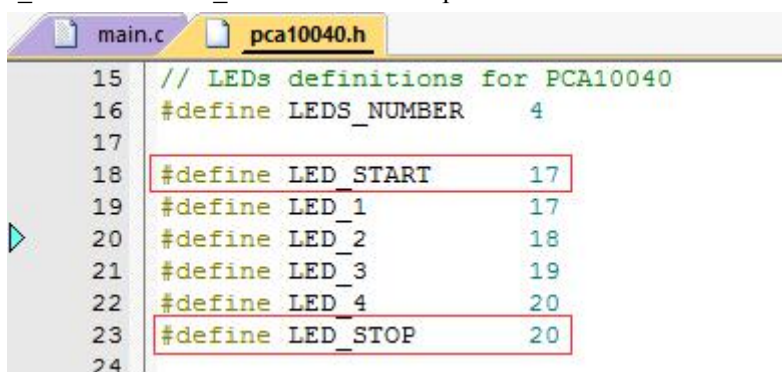


图 3：LED 相关引脚定义

进入 `nrf_gpio_range_cfg_output` 函数，其代码如下：

```
__STATIC_INLINE void nrf_gpio_range_cfg_output(uint32_t
pin_range_start, uint32_t pin_range_end)
{
    for (; pin_range_start <= pin_range_end; pin_range_start++)
    {
        nrf_gpio_cfg_output(pin_range_start);
    }
}
```

上述代码执行后，GPIO P0.17~P0.20 被配置为：

- 电平检测禁止。
- 标准驱动。
- 上拉禁止。
- 输入缓冲断开。
- 方向配置为输出。

2. 配置 P0.13~P0.16 为输入：

和配置 GPIO 为输出一样，我们使用 GPIO 配置函数 `nrf_gpio_range_cfg_input` 来配置 GPIO 为输入，这个函数和“`nrf_gpio_range_cfg_output`”功能类似，适合用来配置管脚号连续的多 GPIO 为输入。程序如下：

```
nrf_gpio_range_cfg_input(BUTTON_START, BUTTON_STOP, NRF_GPIO_PIN_PULLUP);
//配置 P0.13~P0.16 为输入，同时，使能上拉
```

其中 BUTTON_START 和 BUTTON_STOP 的定义在“pca10040.h”文件中，如下图：

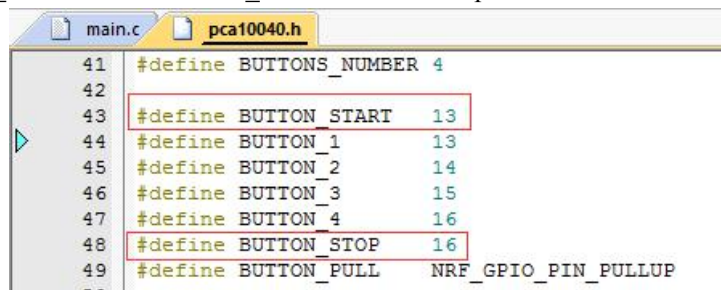


图 4：按键相关定义

进入 `nrf_gpio_range_cfg_input` 函数，其代码如下：

```
__STATIC_INLINE void nrf_gpio_range_cfg_input(uint32_t pin_range_start,
uint32_t pin_range_end, nrf_gpio_pin_pull_t pull_config)
{
    for (; pin_range_start <= pin_range_end; pin_range_start++)
    {
        nrf_gpio_cfg_input(pin_range_start, pull_config);
    }
}
```

上述代码执行后，GPIO P0.13~P0.16 被配置为：

- 电平检测禁止。
- 标准驱动。
- 上拉打开。
- 连接输入。
- 方向配置为输入。

4.3. 驱动 LED

GPIO P0.17 配置好后，即可通过 P0.17 输出高低电平控制指示灯 D1 的亮灭，调用下列函数即可实现 P0.17 输出高低电平。

1. `nrf_gpio_pin_set(LED_1);`

进入该函数，其代码如下：

```
__STATIC_INLINE void nrf_gpio_pin_set(uint32_t pin_number)
{
    NRF_GPIO->OUTSET = (1UL << pin_number);
}
```

从上面的代码可以看出，该函数正是通过向 **OUTSET** 寄存器相应的位写“1”来实现 GPIO 输出高电平的，和我们前面描述的一致。

2. `nrf_gpio_pin_clear(LED_1);`

功能：根据输入的管脚号设置该管脚输出低电平。

3. `nrf_gpio_pin_toggle(LED_1);`

功能：根据输入的管脚号翻转该管脚的输出状态。在对管脚状态取反时调用该函数更方便。

4.4. 检测按键状态

P0.13~P0.16 配置为输入后，即可通过调用读管脚状态函数获取管脚状态，函数如下：

`nrf_gpio_pin_read(BUTTON_1)`

进入该函数，其代码如下：

```
__STATIC_INLINE uint32_t nrf_gpio_pin_read(uint32_t pin_number)
{
    return ((NRF_GPIO->IN >> pin_number) & 1UL);
}
```

通过上述代码可以看出，程序通过读取 IN 寄存器的相应位来获取对应的 GPIO 的状态。IN 寄存器功能如下：

IN: 0~31 位对应于管脚 P0.00~P0.31，该寄存器的某个位=0：表示相应的管脚输入为低电平。=1：表示相应的管脚输入为高电平。

5. 开发板电路连接

本实验需要用跳线帽短接 P17~P20 管脚，如下图红框所示：

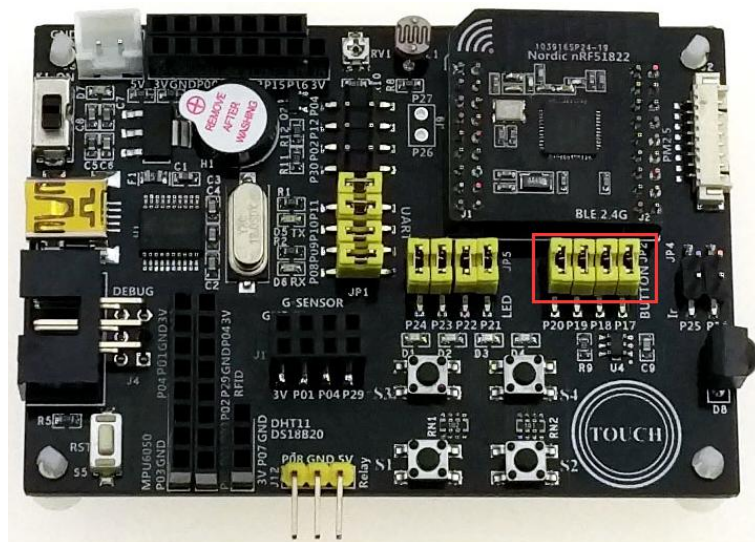



图 5：开发板跳线连接


6. 实验步骤

- 拷贝出“...\6 - 开发板应用\3 - 基础实验\实验 3 - GPIO 输入按键检测”目录下的 key 文件夹，存放到合适的目录，如“D:\NRF52832”。**强烈建议不要在资料包中直接打开工程，因为包含了中文路径且工程路径较深，可能会出现问題。**
- 启动 MDK5.18A。
- 在 MDK5 中执行“Project→Open Project”打开“...\key\project\”目录下的工程“key.uvproj”。

- 点击编译按钮编译工程 。注意查看编译输出栏，观察编译的结果，如果有错误，修改程序，直到编译成功为止。编译后生成的 HEX 文件“key.hex”位于工程目录下的“Objects”文件夹中。

```
linking...
Program Size: Code=408 RO-data=224 RW-data=4 ZI-data=2052
FromELF: creating hex file...
".\build\led.axf" - 0 Error(s), 0 Warning(s). 错误：0，警告：0表示编译通过
Build Time Elapsed: 00:00:04
```

- 点击下载按钮下载程序 。如果需要对程序进行仿真，点击 Debug 按

钮即可将程序下载到 NRF52832 进行仿真。

- 观察实验现象：运行程序，分别按下 S1~S4 按键，对应的指示灯 D1~D4 应点亮，释放按键后熄灭。

7. 实验程序

```
/*
 * 描 述 : main 函数
 * 入 参 : 无
 * 返回值 : 无
 */
int main(void)
{
    //配置 P0.17~P0.20 为输出
    nrf_gpio_range_cfg_output(LED_START, LED_STOP);
    nrf_gpio_pin_set(LED_1); //LED 初始状态为熄灭
    nrf_gpio_pin_set(LED_2);
    nrf_gpio_pin_set(LED_3);
    nrf_gpio_pin_set(LED_4);

    nrf_gpio_range_cfg_input(BUTTON_START,BUTTON_STOP,NRF_GPIO_PIN_PULLUP); //配置 P0.13~P0.16 为输入

    while (true)
    {
        //检测按键 s1 是否按下
        if(nrf_gpio_pin_read(BUTTON_1) == 0)
        {
            nrf_gpio_pin_clear(LED_1);
            while(nrf_gpio_pin_read(BUTTON_1) == 0); //等待按键释放
            nrf_gpio_pin_set(LED_1);
        }
        //检测按键 s2 是否按下
        if(nrf_gpio_pin_read(BUTTON_2) == 0)
        {
            nrf_gpio_pin_clear(LED_2);
            while(nrf_gpio_pin_read(BUTTON_2) == 0); //等待按键释放
            nrf_gpio_pin_set(LED_2);
        }
        //检测按键 s3 是否按下
        if(nrf_gpio_pin_read(BUTTON_3) == 0)
```

```
{
    nrf_gpio_pin_clear(LED_3);
    while(nrf_gpio_pin_read(BUTTON_3) == 0); //等待按键释放
    nrf_gpio_pin_set(LED_3);
}
//检测按键 S4 是否按下
if(nrf_gpio_pin_read(BUTTON_4) == 0)
{
    nrf_gpio_pin_clear(LED_4);
    while(nrf_gpio_pin_read(BUTTON_4) == 0); //等待按键释放
    nrf_gpio_pin_set(LED_4);
}
}
```