第3讲

GPIO：输入

主要内容

- GPIO寄存器
- GPIO输入：按键
- GPIO: 按键、蜂鸣器
- 动手练习3

main程序结构

- HAL初始化
- 时钟初始化
- **外设初始化：GPIO、串口…**
- 用户代码
- ……

GPIO寄存器

GPIO寄存器

■GPIO寄存器，11个，32-bit

- ✓ 配置寄存器(configuration registers)：

  GPIOx_MODER,GPIOx_OTYPER, GPIOx_OSPEEDR, GPIOx_PUPDR

- ✓ 数据寄存器(data registers):

  GPIOx_IDR, GPIOx_ODR

- ✓ 按位置位/复位寄存器(bit set/reset register)：GPIOx_BSRR
- ✓ 按位复位寄存器(**bit reset register): GPIOx_BRR**
- ✓ 锁存寄存器(locking register)：GPIOx_LCKR
- ✓ alternate function selection registers：GPIOx_AFRH，GPIOx_AFRL

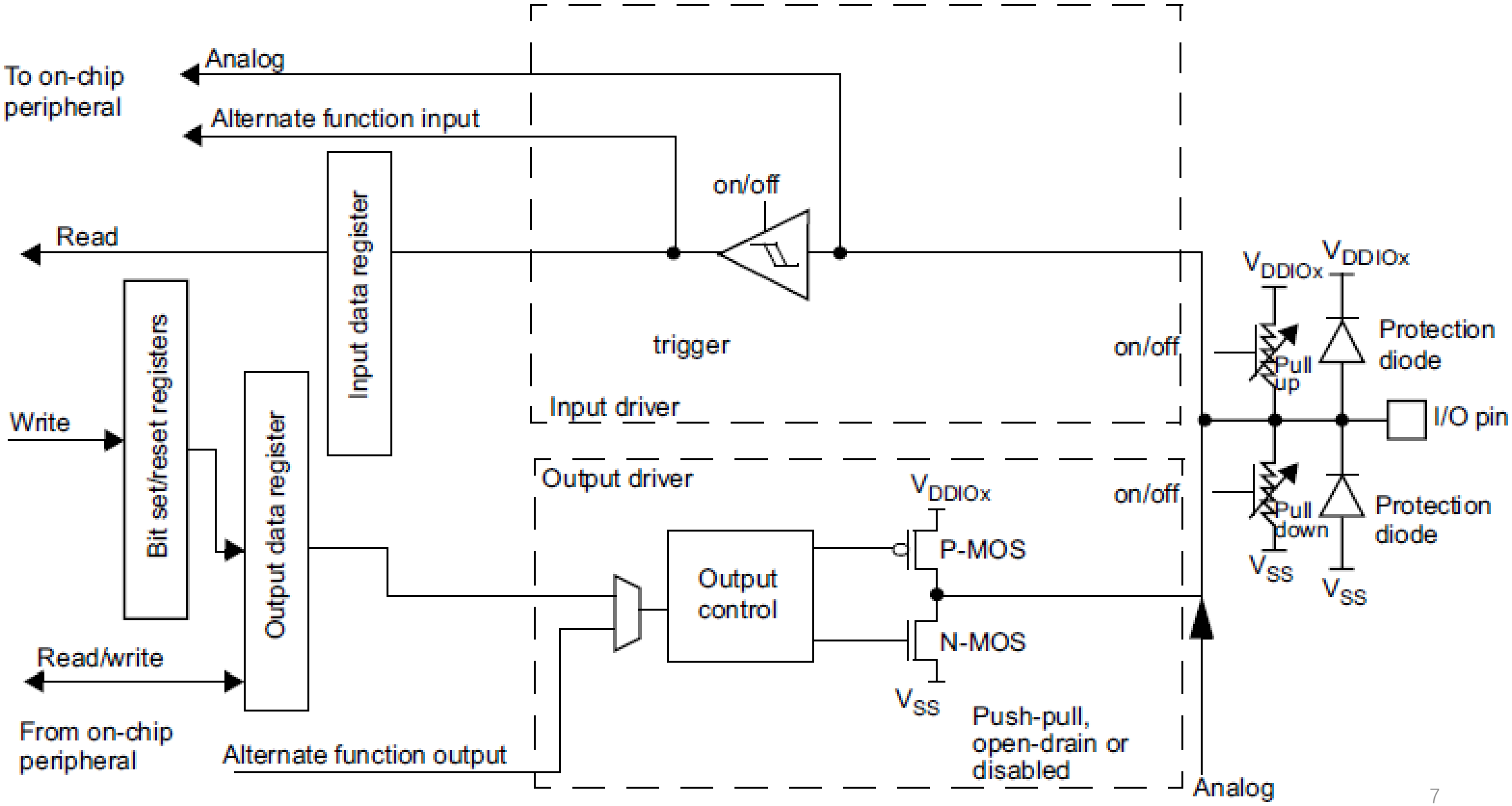
GPIO寄存器

- **GPIO端口可以由软件分别配置成多种模式:**
  - ✓ 输入：浮空，上拉/下拉，模拟

    floating, pull-up/down, analog
  - ✓ 输出：推挽，开漏；+上拉/下拉

    push-pull or open drain + pull-up/down

I/O端口的基本结构


To on-chip peripheral
Analog
Alternate function input
on/off
Read
Input data register
trigger
Input driver
on/off
V_{DDIOx}
V_{DDIOx}
Protection diode
Pull up
I/O pin
Write
Bit set/reset registers
Output data register
Output driver
V_{DDIOx}
on/off
P-MOS
Output control
Pull down
Protection diode
V_{SS}
V_{SS}
N-MOS
Read/write
From on-chip peripheral
Alternate function output
V_{SS}
Push-pull, open-drain or disabled
Analog

初始化外设，如何编程?

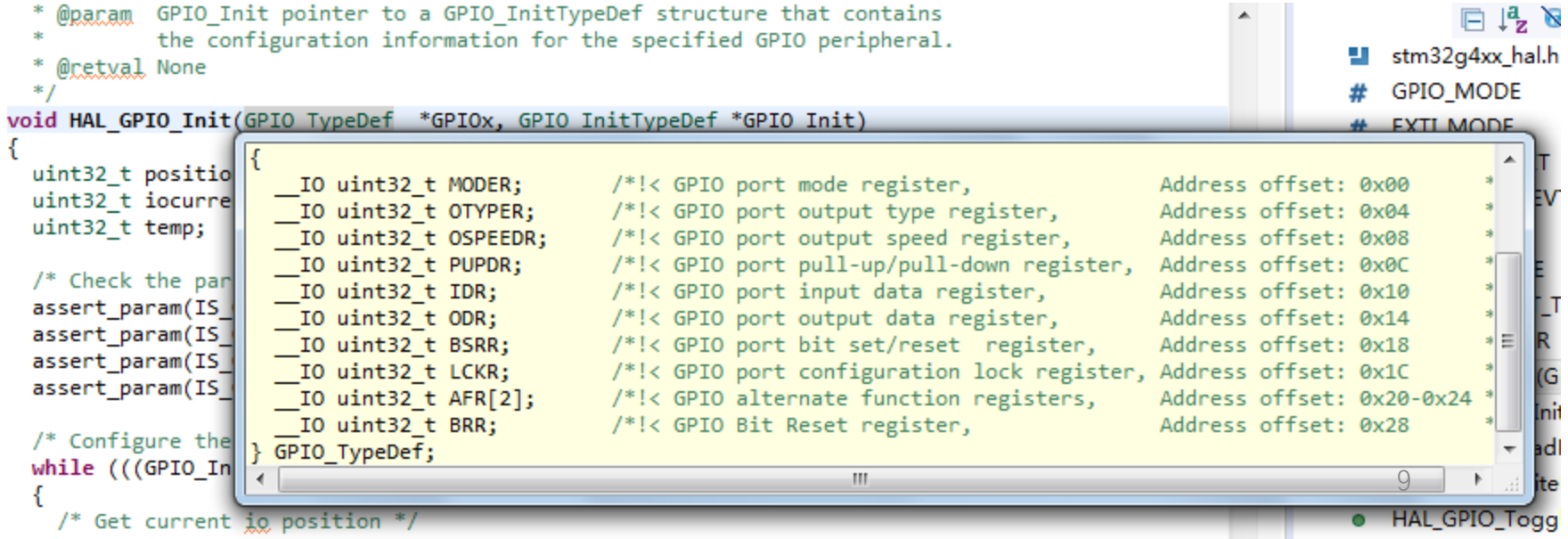
- 初始化外设，用外设初始化函数，在stm32g4xx_hal_ppp.c/.h中.
- 以GPIO为例，stm32g4xx_hal_gpio.c 中，GPIO初始化函数为：
- void HAL_GPIO_Init(GPIO_TypeDef *GPIOx, GPIO_InitTypeDef *GPIO_Init)
- 在函数定义前面的注释中，有对参数的说明

✓ param GPIOx: where x can be (A..G depending on device used) to select the GPIO peripheral

✓ param GPIO_Init: pointer to a GPIO_InitTypeDef structure that contains the configuration information for the specified GPIO peripheral.

初始化外设，如何编程？

- void HAL_GPIO_Init(GPIO_TypeDef *GPIOx, GPIO_InitTypeDef *GPIO_Init)
- 两个入口参数. 不参考其他代码时，如何查找参数类型等信息？
- 鼠标移至GPIO_TypeDef…，或右键，选择“Open Declaration”



stm32g4xx_hal_gpio.h

```
typedef struct
{
  uint32_t Pin;   /*!< Specifies the GPIO pins to be configured.
                       This parameter can be any value of @ref GPIO_pins */

  uint32_t Mode;  /*!< Specifies the operating mode for the selected pins.
                       This parameter can be a value of @ref GPIO_mode */

  uint32_t Pull;  /*!< Specifies the Pull-up or Pull-Down activation for the selected pins.
                       This parameter can be a value of @ref GPIO_pull */

  uint32_t Speed; /*!< Specifies the speed for the selected pins.
                       This parameter can be a value of @ref GPIO_speed */

  uint32_t Alternate;  /*!< Peripheral to be connected to the selected pins
                       This parameter can be a value of @ref GPIOEx_Alternate_function_selection */

} GPIO_InitTypeDef;
```

stm32f1xx_hal_gpio.h

```
/* @defgroup GPIO_pins GPIO pins */
#define GPIO_PIN_0                 ((uint16_t)0x0001)  /* Pin 0 selected    */
#define GPIO_PIN_1                 ((uint16_t)0x0002)  /* Pin 1 selected    */
……
/* @defgroup GPIO_mode GPIO mode */
#define GPIO_MODE_INPUT 0x00000000U /* Input Floating Mode */
#define GPIO_MODE_OUTPUT_PP 0x00000001U/*Output Push Pull Mode*/
……
/* @defgroup GPIO_speed GPIO speed */
#define  GPIO_SPEED_FREQ_LOW         (0x00000000U)
#define  GPIO_SPEED_FREQ_MEDIUM      (0x00000001U)
#define  GPIO_SPEED_FREQ_HIGH        (0x00000002U)
#define  GPIO_SPEED_FREQ_VERY_HIGH   (0x00000003U)
 /* @defgroup GPIO_pull GPIO pull */
#define  GPIO_NOPULL        0x00000000U
#define  GPIO_PULLUP        0x00000001U
#define  GPIO_PULLDOWN      0x00000002U
```

GPIO speed

低速：GPIO_SPEED_FREQ_LOW，最高到5MHz

中速：GPIO_SPEED_FREQ_MEDIUM，5~25MHz

高速：GPIO_SPEED_FREQ_HIGH，25~50MHz

很高速：GPIO_SPEED_FREQ_VERY_HIGH，50~120MHz

void HAL_GPIO_Init(GPIO_TypeDef *GPIOx, GPIO_InitTypeDef *GPIO_Init)

stm32g4xx_hal_gpio.h

```
typedef struct
{
    uint32_t Pin;
    uint32_t Mode;
    uint32_t Pull;
    uint32_t Speed;
    uint32_t Alternate
} GPIO_InitTypeDef;
```

```
GPIO_InitTypeDef  GPIO_Initure;
GPIO_Initure.Pin = GPIO_PIN_0|GPIO_PIN_1;
GPIO_Initure.Mode = GPIO_MODE_OUTPUT_PP;
GPIO_Initure.Pull = GPIO_PULLUP;
GPIO_Initure.Speed = GPIO_SPEED_HIGH;
HAL_GPIO_Init ( GPIOA, & GPIO_Initure );
```

stm32g431xx.h

```
#define GPIOA   ((GPIO_TypeDef *)GPIOA_BASE)
```

ODR寄存器

- 设置IO输出高(ODRy=1)/低电平(ODRy=0)
- 输出模式下有效，输入模式下不起作用.
- 可通过操作ODR寄存器，改变IO端口输出电平

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OD15	OD14	OD13	OD12	OD11	OD10	OD9	OD8	OD7	OD6	OD5	OD4	OD3	OD2	OD1	OD0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

ODRy[15:0]：端口输出数据(y = 0…15) 这些位可读可写并只能以字(16位)的形式操作

BSSR寄存器: 置位/复位寄存器

- 与ODR有类似作用, 可设置IO端口输出高低电平.
- 低16位(0~15)置位, 写1, 对应端口输出高; 写0, 对IO口无影响.
- 高16位复位, 写1, 对应端口输出低; 写0, 对IO口无影响.
- ODR: 先读ODR的值, 然后对整个ODR重新赋值.
- BSRR: 不需要先读, 直接设置.

```
GPIOA->ODR |= 1 << 5;//置位PA5,不改变其他位
GPIOA->BSRR = 1 << 5; ;//PA5输出1
GPIOA->BSRR = 1 << 21;;//PA5输出0
```

BSSR寄存器: 置位/复位寄存器

- GPIOA->BSRR = 1 << 5;

```
0800053e:   mov.w   r3, #1207959552 ; 0x48000000
08000542:   movs    r2, #32
08000544:   str     r2, [r3, #24]
```

此处#24指什么？

- 库函数操作BSSR寄存器的函数为:

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

```
HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_SET);
HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_RESET);
```

IDR寄存器

- 读GPIO端口上的电平值
- HAL库中的函数为：

  GPIO_PinState HAL_GPIO_ReadPin(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin)，如函数：

  HAL_GPIO_ReadPin(GPIOC, GPIO_Pin_0); //该函数的返回值就是PC0端口上的电平值
- 此函数通过读取IDR寄存器，得到IO端口电平

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

IDRy[15:0]：端口输入数据(y = 0…15) 这些位只读并只能以字(16位)的形式读出

GPIO操作步骤

- 使能IO时钟. 调用函数: __HAL_RCC_GPIOx_CLK_ENABLE();
- 初始化IO, 调用函数: HAL_GPIO_Init();
- IO读写

✓ HAL_GPIO_ReadPin(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin)

HAL_GPIO_ReadPin(GPIOC, GPIO_Pin_0); //该函数的返回值就是PC0端口上的电平值

✓ HAL_GPIO_WritePin(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin, GPIO_PinState PinState)

HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, GPIO_PIN_SET);//PA5输出高电平

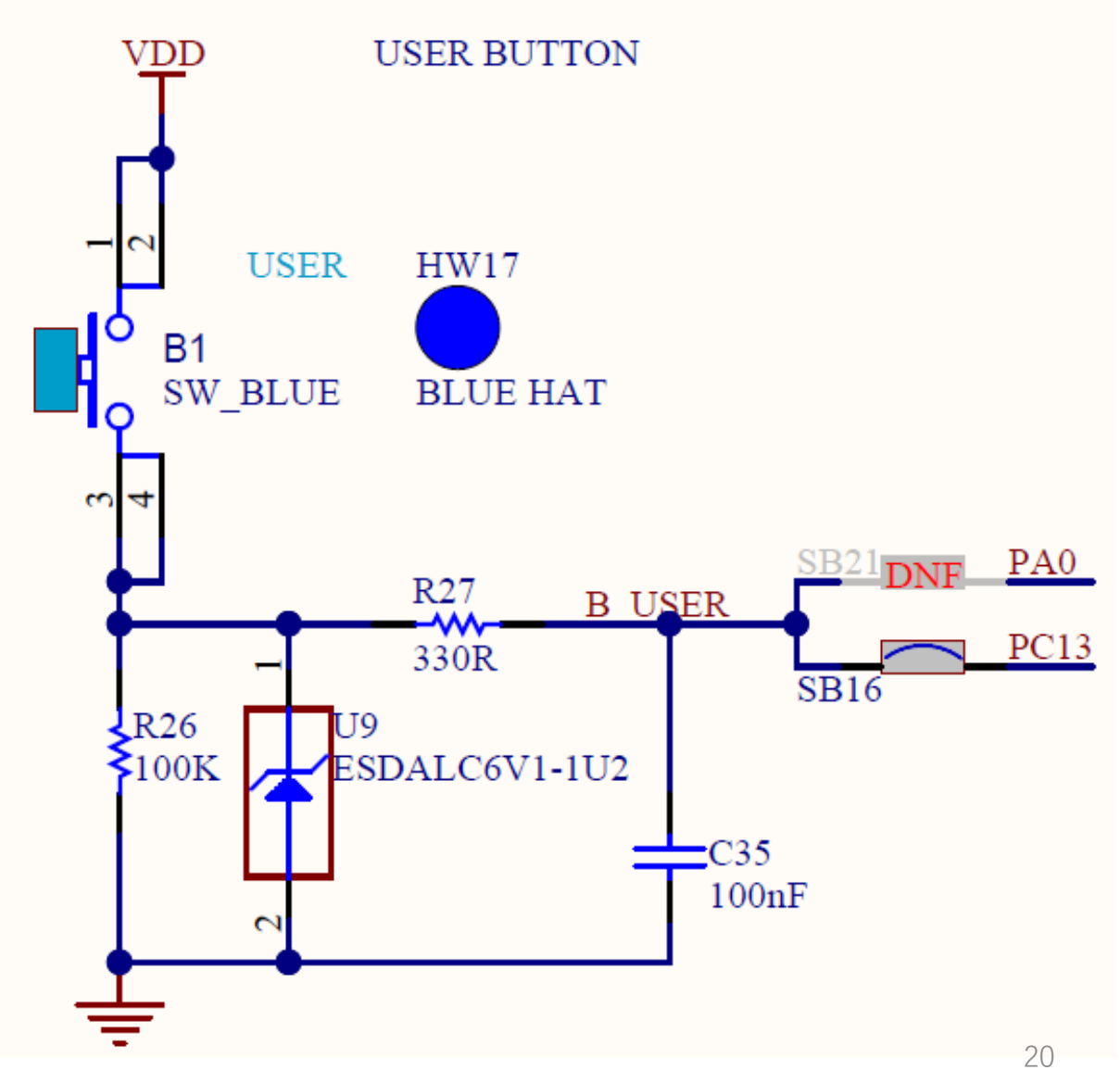
GPIO输入控制

练习3：按键输入

- 完成功能：
  - ✓ 用B1控制LD2的亮灭
  - ✓ NUCLEO-G431RB板：

    PA5 -> LD2

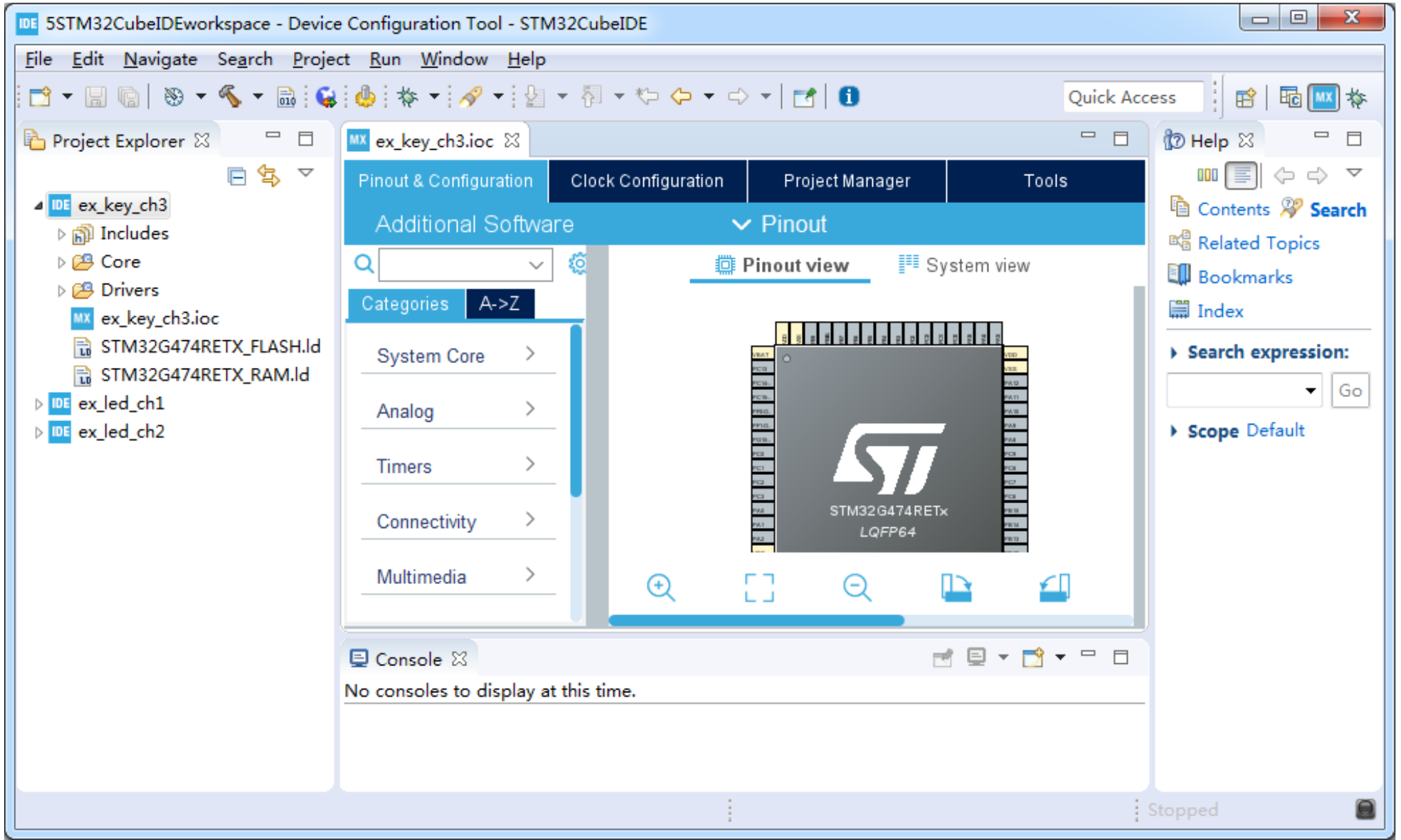
    PC13 <- B1



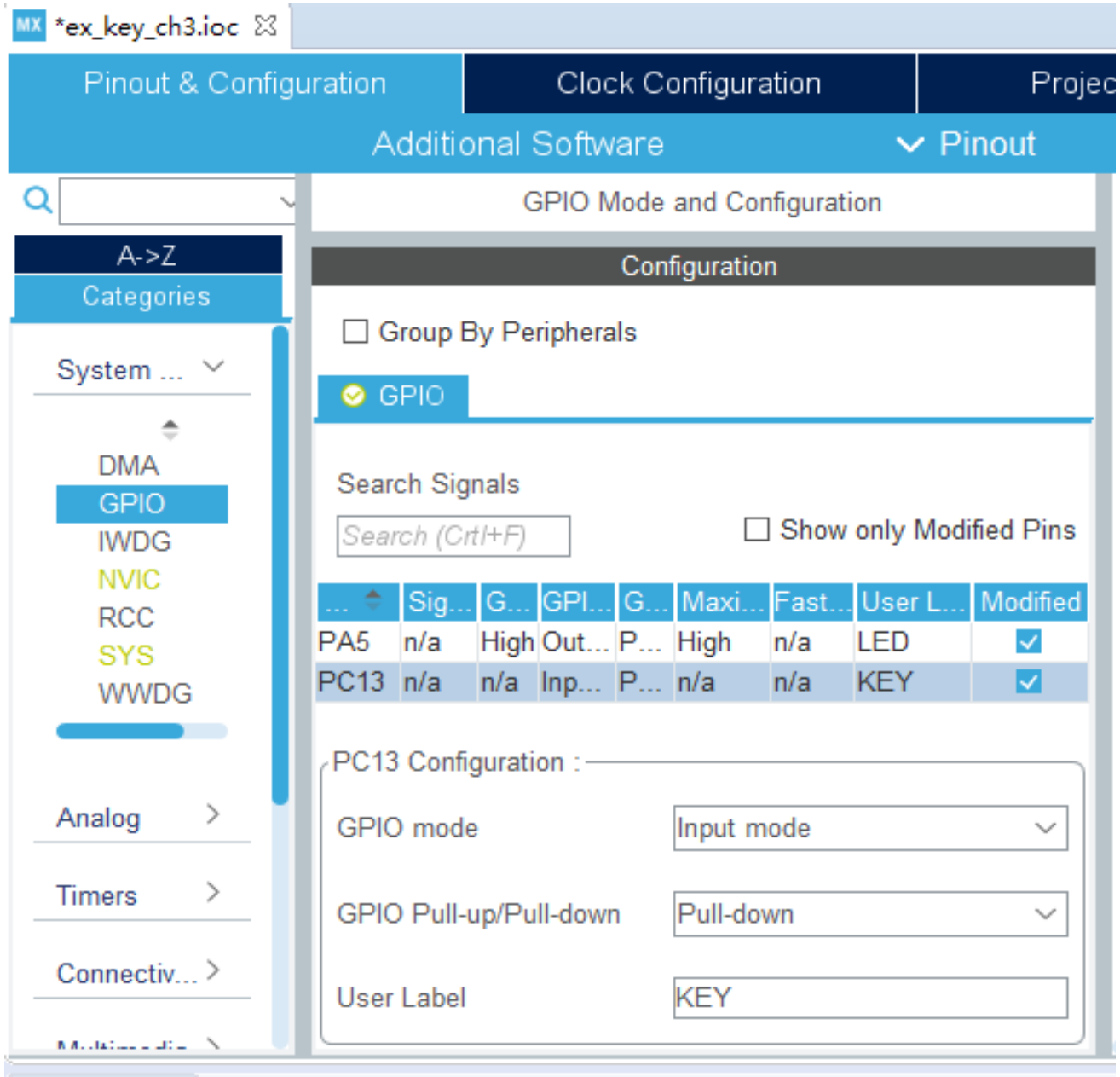
NUCLEO64 STM32G4

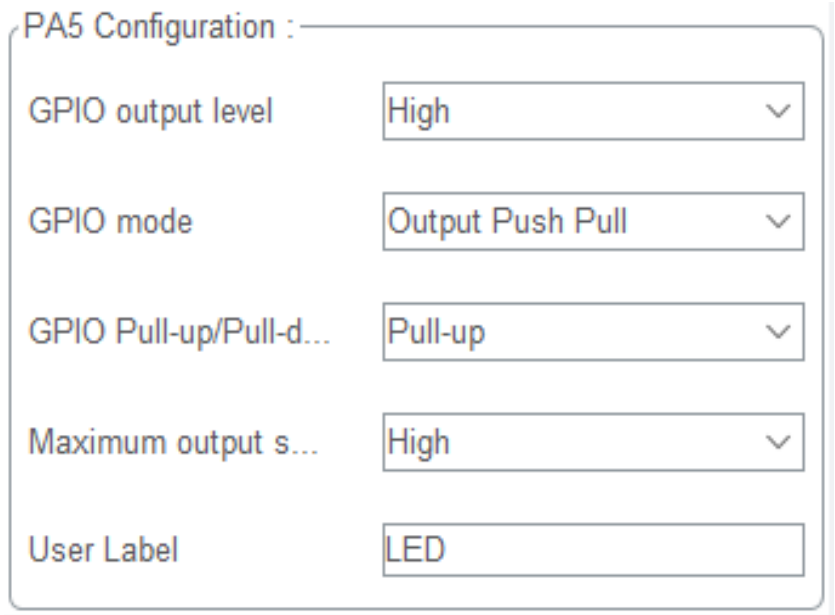
en.mb1367-g431rb-c04_schematic.pdf

建立新工程


5STM32CubeIDEworkspace - Device Configuration Tool - STM32CubeIDE
File Edit Navigate Search Project Run Window Help
Quick Access
Project Explorer
ex_key_ch3
Includes
Core
Drivers
ex_key_ch3.ioc
STM32G474RETX_FLASH.ld
STM32G474RETX_RAM.ld
ex_led_ch1
ex_led_ch2
ex_key_ch3.ioc
Pinout & Configuration
Clock Configuration
Project Manager
Tools
Additional Software
Pinout
Categories
A->Z
System Core
Analog
Timers
Connectivity
Multimedia
Pinout view
System view
STM32G474RETx
LQFP64
Help
Contents
Search
Related Topics
Bookmarks
Index
Search expression:
Go
Scope Default
Console
No consoles to display at this time.
Stopped

配置GPIO


*ex_key_ch3.ioc
Pinout & Configuration
Clock Configuration
Projec
Additional Software
Pinout
A->Z
Categories
System ...
DMA
GPIO
IWDG
NVIC
RCC
SYS
WWDG
Analog
Timers
Connectiv...
GPIO Mode and Configuration
Configuration
Group By Peripherals
GPIO
Search Signals
Search (Crtl+F)
Show only Modified Pins
... Sig... G... GPI... G... Maxi... Fast... User L... Modified
PA5 n/a High Out... P... High n/a LED
PC13 n/a n/a Inp... P... n/a n/a KEY
PC13 Configuration :
GPIO mode
Input mode
GPIO Pull-up/Pull-down
Pull-down
User Label
KEY


PA5 Configuration :
GPIO output level
High
GPIO mode
Output Push Pull
GPIO Pull-up/Pull-d...
Pull-up
Maximum output s...
High
User Label
LED

其他硬件参数配置

■选择时钟源和Debug模式

✓ System Core->RC->将高速时钟（HSE）选择为Crystal/Ceramic Resonator

✓ SYS->Debug选择为Serial Wire

■配置系统时钟

✓ 在“Clock Configuration”中，将系统时钟（SYSCLK）配置为170Mhz

■ 保存硬件配置界面（*.ioc），启动代码生成

GPIO初始化函数(1)

```
static void MX_GPIO_Init(void)
{
    GPIO_InitTypeDef GPIO_InitStruct = {0};
    /* GPIO Ports Clock Enable */
      __HAL_RCC_GPIOC_CLK_ENABLE();
      __HAL_RCC_GPIOF_CLK_ENABLE();
      __HAL_RCC_GPIOA_CLK_ENABLE();
      /*Configure GPIO pin Output Level */
      HAL_GPIO_WritePin(LED_GPIO_Port, LED_Pin, GPIO_PIN_RESET);
      /*Configure GPIO pin : KEY_Pin */
      GPIO_InitStruct.Pin = KEY_Pin;
      GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
      GPIO_InitStruct.Pull = GPIO_PULLDOWN;
      HAL_GPIO_Init(KEY_GPIO_Port, &GPIO_InitStruct);
    ……
}
```

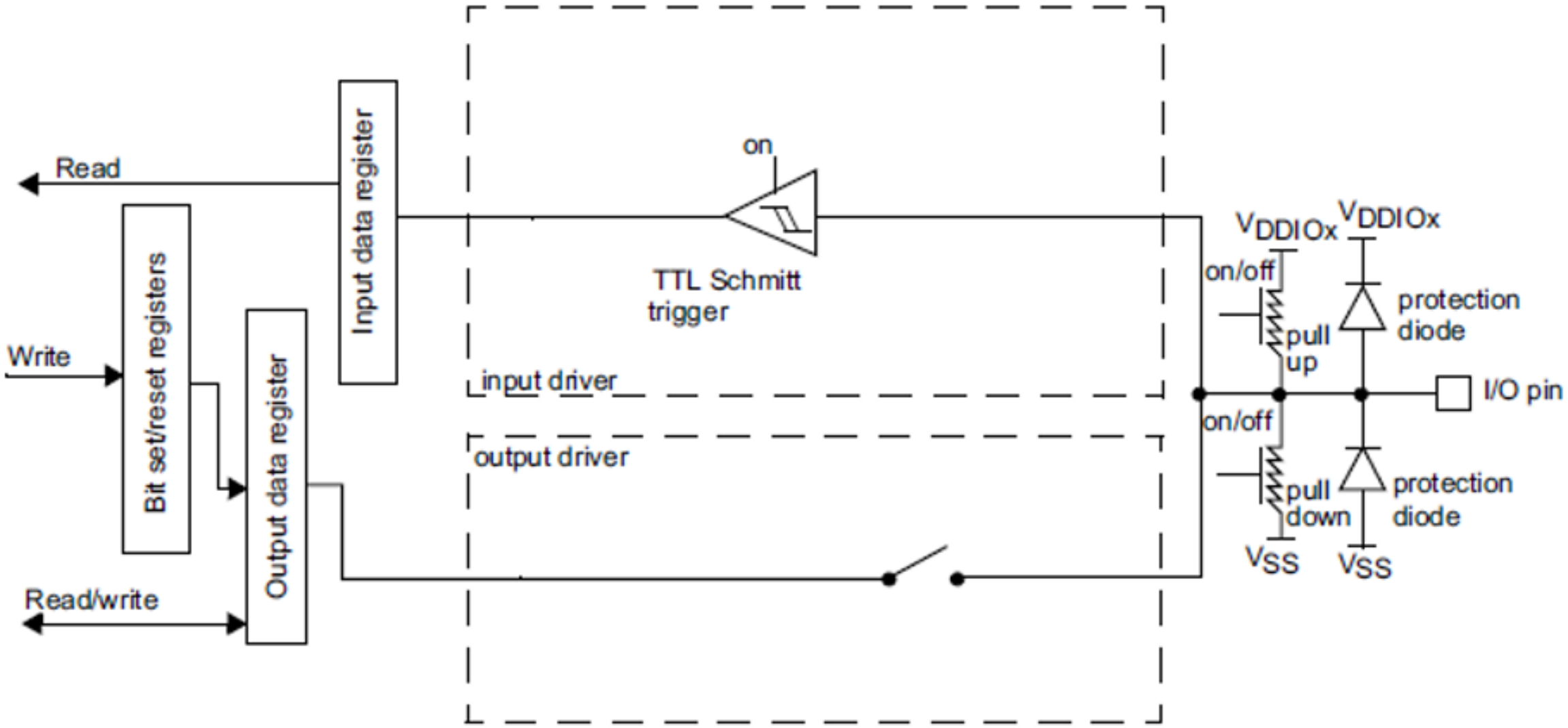
GPIO初始化函数(2)

```
static void MX_GPIO_Init(void)
{
      ……
      /*Configure GPIO pin : LED_Pin */
      GPIO_InitStruct.Pin = LED_Pin;
       GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
       GPIO_InitStruct.Pull = GPIO_PULLUP;
      GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_HIGH;
       HAL_GPIO_Init(LED_GPIO_Port, &GPIO_InitStruct);
}
```

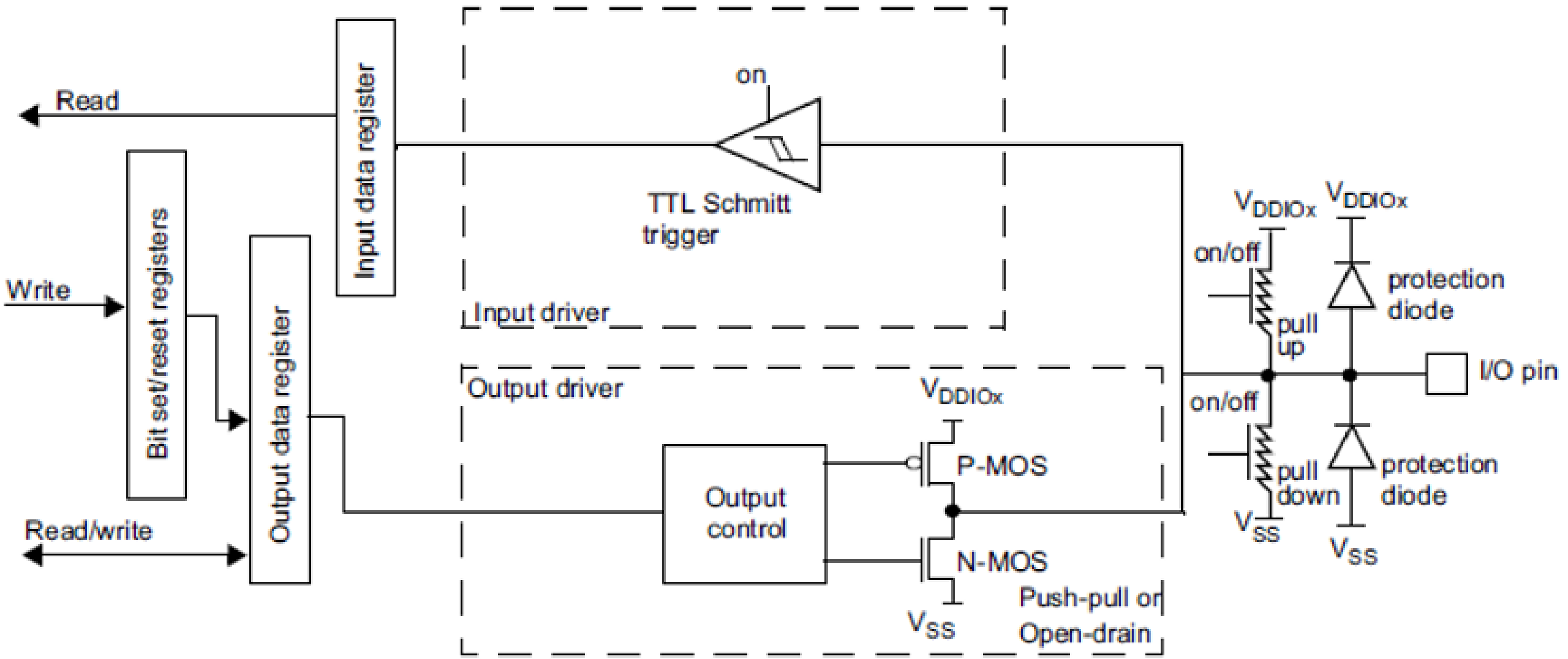
HAL_GPIO_Init

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

    while (((GPIO_Init->Pin) >> position) != 0U)
     {
       /* Get current io position */
       iocurrent = (GPIO_Init->Pin) & (1UL << position);
       if (iocurrent != 0x00u)
       {
   …
   }

}
```

IO作为输入时的电路框图

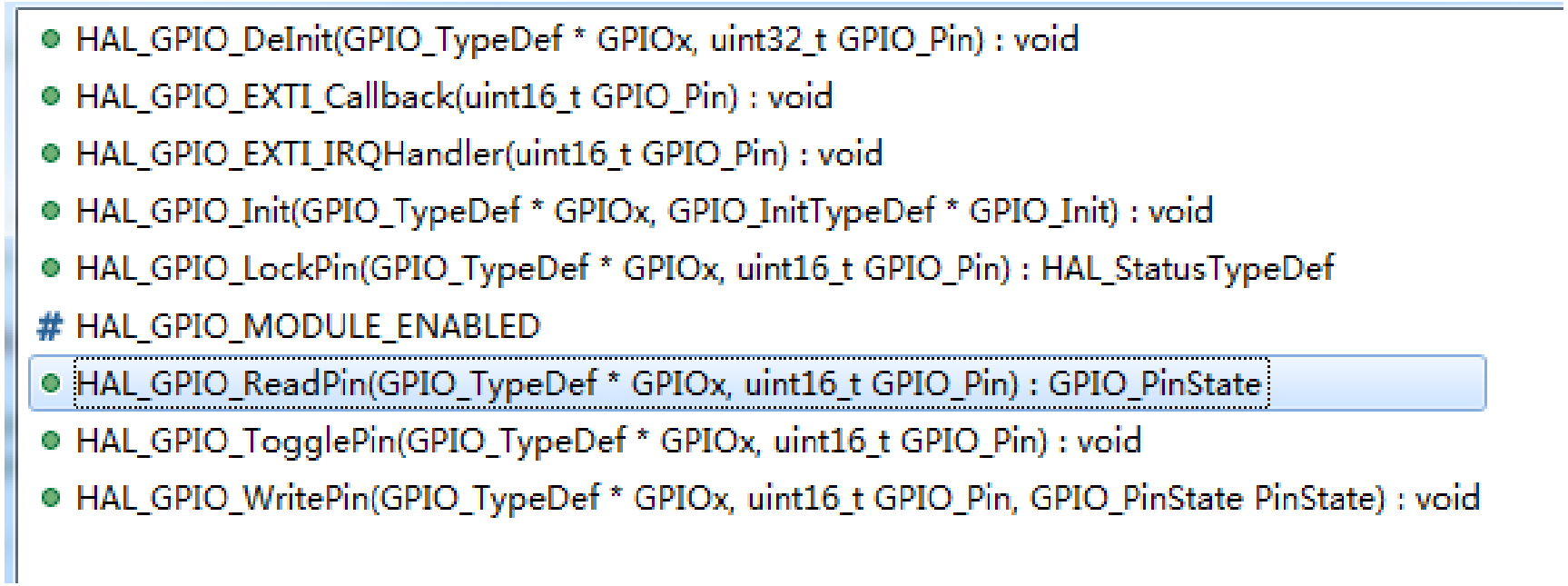


IO作为输出时的电路框图



如何查找库函数

- 键入“HAL_GPIO_”，然后用快捷键“Alt+/”，启动代码自动提示功能，会显示出固件库中所有以“HAL_GPIO_”开头的库函数



- stm32g4xx_hal_ppp.h

HAL_GPIO_ReadPin()

```
GPIO_PinState HAL_GPIO_ReadPin(GPIO_TypeDef *GPIOx, uint16_t GPIO_Pin)
{
    GPIO_PinState bitstatus;
    /* Check the parameters */
    assert_param(IS_GPIO_PIN(GPIO_Pin));
    if ((GPIOx->IDR & GPIO_Pin) != 0x00U)
    {
        bitstatus = GPIO_PIN_SET;
    }
    else
    {
        bitstatus = GPIO_PIN_RESET;
    }
    return bitstatus;
}
```

GPIO读写语句

- 配置端口的模式时，PC13的User Label: KEY，PA5的User Label: LED
- main.h文件中，可以看到如下的宏定义：

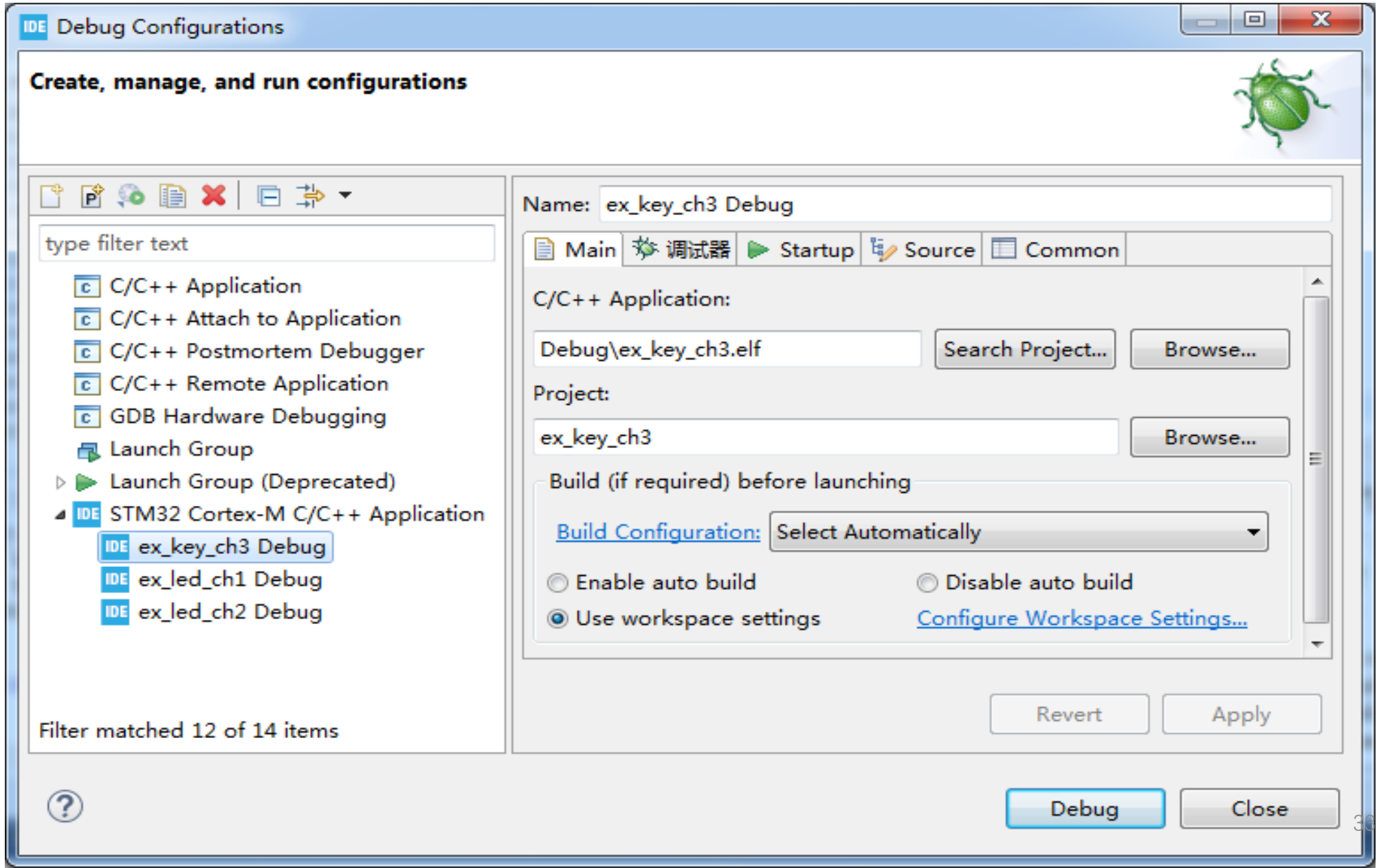
```
#define KEY_Pin GPIO_PIN_13
#define KEY_GPIO_Port GPIOC
#define LED_Pin GPIO_PIN_5
#define LED_GPIO_Port GPIOA
```

```
HAL_GPIO_ReadPin(KEY_GPIO_Port, KEY_Pin);

HAL_GPIO_WritePin(LED_GPIO_Port, LED_Pin, GPIO_PIN_SET);
```

主循环代码

```
/* USER CODE BEGIN 1 */
 GPIO_PinState KEY;
/* USER CODE END 1 */


  /* Infinite loop */
    while (1)
    {
      /* USER CODE BEGIN 3 */
              KEY = HAL_GPIO_ReadPin(KEY_GPIO_Port, KEY_Pin);
              if (KEY == GPIO_PIN_SET)
              {
                      HAL_GPIO_WritePin(LED_GPIO_Port, LED_Pin, GPIO_PIN_RESET);
              }
              else
              {
                      HAL_GPIO_WritePin(LED_GPIO_Port, LED_Pin, GPIO_PIN_SET);
              }
    }
    /* USER CODE END 3 */
```

下载与运行


Debug Configurations
Create, manage, and run configurations
type filter text
C/C++ Application
C/C++ Attach to Application
C/C++ Postmortem Debugger
C/C++ Remote Application
GDB Hardware Debugging
Launch Group
Launch Group (Deprecated)
STM32 Cortex-M C/C++ Application
ex_key_ch3 Debug
ex_led_ch1 Debug
ex_led_ch2 Debug
Filter matched 12 of 14 items
Name: ex_key_ch3 Debug
Main
调试器
Startup
Source
Common
C/C++ Application:
Debug\ex_key_ch3.elf
Search Project...
Browse...
Project:
ex_key_ch3
Browse...
Build (if required) before launching
Build Configuration:
Select Automatically
Enable auto build
Disable auto build
Use workspace settings
Configure Workspace Settings...
Revert
Apply
Debug
Close

代码修改

■ 下载完成后，点击主菜单上的运行（Resume）按钮，就可以运行程序。

✓ 板上的LD2在初始时是点亮的，按下B1键，LD2就会熄灭

■ 如果要改成B1没按灯灭，按下灯亮，如何修改？

✓ 修改硬件配置：把硬件配置的PB5的输出电平初值，由HIGH改为Low

✓ 修改主循环代码

```
int main(void)
{
 /* USER CODE BEGIN 1 */
 GPIO_PinState KEY;
 /* USER CODE END 1 */
 HAL_Init();
 SystemClock_Config();
 MX_GPIO_Init();
 while (1)
 {
  /* USER CODE BEGIN 3 */
            KEY = HAL_GPIO_ReadPin(KEY_GPIO_Port, KEY_Pin);
            if (KEY == GPIO_PIN_SET)
                HAL_GPIO_WritePin(LED_GPIO_Port, LED_Pin, GPIO_PIN_SET);

            else
                HAL_GPIO_WritePin(LED_GPIO_Port, LED_Pin, GPIO_PIN_RESET);

 }
 /* USER CODE END 3 */
}
```

代码修改

■实现按下B1后，LD2会以1Hz的频率闪烁；松开B1后，LD2熄灭

```
while (1)
  {
    /* USER CODE BEGIN 3 */
          KEY = HAL_GPIO_ReadPin(KEY_GPIO_Port, KEY_Pin);
          if (KEY == GPIO_PIN_SET)
          {
                HAL_GPIO_TogglePin(LED_GPIO_Port, LED_Pin);
                HAL_Delay(500);
          }
          else
          {
                HAL_GPIO_WritePin(LED_GPIO_Port, LED_Pin, GPIO_PIN_RESET);
          }
  }
  /* USER CODE END 3 */
```

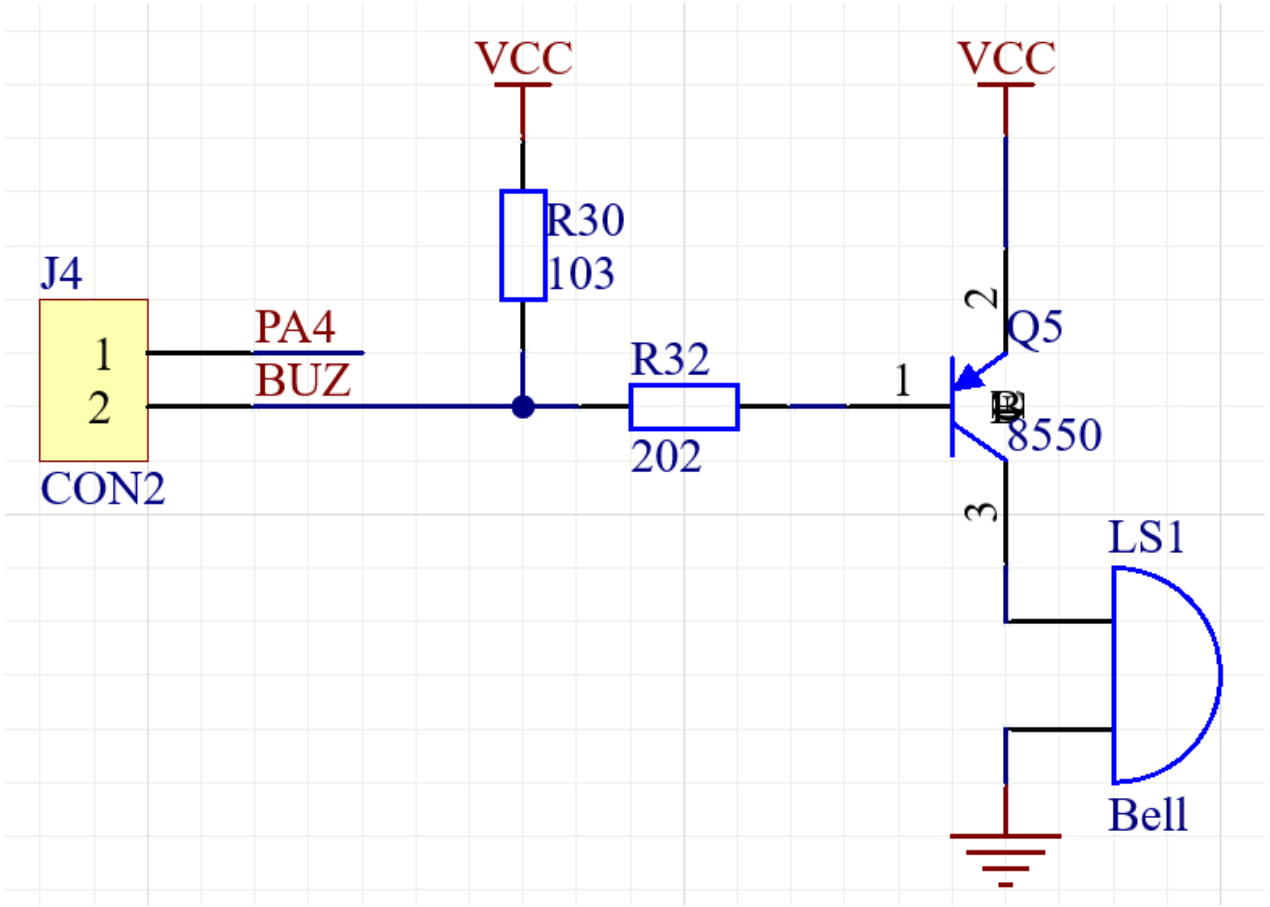
练习3：按键输入

任务3.1、按下B1后，LD2会以1Hz的频率闪烁；松开B1后，LD2以5Hz频率闪烁。

任务3.2、第1次按下B1，LD2以0.25Hz频率闪烁；第2次按下B1，LD2以1Hz频率闪烁；第3次按下B1，LD2以2Hz频率闪烁；再按B1，重复上述过程。

控制蜂鸣器

- 扩展板上的蜂鸣器电路



硬件连线及代码修改

- 可用PA4（J4）控制蜂鸣器
- 代码修改：

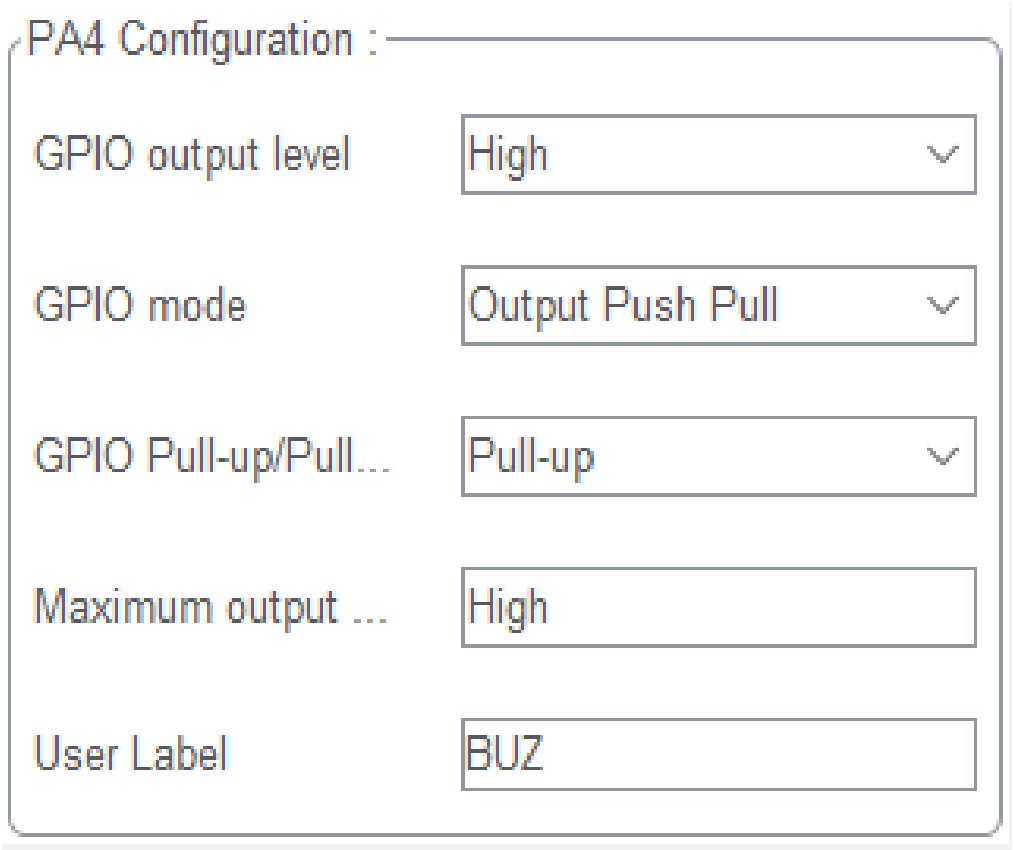
✓ 配置GPIO：

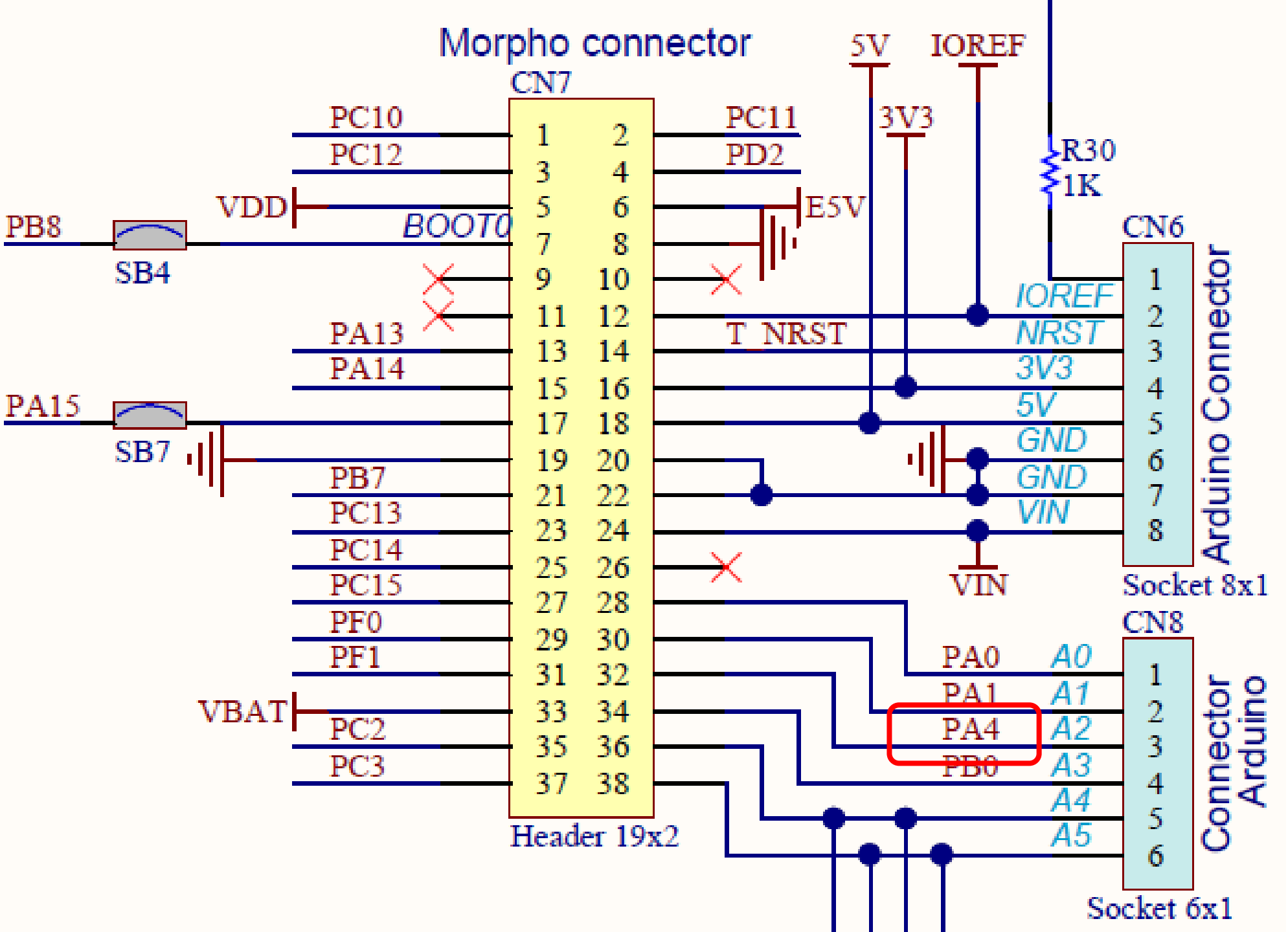
打开.ioc文件，将PA4配置为GPIO_Output

✓ 修改代码

```
HAL_GPIO_TogglePin(BUZ_GPIO_Port, BUZ_Pin);
```

✓ 编译、下载，运行




Morpho connector
CN7
PC10
PC12
VDD
PB8
BOOT0
SB4
PA13
PA14
PA15
SB7
PB7
PC13
PC14
PC15
PF0
PF1
VBAT
PC2
PC3
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
PC11
PD2
E5V
T_NRST
Header 19x2
5V
IOREF
3V3
R30
1K
CN6
IOREF
NRST
3V3
5V
GND
GND
VIN
VIN
Arduino Connector
Socket 8x1
CN8
PA0
PA1
PA4
PB0
A0
A1
A2
A3
A4
A5
Arduino Connector
Socket 6x1

练习3：按键输入

任务3.3、按下B1后，蜂鸣器以1Hz的频率发出响声；松开B1后，蜂鸣器不响。

任务3.4、在任务3.3的基础上，实现用按键切换流水灯的效果（自由发挥）。

任务3.5、编程实现以下功能：

连续按B1键次数为 *N* 时，蜂鸣器响 *N* 次；并将按键次数通过L1~L8，以二进制方式显示出来（亮1灭0；LED1为最低位）

任务3.6、在任务3.5基础上，实现用扩展板上的数码管显示计数

提交网络学堂：每个子任务的工程文件（压缩），代码有简单注释

谢　谢!