

实验4 - GPIO 输出控制蜂鸣器

1. 实验目的

掌握 NRF52832 的 GPIO 的配置方式和输出控制。
学习并掌握蜂鸣器驱动电路的设计和控制程序编写。
了解有源蜂鸣器的特性。

2. 实验内容

使用 NRF52832 的 GPIO P0.12 输出控制蜂鸣器鸣响。
程序运行后，开发板上的蜂鸣器以 200ms 的间隔鸣响。

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

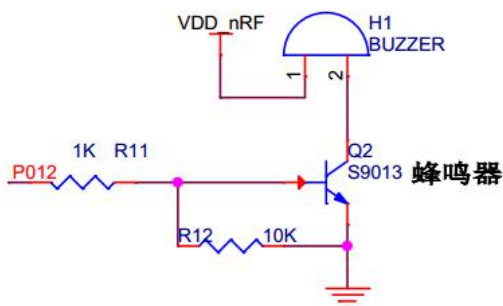


图 1：蜂鸣器驱动电路

开发板上使用的蜂鸣器是 3V 有源蜂鸣器，驱动电路如上图左半部分。当 P0.12 输出高电平时，三极管导通，蜂鸣器鸣响。当 P0.12 输出低电平时，三极管截止，蜂鸣器不鸣响。电路中的 R28 是为了保证三极管可靠的截止。

✧ **相关知识：有源蜂鸣器和无源蜂鸣器**

- 有源蜂鸣器：有源蜂鸣器内部带震荡源，所以只要一通电就会鸣响。
- 无源蜂鸣器：内部不带震荡源，用直流信号无法令其鸣响。必须用频率信号去驱动它。

4.2. GPIO 配置

本实验中，GPIO P0.12 用于驱动蜂鸣器，所以需要将 P0.12 配置为输出。之后通过操作寄存器 OUTSET 和 OUTCLR 控制蜂鸣器鸣响。

- 通过 PIN_CNF[n]寄存器配置 P0.12 为输出。
- 通过向 OUTSET 寄存器相应的位写 1 让 P0.12 输出高电平，蜂鸣器鸣响。
- 通过向 OUTCLR 寄存器相应的位写 1 让 P0.12 输出低电平，蜂鸣器停止鸣响。

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。

按照上述的描述，配置 P0.12 的程序如下：

```
nrf_gpio_cfg_output(BEEP);//配置 P0.21 为输出
```

其中 BEEP 是宏定义：#define BEEP 12

进入 nrf_gpio_cfg_output 函数，其代码如下：

```
__STATIC_INLINE void nrf_gpio_cfg_output(uint32_t pin_number)
{
    NRF_GPIO->PIN_CNF[pin_number]=(GPIO_PIN_CNF_SENSE_Disabled<< GPIO_PIN_CNF_SENSE_Pos)
                                   |(GPIO_PIN_CNF_DRIVE_S0S1 << GPIO_PIN_CNF_DRIVE_Pos)
                                   |(GPIO_PIN_CNF_PULL_Disabled << GPIO_PIN_CNF_PULL_Pos)
                                   |(GPIO_PIN_CNF_INPUT_Disconnect << GPIO_PIN_CNF_INPUT_Pos)
                                   |(GPIO_PIN_CNF_DIR_Output << GPIO_PIN_CNF_DIR_Pos);
}
```

上述代码执行后，GPIO P0.12 被配置为：

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

4.3. 驱动蜂鸣器

GPIO P0.12 配置好后, 即可通过 P0.12 输出高低电平控制蜂鸣器的鸣响, 调用下列函数即可实现 P0.12 输出高低电平。

1. `nrf_gpio_pin_set(BEEP);`

功能: 根据输入的管脚号设置该管脚输出高电平。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(BEEP);`

功能: 根据输入的管脚号设置该管脚输出低电平。进入该函数, 其代码如下:

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

和描述的一样, 该函数通过向 **OUTCLR** 寄存器相应的位写“1”来实现 GPIO 输出低电平的。

3. `nrf_gpio_pin_toggle(BEEP);`

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。

5. 开发板电路连接

本实验需要用跳线帽短接 P12 管脚, 如下图红框所示:

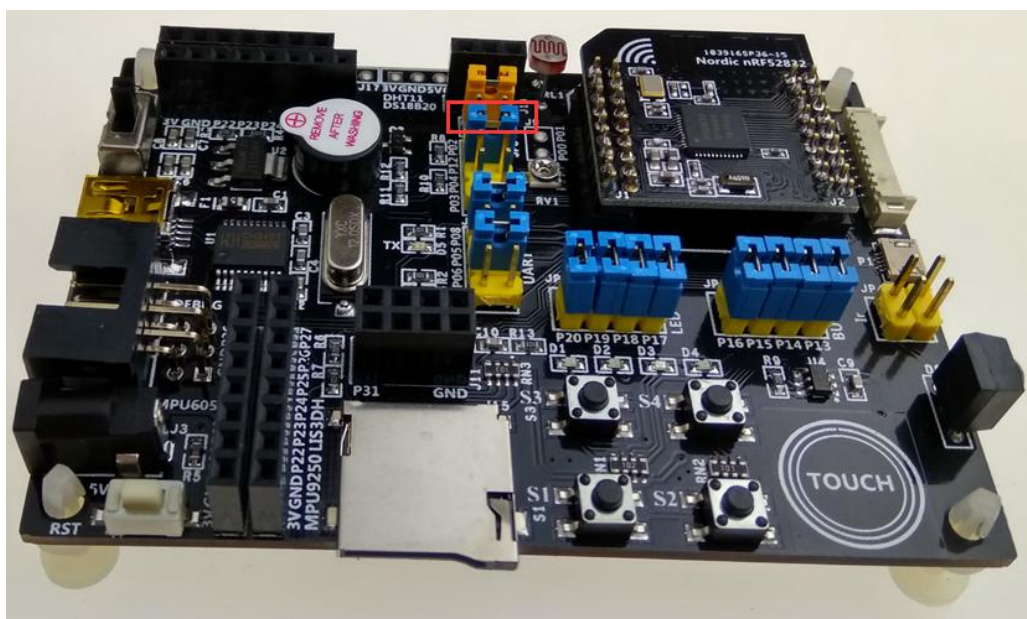



图 2: 开发板跳线连接

6. 实验步骤

- 拷贝出“...\6 - 开发板应用\3 - 基础实验\实验4 - GPIO 输出驱动蜂鸣器”目录下的 gpio_beep 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“...\gpio_beep\project\”目录下的工程“beep.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“beep.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。

7. 实验程序

```
/* *****
* 描 述 : main 函数
* 入 参 : 无
* 返回值 : 无
* ***** */
int main(void)
{
    nrf_gpio_cfg_output(BEEP); //配置 P0.12 为输出
    nrf_gpio_pin_clear(BEEP); //蜂鸣器初始状态为不鸣响

    while (true)
    {
        //蜂鸣器以 200ms 的间隔鸣响
        nrf_gpio_pin_toggle(BEEP);
        nrf_delay_ms(200);
    }
}
```