STM32Cube高效开发教程（基础篇）

第6章 GPIO输入输出

王维波
中国石油大学（华东）控制科学与工程学院

STM32Cube高效开发教程（基础篇）

作者：王维波，鄢志丹，王钊

人民邮电出版社

2021年9月出版

如果有读者需要本书课件的PPT版本用于备课，可以给作者发邮件免费获取，并可加入专门的教学和技术交流QQ群

邮箱：wangwb@upc. edu. cn

第6章 GPIO输入输出

6.1 GPIO功能概述

6.2 GPIO的HAL驱动程序

6.3 GPIO使用示例

GPIO（General Purpose Input-Output）引脚用作一般的数字输入输出引脚。其内部有双向保护二极管，有可配置是否使用的上拉和下拉电阻

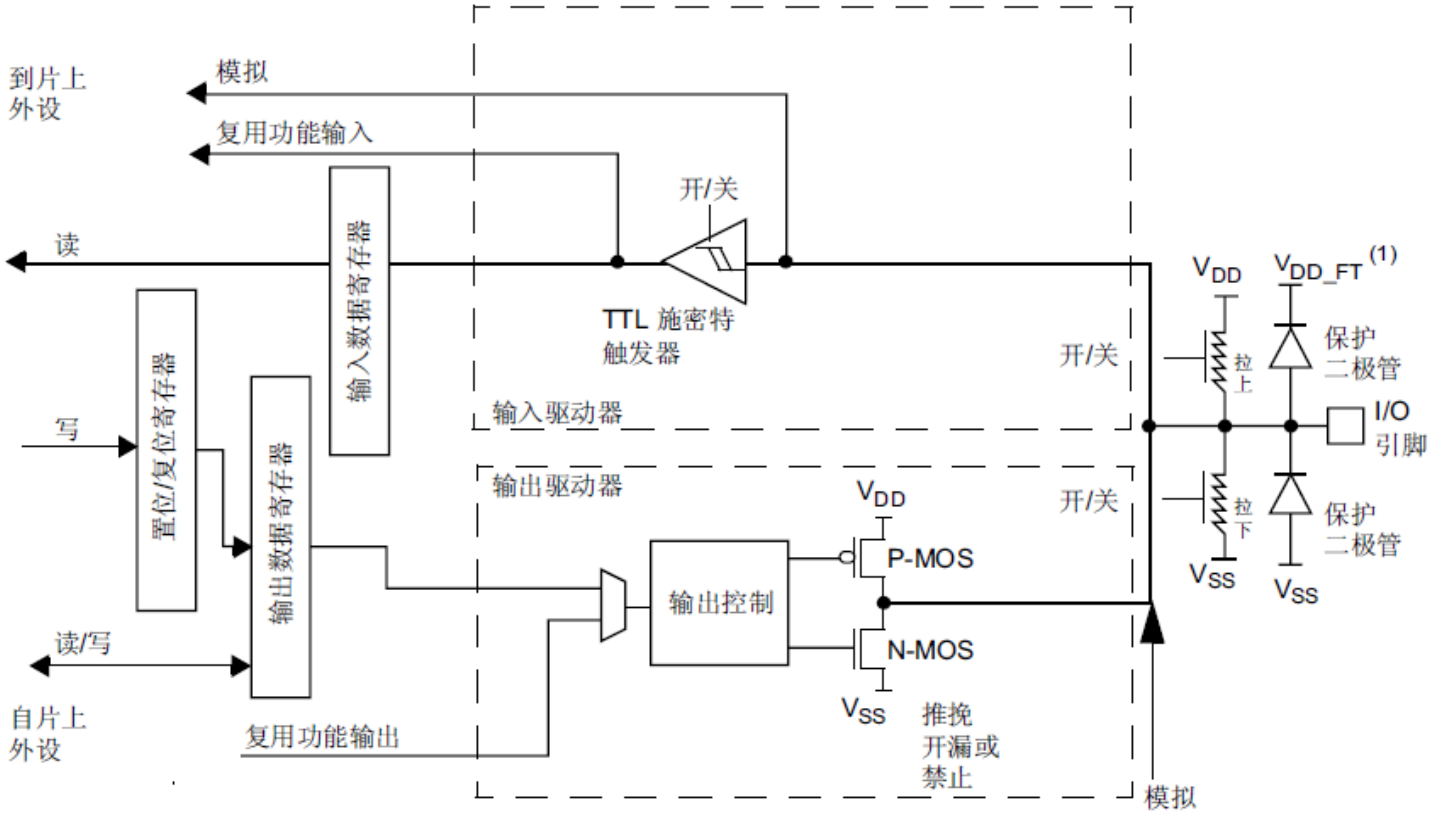


图6-1 GPIO引脚内部结构图

一个GPIO口的16个引脚的功能可以单独设置，每个引脚的输入输出数据可以单独读取或输出。

作为GPIO输入

（1）输入浮空（Input floating），并且不使用上拉或下拉。

（2）输入上拉（Input pull-up），使用内部上拉电阻，引脚外部无输入时读取的引脚输入电平为高电平。

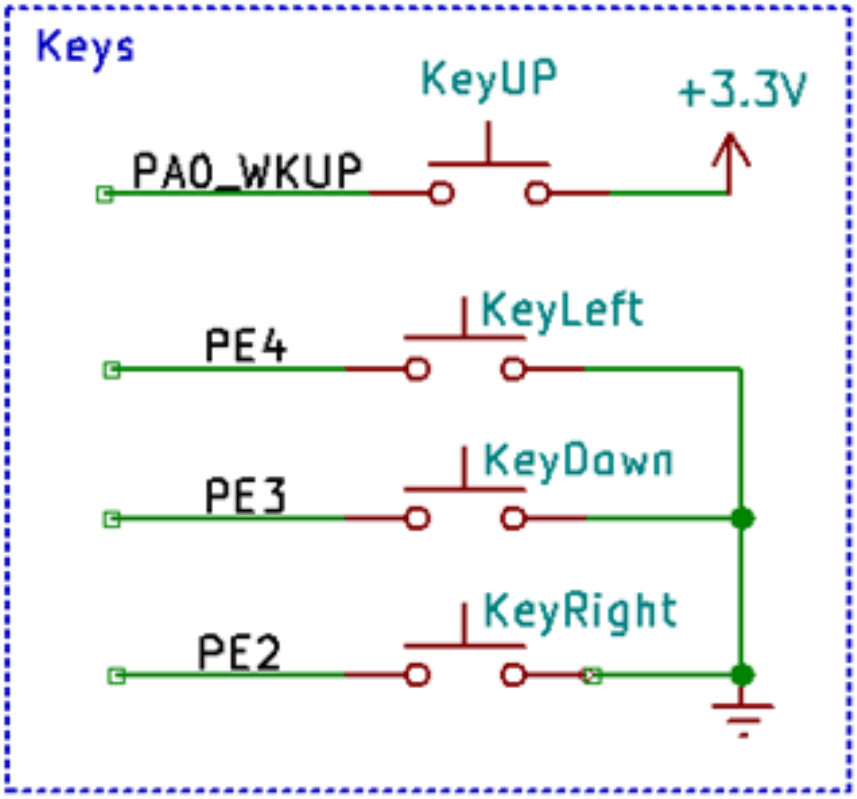
（3）输入下拉（Input pull-down），使用内部下拉电阻，引脚外部无输入时读取的引脚输入电平为低电平。

PA0：下拉输入

未按下时，PA0输入为0，按下后PA0输入为1

PE[2:4]：上拉输入

未按下时，输入为1，按下后输入为0



4个按键的电路

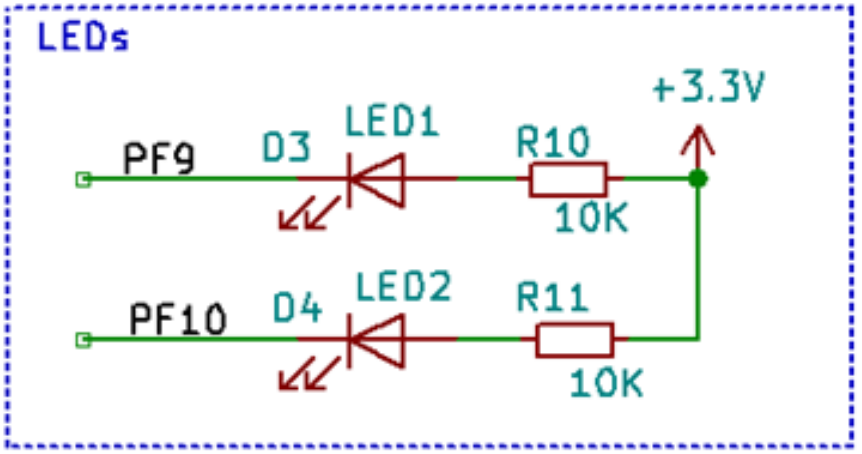
作为GPIO输出

（4）具有上拉或下拉的开漏输出（Output open-drain）。如果没有上拉或下拉，开漏输出1时引脚是高阻态，输出0时引脚是低电平，这种模式可用于共用总线的信号。

（5）具有上拉或下拉的推挽输出（Output push-pull）。如果没有上拉或下拉，推挽输出1时引脚为高电平，输出0时引脚为低电平。若需要增强引脚输出驱动能力，就可以使用上拉。

PF9，PF10：推挽输出

PF9，PF10使用开漏输出可不可以呢？



作为ADC或DAC引脚

（6）模拟（Analog 功能），作为GPIO模拟引脚，用于ADC输入或DAC输出引脚

作为复用功能引脚

（7）具有上拉或下拉的复用功能推挽（Alternate function push-pull）

（8）具有上拉或下拉的复用功能开漏（Alternate function open-drain）

第6章 GPIO输入输出

6.1 GPIO功能概述

6.2 GPIO的HAL驱动程序

6.3 GPIO使用示例

GPIO 操作相关函数

函数	功能
HAL_GPIO_Init()	GPIO引脚初始化
HAL_GPIO_DeInit()	解除引脚的初始化，恢复为复位后的状态
HAL_GPIO_WritePin()	使引脚输出0或1
HAL_GPIO_ReadPin()	读取引脚的输入电平
HAL_GPIO_TogglePin()	切换引脚的输出
HAL_GPIO_LockPin()	锁定引脚配置，而不是锁定引脚的输入或输出状态

1. 初始化函数HAL_GPIO_Init()

```
void  HAL_GPIO_Init(GPIO_TypeDef  *GPIOx, GPIO_InitTypeDef *GPIO_Init);
```

参数GPIOx是GPIO_TypeDef类型的结构体的指针，用GPIOA、GPIOB等宏定义

参数GPIO_Init是一个GPIO_InitTypeDef类型的结构体指针，它定义了引脚的属性，这个结构体的定义如下。

```
typedef struct
{
  uint32_t Pin;          //要配置的引脚，可以是多个引脚
  uint32_t Mode;         //引脚功能模式
  uint32_t Pull;         //上拉或下拉电阻
  uint32_t Speed;        //引脚工作频率
  uint32_t Alternate;    //复用功能选择
}GPIO_InitTypeDef;
```

2. 设置引脚输出的函数HAL_GPIO_WritePin()

```
void HAL_GPIO_WritePin(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin,
GPIO_PinState PinState);
```

PinState是引脚输出电平，这是枚举类型GPIO_PinState

```
typedef enum
{
    GPIO_PIN_RESET = 0,
    GPIO_PIN_SET
}GPIO_PinState;
```

3. 读取引脚输入的函数HAL_GPIO_ReadPin()

```
GPIO_PinState HAL_GPIO_ReadPin(GPIO_TypeDef* GPIOx,
                               uint16_t GPIO_Pin);
```

函数的返回值是枚举类型GPIO_PinState。

- 常量GPIO_PIN_RESET表示输入为0（低电平）
- 常量GPIO_PIN_SET表示输入为1（高电平）

4. 切换引脚输出状态的函数HAL_GPIO_TogglePin()

```
void HAL_GPIO_TogglePin(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin)
```

函数HAL_GPIO_TogglePin()用于切换引脚的输出状态，例如当前引脚输出为高电平，执行此函数后就使引脚输出为低电平。

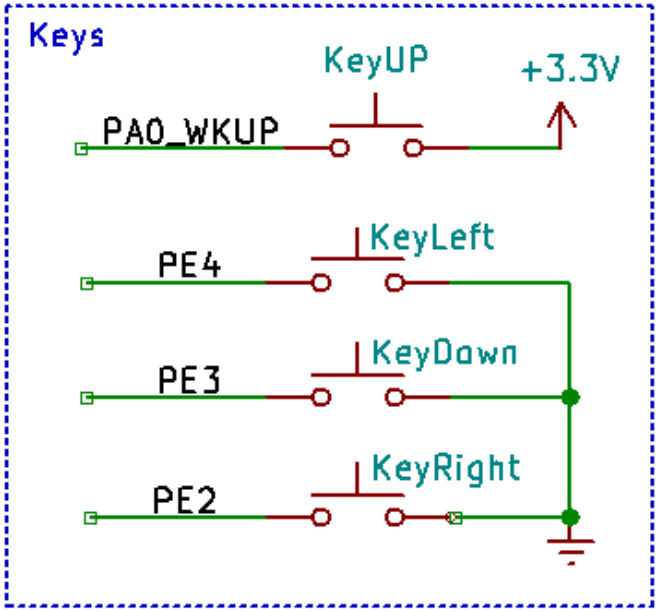
6.3 GPIO使用示例

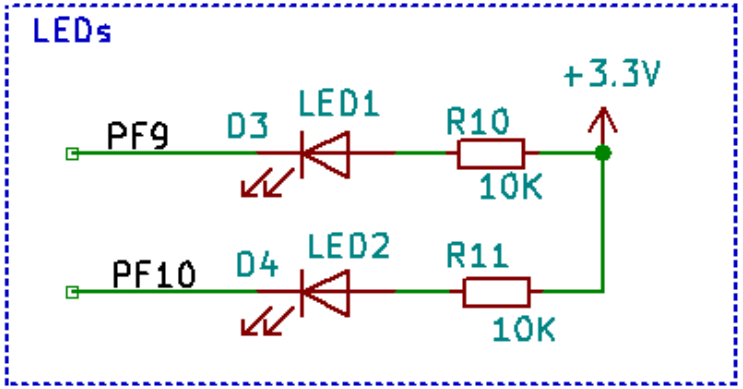
6.3.1 示例功能和CubeMX配置

6.3.2 项目初始化代码分析

6.3.3 添加用户功能代码

6.3.1 示例功能和STM32CubeMX配置



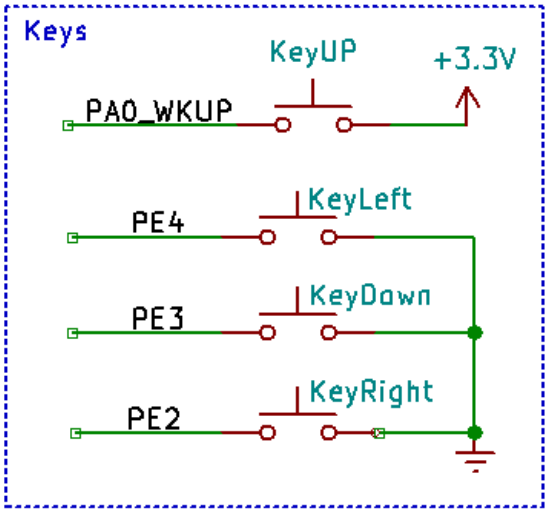


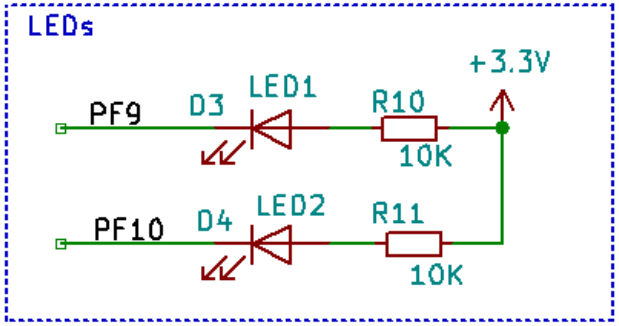
示例Demo6_1KeyLED功能：

（1）按KeyLeft，使LED1输出翻转

（2）按KeyRight，使LED2输出翻转

（3）按下KeyUp键时使LED1和LED2的输出都翻转





按键、LED引脚的GPIO配置

用户标签	MCU引脚号	引脚名称	引脚功能	GPIO模式	上拉或下拉
LED1	Pin21	PF9	GPIO_Output	推挽输出	无
LED2	Pin22	PF10	GPIO_Output	推挽输出	无
KeyRight	Pin01	PE2	GPIO_Input	输入	上拉
KeyDown	Pin02	PE3	GPIO_Input	输入	上拉
KeyLeft	Pin03	PE4	GPIO_Input	输入	上拉
KeyUP	Pin34	PA0	GPIO_Input	输入	下拉

STM32CubeMX的配置过程演示

GPIO引脚配置结果

Pin Name	User Label	GPIO mode	GPIO Pull-up/Pull-down	GPIO output...	Maximum...	Sign...	Mo...
PF8	Buzzer	Output Push Pull	No pull-up and no pull-down	High	High	n/a	☑
PF9	LED1	Output Push Pull	No pull-up and no pull-down	Low	Medium	n/a	☑
PF10	LED2	Output Push Pull	No pull-up and no pull-down	Low	Medium	n/a	☑
PA0-WKUP	KeyUp	Input mode	Pull-down	n/a	n/a	n/a	☑
PE3	KeyDown	Input mode	Pull-up	n/a	n/a	n/a	☑
PE4	KeyLeft	Input mode	Pull-up	n/a	n/a	n/a	☑
PE2	KeyRight	Input mode	Pull-up	n/a	n/a	n/a	☑

PE3 Configuration :

GPIO mode: Input mode

GPIO Pull-up/Pull-down: Pull-up

User Label: KeyDown

6.3.2 项目初始化代码分析

1. 主程序

```
#include "main.h"
#include "gpio.h"
/* Private function prototypes -----------------------------*/
void SystemClock_Config(void);

int main(void)
{
  HAL_Init();         // 复位所有外设，初始化Flash接口和Systick
  SystemClock_Config();    // 配置系统时钟

/* 所有已配置外设的初始化 */
  MX_GPIO_Init();              // GPIO引脚初始化
  while (1)
  {

  }
}
```

2. 文件main.h中的引脚用户标签定义

```
/* Private defines -----------------------------------------------------------*/
#define KeyRight_Pin            GPIO_PIN_2
#define KeyRight_GPIO_Port      GPIOE

#define KeyDown_Pin             GPIO_PIN_3
#define KeyDown_GPIO_Port       GPIOE

#define KeyLeft_Pin             GPIO_PIN_4
#define KeyLeft_GPIO_Port       GPIOE

#define KeyUp_Pin               GPIO_PIN_0
#define KeyUp_GPIO_Port         GPIOA

#define LED1_Pin                GPIO_PIN_9      //PF9=LED1的引脚号Pin9
#define LED1_GPIO_Port          GPIOF           //PF9=LED1的端口PF

#define LED2_Pin                GPIO_PIN_10
#define LED2_GPIO_Port          GPIOF
```

3. GPIO引脚初始化

文件gpio.c中的函数MX_GPIO_Init()的代码分析

(1) 使能端口时钟

```
/* GPIO 端口时钟使能 */
 __HAL_RCC_GPIOE_CLK_ENABLE();
 __HAL_RCC_GPIOC_CLK_ENABLE();
 __HAL_RCC_GPIOF_CLK_ENABLE();
 __HAL_RCC_GPIOH_CLK_ENABLE();
 __HAL_RCC_GPIOA_CLK_ENABLE();
 __HAL_RCC_GPIOB_CLK_ENABLE();
```

(2) 设置LED两个引脚初始输出电平

```
/* 配置GPIO引脚输出电平*/
 HAL_GPIO_WritePin(GPIOF, GPIO_PIN_9|GPIO_PIN_10, GPIO_PIN_RESET);
```

(3) 配置PE2,PE3,PE4三个输入上拉的按键

```
/*配置GPIO引脚: PE2 PE3 PE4， 三个输入上拉的按键*/
 GPIO_InitStruct.Pin = GPIO_PIN_2|GPIO_PIN_3|GPIO_PIN_4;
 GPIO_InitStruct.Mode = GPIO_MODE_INPUT;   //GPIO输入模式
 GPIO_InitStruct.Pull = GPIO_PULLUP;              //输入上拉
 HAL_GPIO_Init(GPIOE, &GPIO_InitStruct);
```

(4) 配置两个推挽输出的LED引脚

```
/*配置GPIO引脚: PF9 PF10， 两个推挽输出的LED*/
GPIO_InitStruct.Pin = GPIO_PIN_9|GPIO_PIN_10;
GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;     //推挽输出模式
GPIO_InitStruct.Pull = GPIO_NOPULL;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
HAL_GPIO_Init(GPIOF, &GPIO_InitStruct);
```

(5) 配置PA0，输入下拉按键

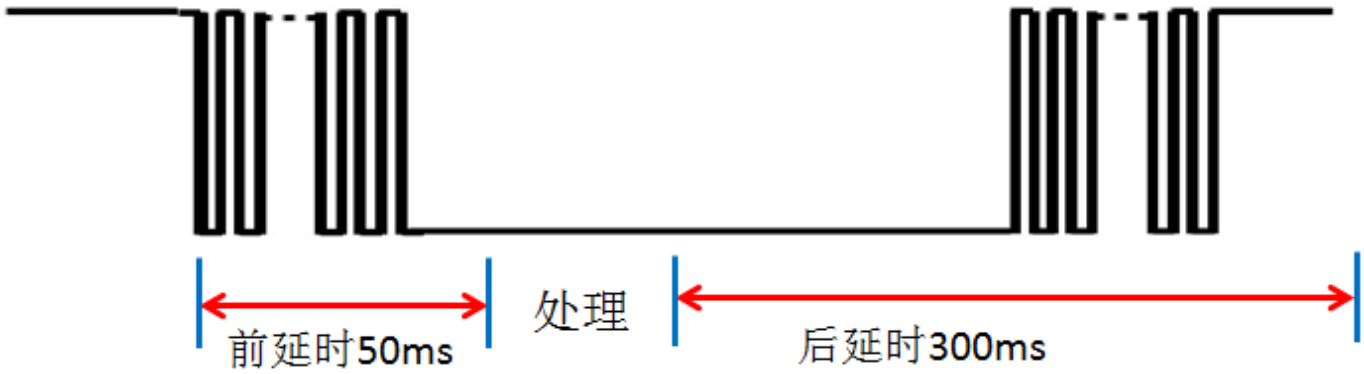
```
/*配置GPIO引脚: PA0 ,一个输入下拉的按键*/
 GPIO_InitStruct.Pin = GPIO_PIN_0;
 GPIO_InitStruct.Mode = GPIO_MODE_INPUT;  //GPIO输入模式
 GPIO_InitStruct.Pull = GPIO_PULLDOWN;        //输入下拉
 HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
```

使用函数HAL_GPIO_Init()进行GPIO引脚配置，可以一次配置一个端口的多个同类型引脚，但是不同类型的引脚需要分开配置

6.3.3 编写按键和LED的驱动程序

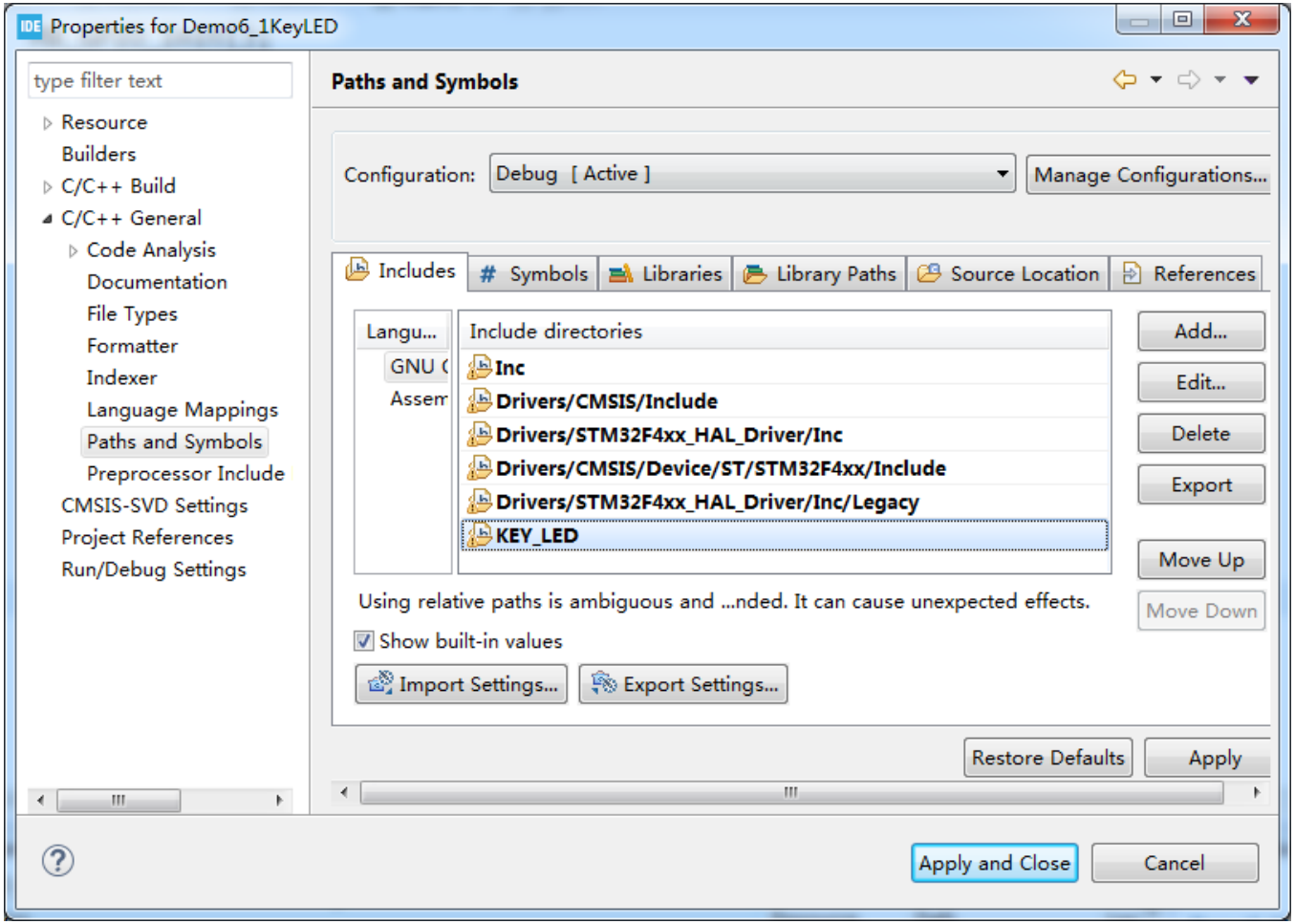
- 创建子目录 KEY_LED
- 创建文件keyled.h和keyled.c
- 头文件keyled.h代码分析【看源代码】
- 文件keyled.c的代码分析【看源代码】

按键抖动的问题



6.3.4 使用驱动程序实现示例功能

- 添加头文件和源程序搜索路径
- 代码沙箱（Sand box）



代码沙箱

```
/* Private includes ----------------------------------------*/
/* USER CODE BEGIN Includes */

/* USER CODE END Includes */
```

在使用STM32CubeMX再次生成代码时，在沙箱内填写的用户代码不会被覆盖，而如果写在其他地方，代码会丢失。

本书在显示代码时，会将添加了用户代码的沙箱段的BEGIN和END两个注释语句用粗体显示。

练习任务

1. 开发板上的蜂鸣器电路如下图所示，信号BEEP连接MCU的PF8引脚，PF8输出低电平时蜂鸣器发声，PF8输出高电平时蜂鸣器不发声。创建一个CubeMX项目对硬件进行配置，然后在CubeIDE中编程对蜂鸣器进行控制，使蜂鸣器发声。