



南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

Embedded System and Microcomputer Principle

LAB2 General-purpose Input/Output

2022 Fall
wangq9@mail.sustech.edu.cn



CONTENTS

- 1 GPIO Function Description
- 2 How to create a new project
- 3 How to programming
- 4 Practice



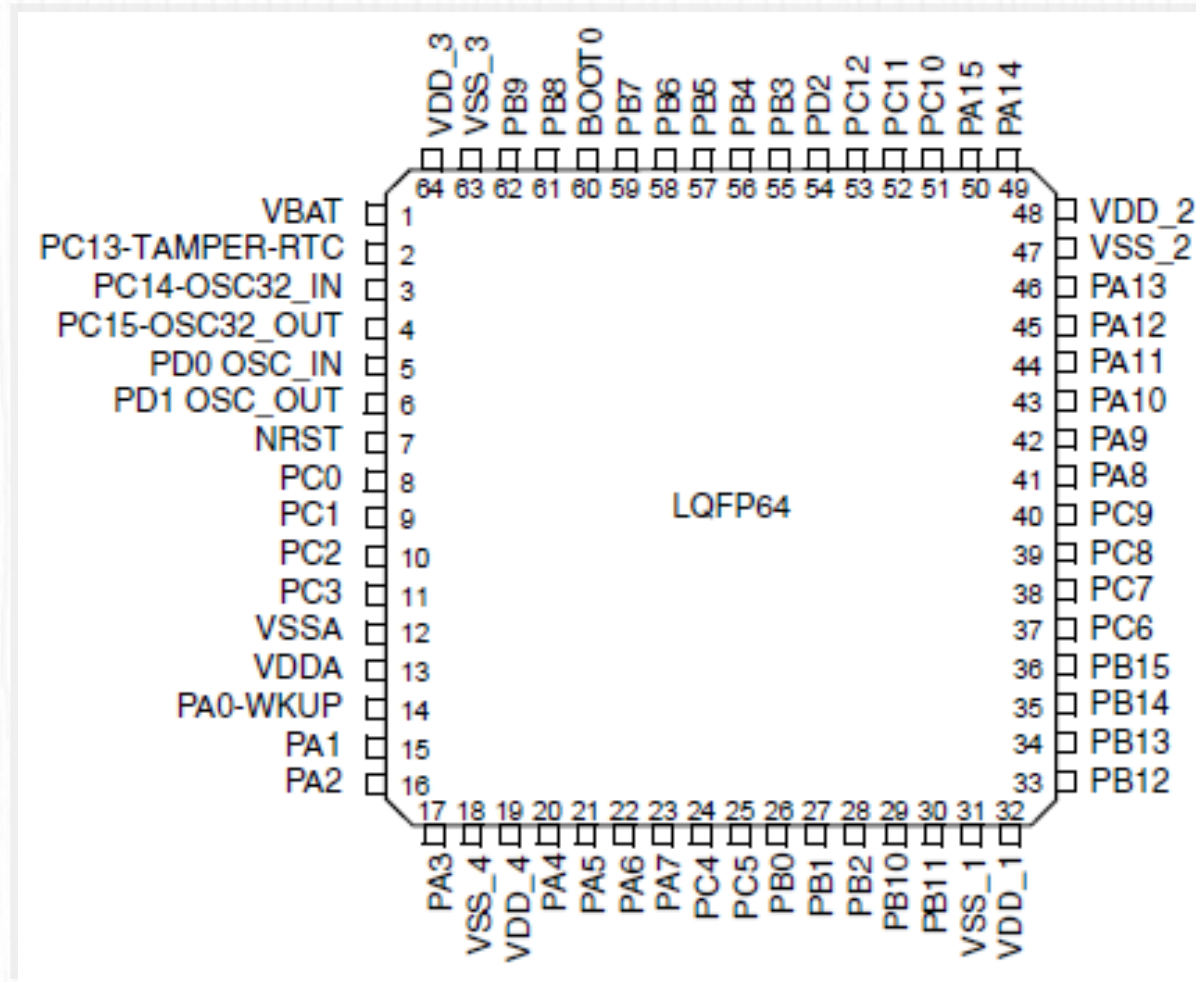
01

GPIO Function Description



1. GPIO Function Description

- There are 4 groups of I/O in STM32F103RCT6
- 51 I/O ports
 - GPIOA0~A15
 - GPIOB0~B15
 - GPIOC0~C15
 - GPIOD0~D2
 - $16 \times 3 + 3 = 51$



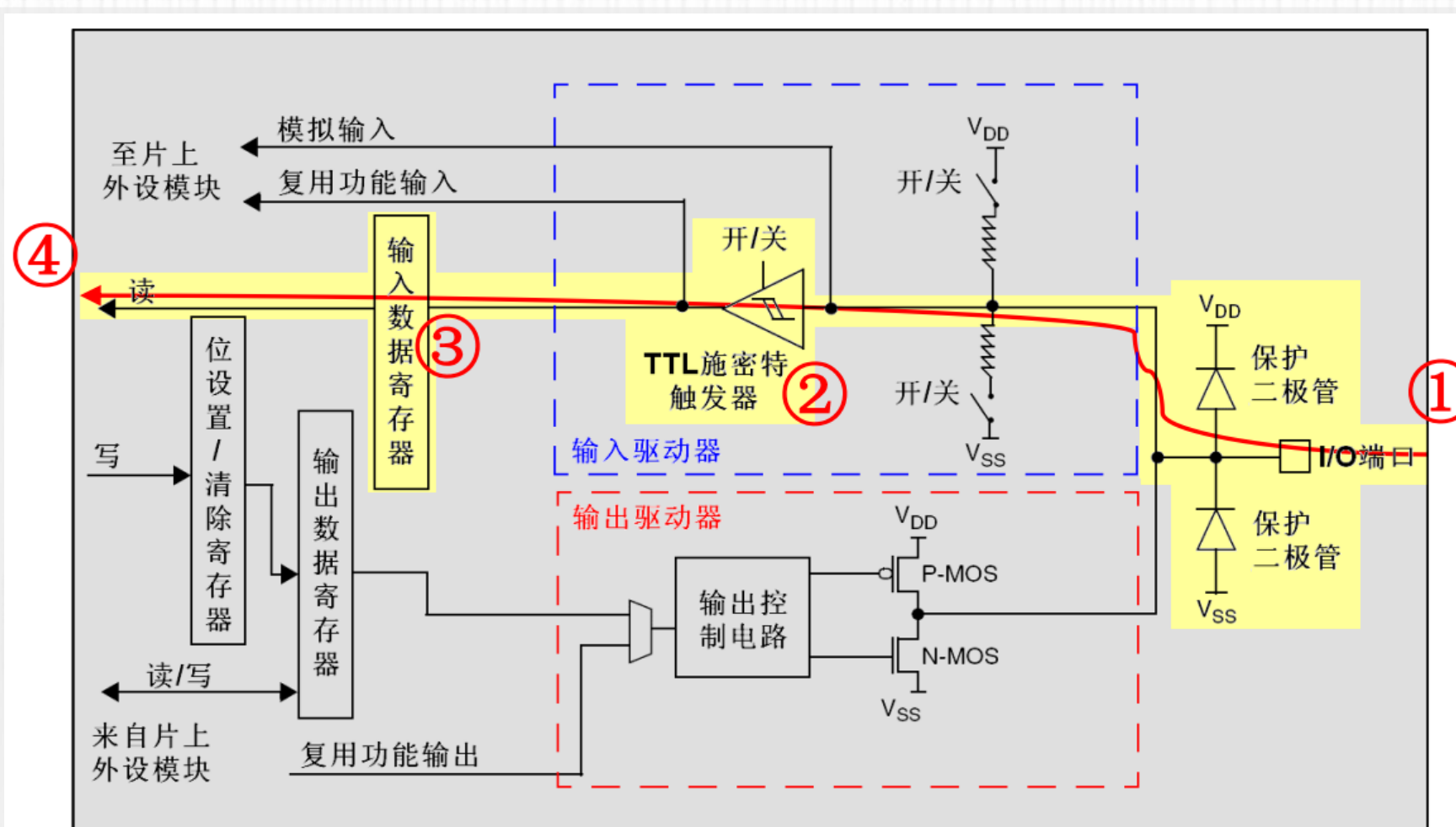


1. GPIO Function Description

- Each GPIO port in STM32 can be individually configured by software in 8 modes
 - 输入浮空 Input floating
 - 输入上拉 Input pull-up
 - 输入下拉 Input pull-down
 - 模拟输入 Analog
 - 开漏输出 Output open-drain with pull-up or pull-down capability
 - 开漏复用功能 Alternate function open-drain with pull-up or pull-down capability
 - 推挽式输出 Output push-pull with pull-up or pull-down capability
 - 推挽式复用功能 Alternate function push-pull with pull-up or pull-down capability
- More about GPIO and the corresponding feature
 - <https://blog.stratifylabs.co/device/2013-10-21-Understanding-Microcontroller-Pin-Input-Output-Modes/>
 - <http://www.openedv.com/posts/list/21980.htm>

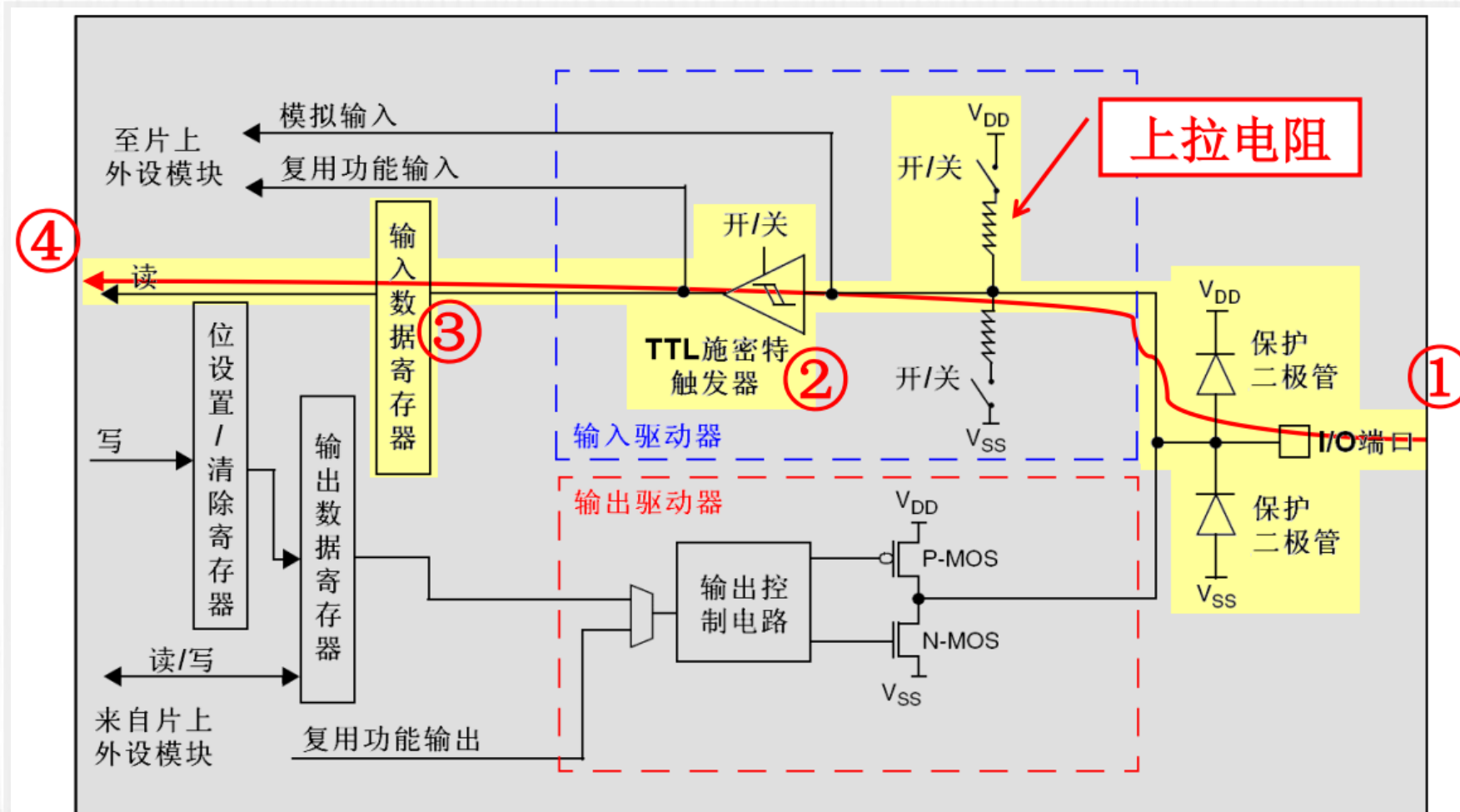
1. GPIO Function Description

- Input floating 输入浮空



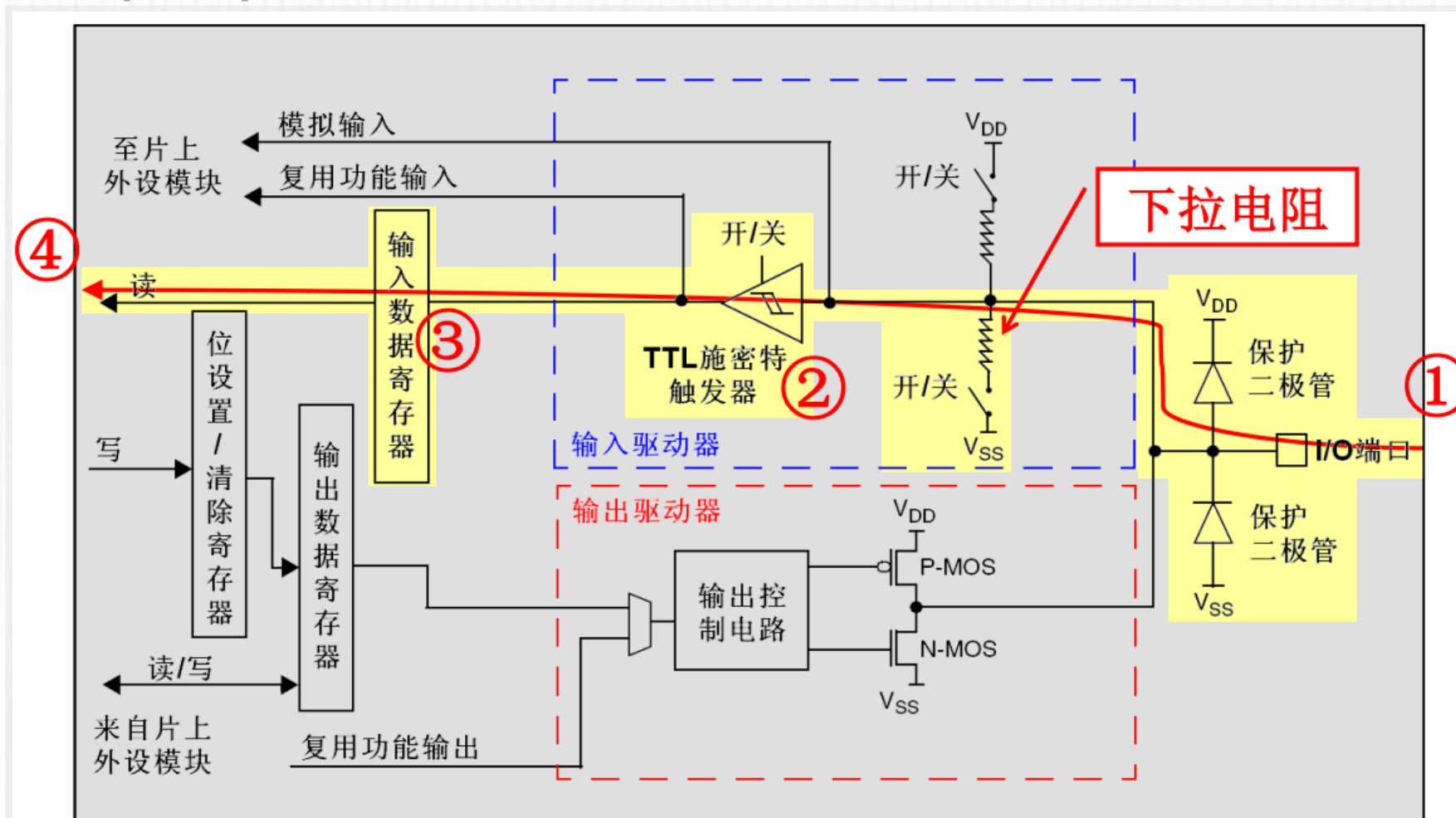
1. GPIO Function Description

- Input pull-up 输入上拉



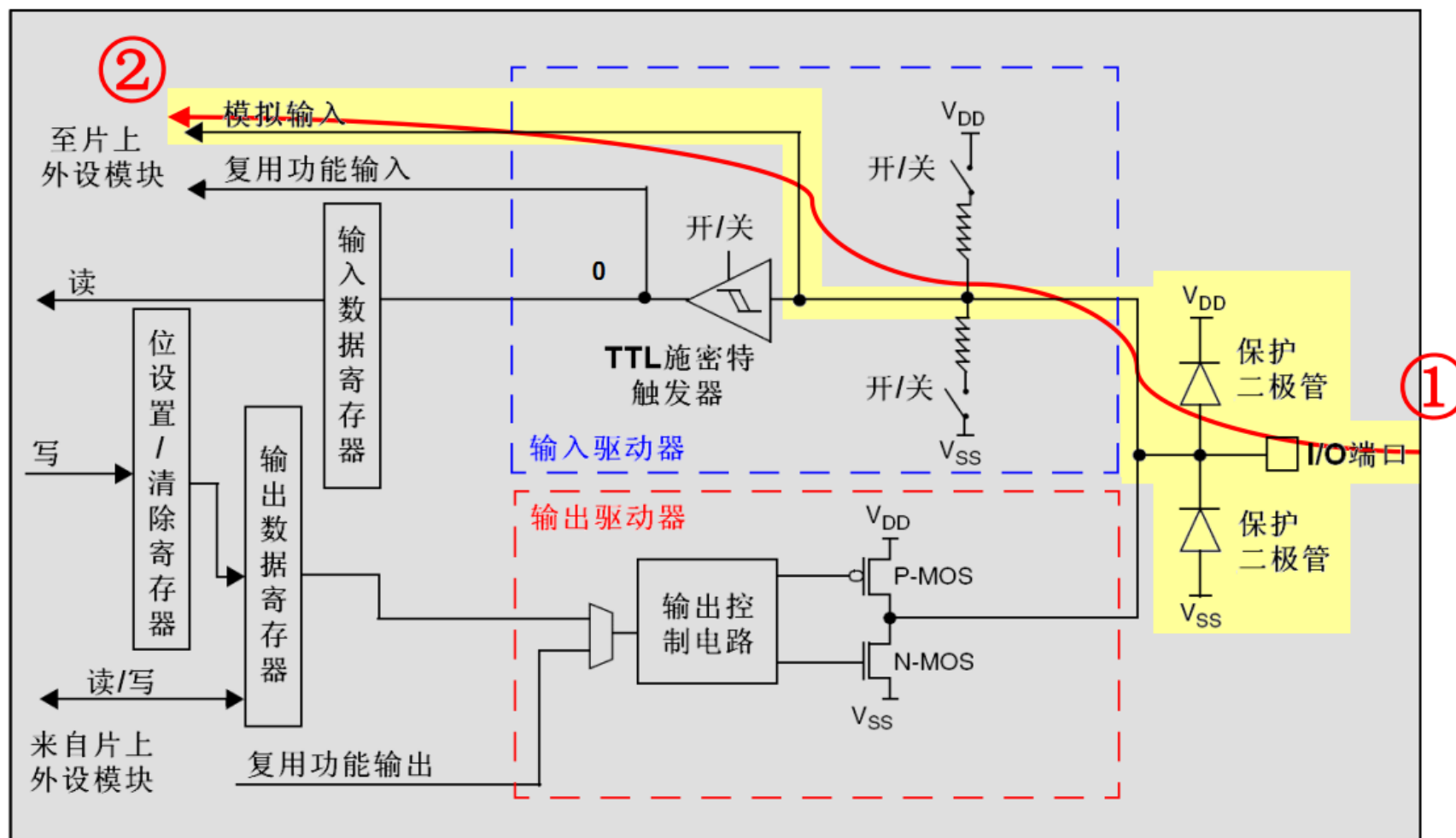
1. GPIO Function Description

- Input pull-down 输入下拉



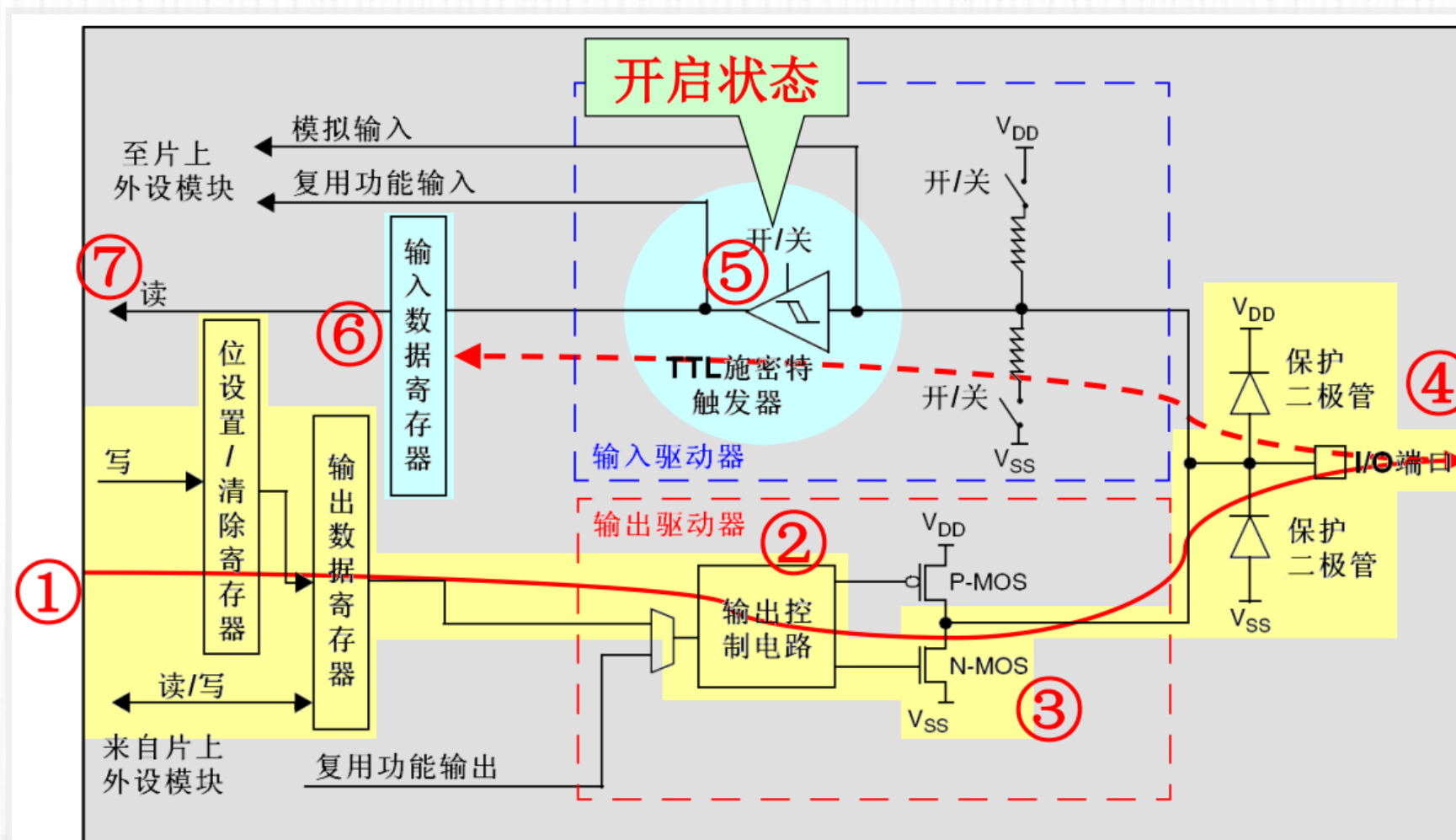
1. GPIO Function Description

- Analog 模拟输入



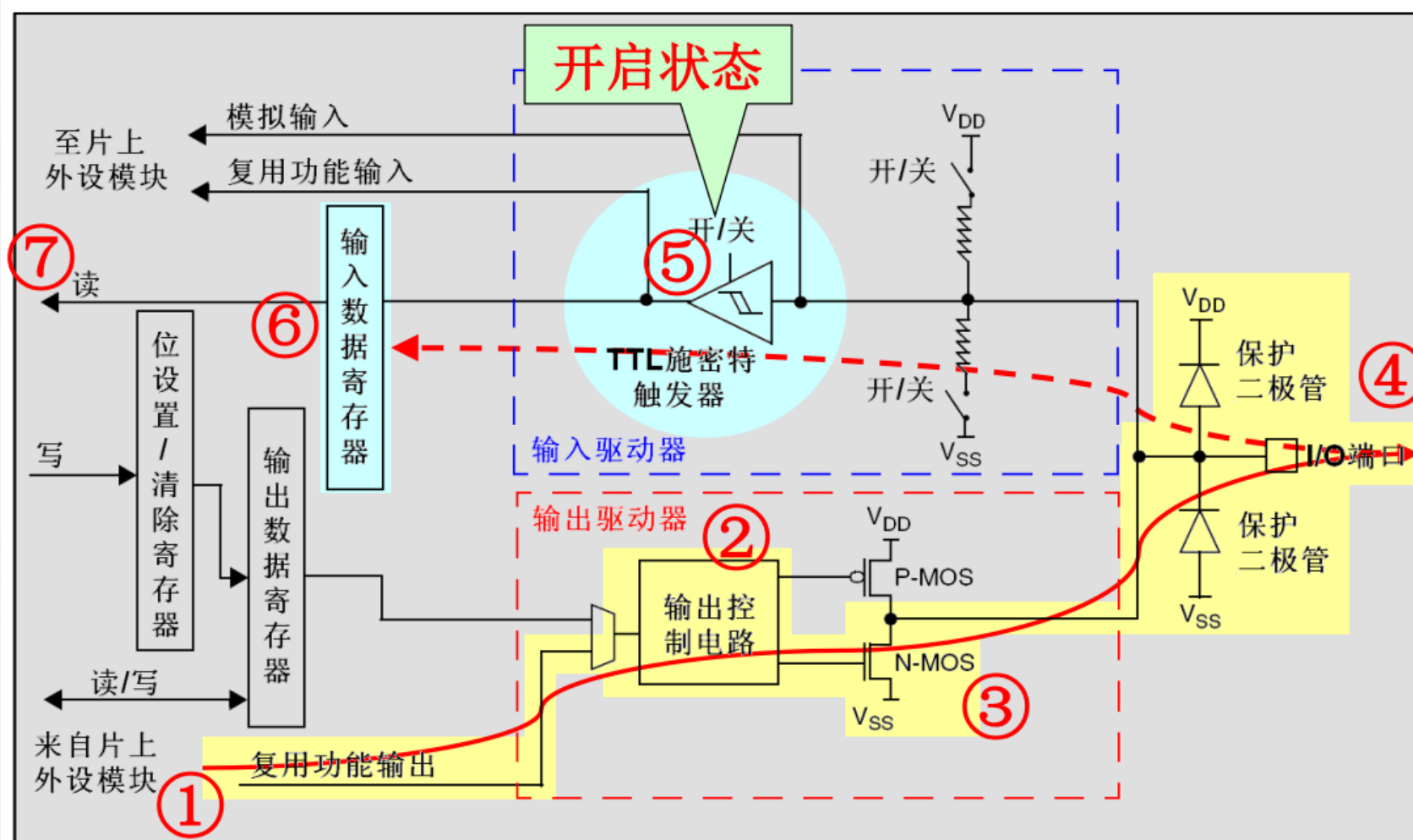
1. GPIO Function Description

- Output open-drain 开漏输出



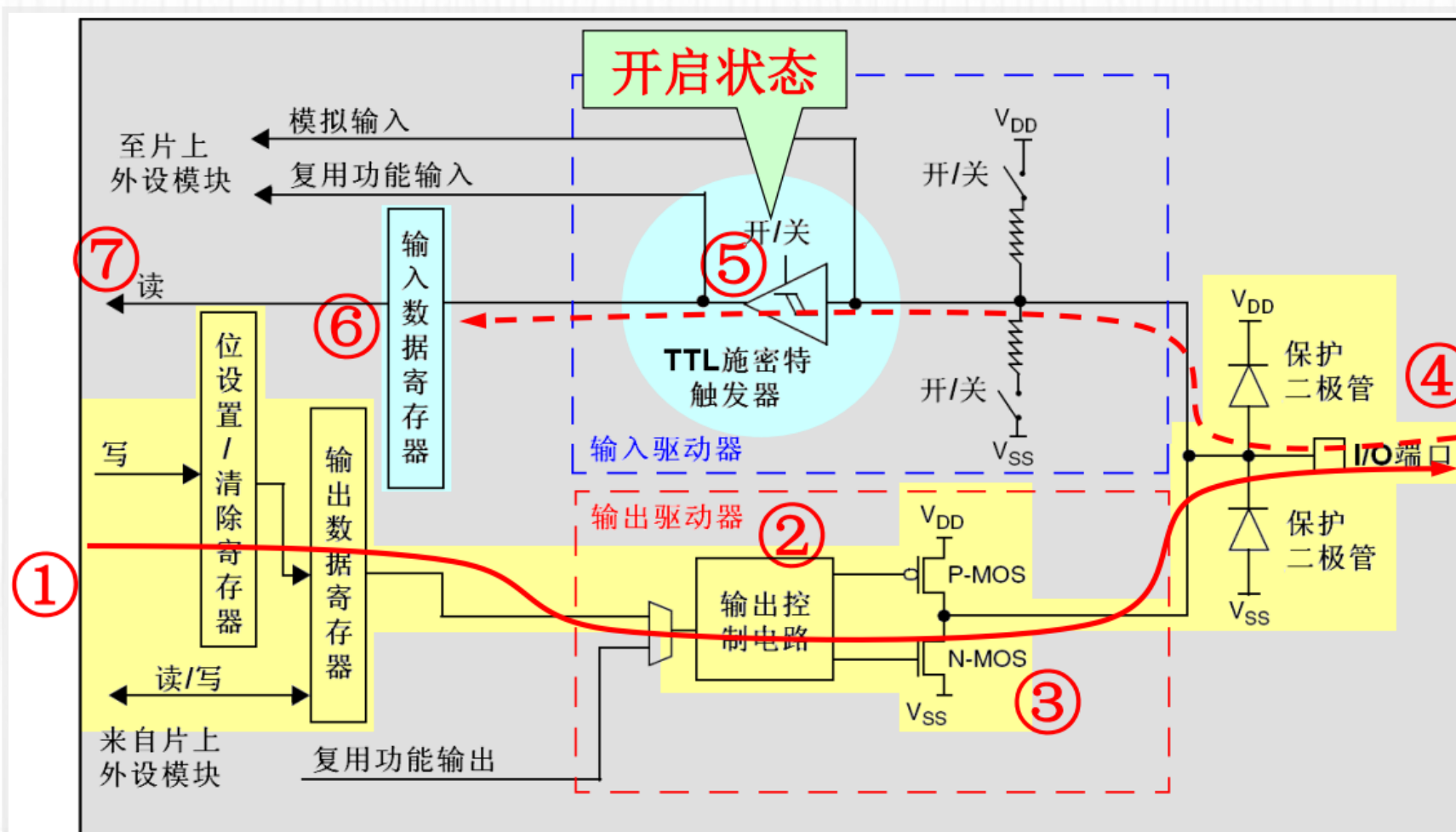
1. GPIO Function Description

- Alternate function open-drain 开漏复用功能



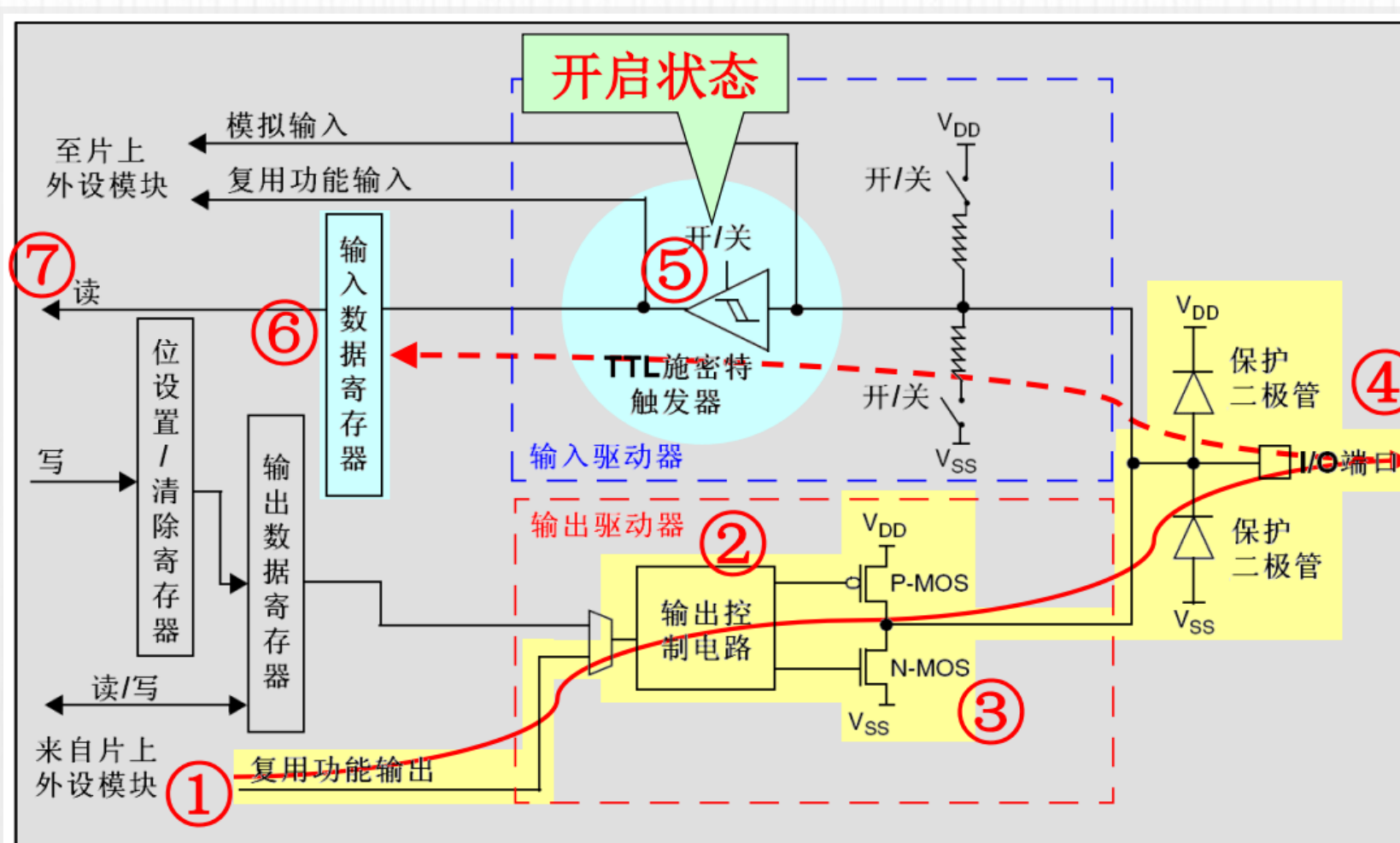
1. GPIO Function Description

- Output push-pull 推挽输出



1. GPIO Function Description

- Alternate function push-pull 推挽式复用功能





1. GPIO Function Description

- Each group GPIO ports has **7** registers
 - two 32-bit configuration registers (GPIOx_CRL, GPIOx_CRH)
 - GPIOx_CRL: Port configuration register low
 - GPIOx_CRH: Port configuration register high
 - two 32-bit data registers (GPIOx_IDR, GPIOx_ODR)
 - GPIOx_IDR: Port input data register
 - GPIOx_ODR: Port output data register
 - a 32-bit set/reset register (GPIOx_BSRR)
 - a 16-bit reset register (GPIOx_BRR)
 - a 32-bit locking register (GPIOx_LCKR)
- Each I/O port bit is freely programmable, however the I/O port registers have to be accessed as 32-bit words (half-word or byte accesses are not allowed)



1. GPIO Function Description

-- 端口配置低寄存器(GPIOx_CRL)

CNF7[1:0]		MODE7[1:0]		CNF6[1:0]		MODE6[1:0]		CNF5[1:0]		MODE5[1:0]		CNF4[1:0]		MODE4[1:0]	
rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNF3[1:0]		MODE3[1:0]		CNF2[1:0]		MODE2[1:0]		CNF1[1:0]		MODE1[1:0]		CNF0[1:0]		MODE0[1:0]	
rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW

位31:30 27:26 23:22 19:18 15:14 11:10 7:6 3:2	CNFy[1:0]: 端口x配置位(y = 0...7) (Port x configuration bits) 软件通过这些位配置相应的I/O端口, 请参考表17端口位配置表。 在输入模式(MODE[1:0]=00): 00: 模拟输入模式 01: 浮空输入模式(复位后的状态) 10: 上拉/下拉输入模式 11: 保留 在输出模式(MODE[1:0]>00): 00: 通用推挽输出模式 01: 通用开漏输出模式 10: 复用功能推挽输出模式 11: 复用功能开漏输出模式
位29:28 25:24 21:20 17:16 13:12 9:8, 5:4 1:0	MODEy[1:0]: 端口x的模式位(y = 0...7) (Port x mode bits) 软件通过这些位配置相应的I/O端口, 请参考表17端口位配置表。 00: 输入模式(复位后的状态) 01: 输出模式, 最大速度10MHz 10: 输出模式, 最大速度2MHz 11: 输出模式, 最大速度50MHz



1. GPIO Function Description

-- 端口配置低寄存器(GPIOx_CRL)

表17 端口位配置表

配置模式		CNF1	CNF0	MODE1	MODE0	PxODR寄存器		
通用输出	推挽(Push-Pull)	0	0	01 10 11 见表18		0 或 1		
	开漏(Open-Drain)		1			0 或 1		
复用功能输出	推挽(Push-Pull)	1	0					不使用
	开漏(Open-Drain)		1					不使用
输入	模拟输入	0	0	00				不使用
	浮空输入		1					不使用
	下拉输入	1	0					0
	上拉输入							1

表18 输出模式位

MODE[1:0]	意义
00	保留
01	最大输出速度为10MHz
10	最大输出速度为2MHz
11	最大输出速度为50MHz



1. GPIO Function Description

-- 端口配置高寄存器(GPIOx_CRH)

CNF15[1:0]		MODE15[1:0]		CNF14[1:0]		MODE14[1:0]		CNF13[1:0]		MODE13[1:0]		CNF12[1:0]		MODE12[1:0]	
rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNF11[1:0]		MODE11[1:0]		CNF10[1:0]		MODE10[1:0]		CNF9[1:0]		MODE9[1:0]		CNF8[1:0]		MODE8[1:0]	
rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW

位31:30	CNFy[1:0]: 端口x配置位(y = 8...15) (Port x configuration bits) 软件通过这些位配置相应的I/O端口, 请参考表17端口位配置表。 在输入模式(MODE[1:0]=00): 00: 模拟输入模式 01: 浮空输入模式(复位后的状态) 10: 上拉/下拉输入模式 11: 保留 在输出模式(MODE[1:0]>00): 00: 通用推挽输出模式 01: 通用开漏输出模式 10: 复用功能推挽输出模式 11: 复用功能开漏输出模式
27:26	
23:22	
19:18	
15:14	
11:10	
7:6	MODEy[1:0]: 端口x的模式位(y = 8...15) (Port x mode bits) 软件通过这些位配置相应的I/O端口, 请参考表17端口位配置表。 00: 输入模式(复位后的状态) 01: 输出模式, 最大速度10MHz 10: 输出模式, 最大速度2MHz 11: 输出模式, 最大速度50MHz
3:2	
位9:28	
25:24	
21:20	
17:16	
13:12	
9:8, 5:4	
1:0	



1. GPIO Function Description

-- 端口输入数据寄存器(GPIOx_IDR)



位31:16	保留，始终读为0。
位15:0	IDRy[15:0] : 端口输入数据(y = 0...15) (Port input data) 这些位为只读并只能以字(16位)的形式读出。读出的值为对应I/O口的状态。



1. GPIO Function Description

-- 端口输出数据寄存器(GPIOx_ODR)



位31:16	保留，始终读为0。
位15:0	ODRy[15:0]: 端口输出数据(y = 0...15) (Port output data) 这些位可读可写并只能以字(16位)的形式操作。 注：对GPIOx_BSRR(x = A...E)，可以分别地对各个ODR位进行独立的设置/清除。



1. GPIO Function Description

-- 端口位设置/清除寄存器(GPIOx_BSRR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W

位31:16

BRy: 清除端口x的位y (y = 0...15) (Port x Reset bit y)

这些位只能写入并只能以字(16位)的形式操作。

0: 对对应的ODRy位不产生影响

1: 清除对应的ODRy位为0

注: 如果同时设置了BSy和BRy的对应位, BSy位起作用。

位15:0

BSy: 设置端口x的位y (y = 0...15) (Port x Set bit y)

这些位只能写入并只能以字(16位)的形式操作。

0: 对对应的ODRy位不产生影响

1: 设置对应的ODRy位为1



1. GPIO Function Description

-- 端口位清除寄存器(GPIOx_BRR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W

位31:16	保留。
位15:0	BRy : 清除端口x的位y (y = 0...15) (Port x Reset bit y) 这些位只能写入并只能以字(16位)的形式操作。 0: 对对应的ODRy位不产生影响 1: 清除对应的ODRy位为0



1. GPIO Function Description

-- 端口复用功能

- STM32的大部分端口都具有复用功能
- 复用, 就是一些端口不仅仅可以做为通用IO口, 还可以复用为一些外设引脚, 比如PA9, PA10可以复用为STM32的串口1引脚
- 作用: 最大限度的利用端口资源

Pins						Pin name	Type ⁽¹⁾	I / O Level ⁽²⁾	Main function ⁽³⁾ (after reset)	Alternate functions	
BGA144	BGA100	WLCSP64	LQFP64	LQFP100	LQFP144					Default	Remap
D12	C9	D2	42	68	101	PA9	I/O	FT	PA9	USART1_TX ⁽⁷⁾ / TIM1_CH2 ⁽⁷⁾	
D11	D10	D3	43	69	102	PA10	I/O	FT	PA10	USART1_RX ⁽⁷⁾ / TIM1_CH3 ⁽⁷⁾	

1. GPIO Function Description

-- 端口重映射功能

- 就是可以把某些功能引脚映射到其他引脚
- 比如串口1默认引脚是PA9,PA10, 可以通过配置重映射映射到PB6,PB7
- 作用：方便布线
- 所有IO口都可以作为中断输入

Pins						Pin name	Type ⁽¹⁾	I / O Level ⁽²⁾	Main function ⁽³⁾ (after reset)	Alternate functions	
BGA144	BGA100	WLCSP64	LQFP64	LQFP100	LQFP144					Default	Remap
C6	B5	B5	58	92	136	PB6	I/O	FT	PB6	I2C1_SCL ⁽⁷⁾ / TIM4_CH1 ⁽⁷⁾	USART1_TX
D6	A5	C5	59	93	137	PB7	I/O	FT	PB7	I2C1_SDA ⁽⁷⁾ / FSMC_NADV / TIM4_CH2 ⁽⁷⁾	USART1_RX

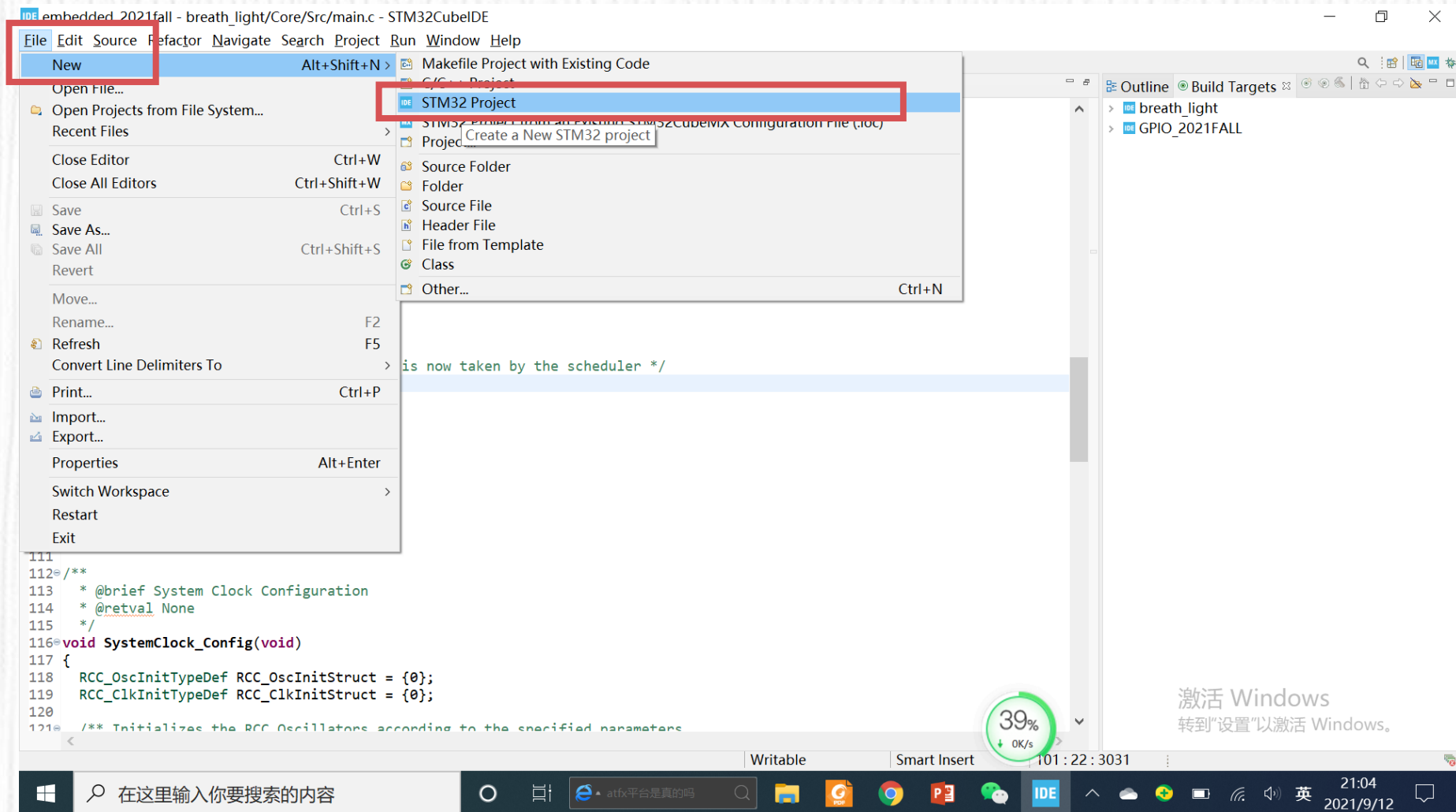


02

How to create a new project

2. How to create a new project

- Create a new STM32 project





2. How to create a new project

- Target selection -- STM32F103RCTx

IDE STM32 Project

Target Selection
Select STM32 target or STM32Cube example

MCU/MPU Selector | Board Selector | Example Selector | Cross Selector

MCU/MPU Filters

- ★
- 🔍
- 📁
- 🔄

Part Number

Core >

Series >

Line >

Package >

Other >

Peripheral >

Features | Block Diagram | Docs & Resources | Datasheet | Buy


STM32F1 Series

STM32F103RC

Mainstream Performance line, Arm Cortex-M3 MCU with 256 Kbytes of Flash memory, 72 MHz CPU, motor control, USB and CAN

ACTIVE Active
Product is in mass production

Unit Price for 10kU (US\$) : 2.778

 LQFP64

STM32F103RCTx

MCUs/MPUs List: 2 items [+ Display similar items](#) [Export](#)

	Part No	Reference	Marketing	Unit Price	Board	Package	Flash	RAM	IO	Freq.
★	STM32F103RC	STM32F103RCTx	Active	2.778		LQFP64	256 kBytes	48 kBytes	51	72 MHz
★	STM32F103RC	STM32F103RCYx	Active	2.778		WLCSP64	256 kBytes	48 kBytes	50	72 MHz



2. How to create a new project

- Enter the project name – only ASCII characters
- Keep other options as default

IDE STM32 Project

Setup STM32 project

Project

Project Name: demo_for_lab

☒ Use default location

Location: D:/wangqing/embedded_system Browse...

Options

Targeted Language

☒ C ☐ C++

Targeted Binary Type

☒ Executable ☐ Static Library

Targeted Project Type

☒ STM32Cube ☐ Empty

? < Back Next > Finish Cancel



2. How to create a new project

- Check the project information
- Click ***Finish***

The screenshot shows the 'STM32 Project' dialog box in the IDE, specifically the 'Firmware Library Package Setup' tab. The dialog is titled 'Setup STM32 target's firmware'. It contains three main sections: 'Target and Firmware Package', 'Firmware and Software Package Repository', and 'Code Generator Options'. The 'Target and Firmware Package' section shows 'Target Reference: STM32F103RCTx' and 'Firmware Package Name and Version: STM32Cube FW_F1 V1.8.4'. The 'Firmware and Software Package Repository' section shows 'Location: C:\Users\wq\STM32Cube\Repository' and a link to 'Firmware Updater'. The 'Code Generator Options' section has three radio buttons: 'Add necessary library files as reference in the toolchain project configuration file', 'Copy all used libraries into the project folder', and 'Copy only the necessary library files', with the third option selected. At the bottom, there are buttons for '?', '< Back', 'Next >', 'Finish' (highlighted), and 'Cancel'.

IDE STM32 Project

Firmware Library Package Setup

Setup STM32 target's firmware

Target and Firmware Package

Target Reference: STM32F103RCTx

Firmware Package Name and Version: STM32Cube FW_F1 V1.8.4

Firmware and Software Package Repository

Location:
C:\Users\wq\STM32Cube\Repository

See ['Firmware Updater'](#) for settings related to package installation

Code Generator Options

☐ Add necessary library files as reference in the toolchain project configuration file

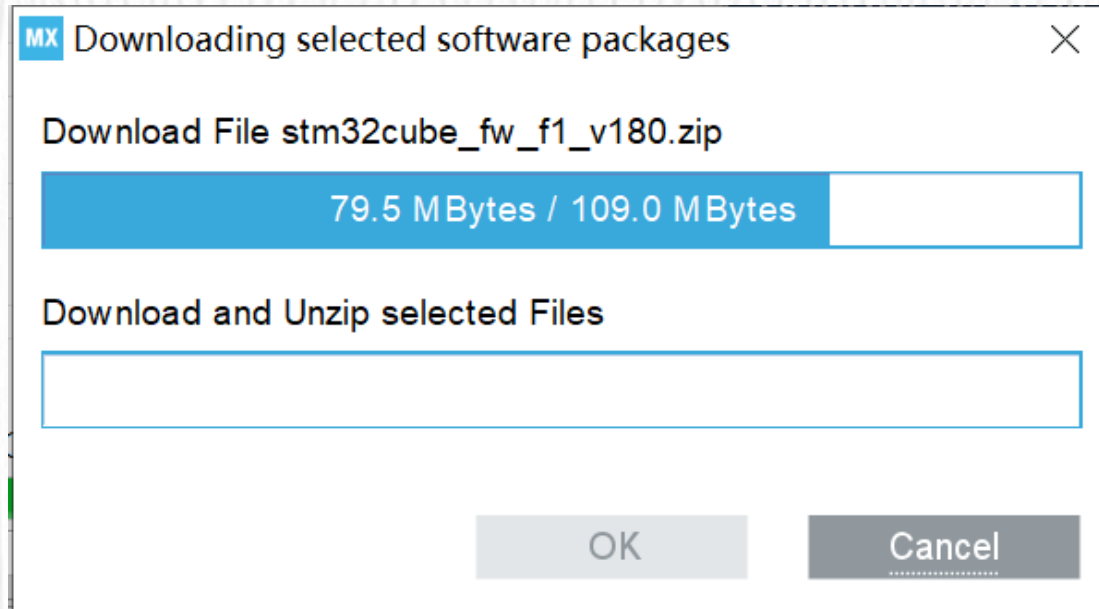
☐ Copy all used libraries into the project folder

☒ Copy only the necessary library files

? < Back Next > Finish Cancel

2. How to create a new project

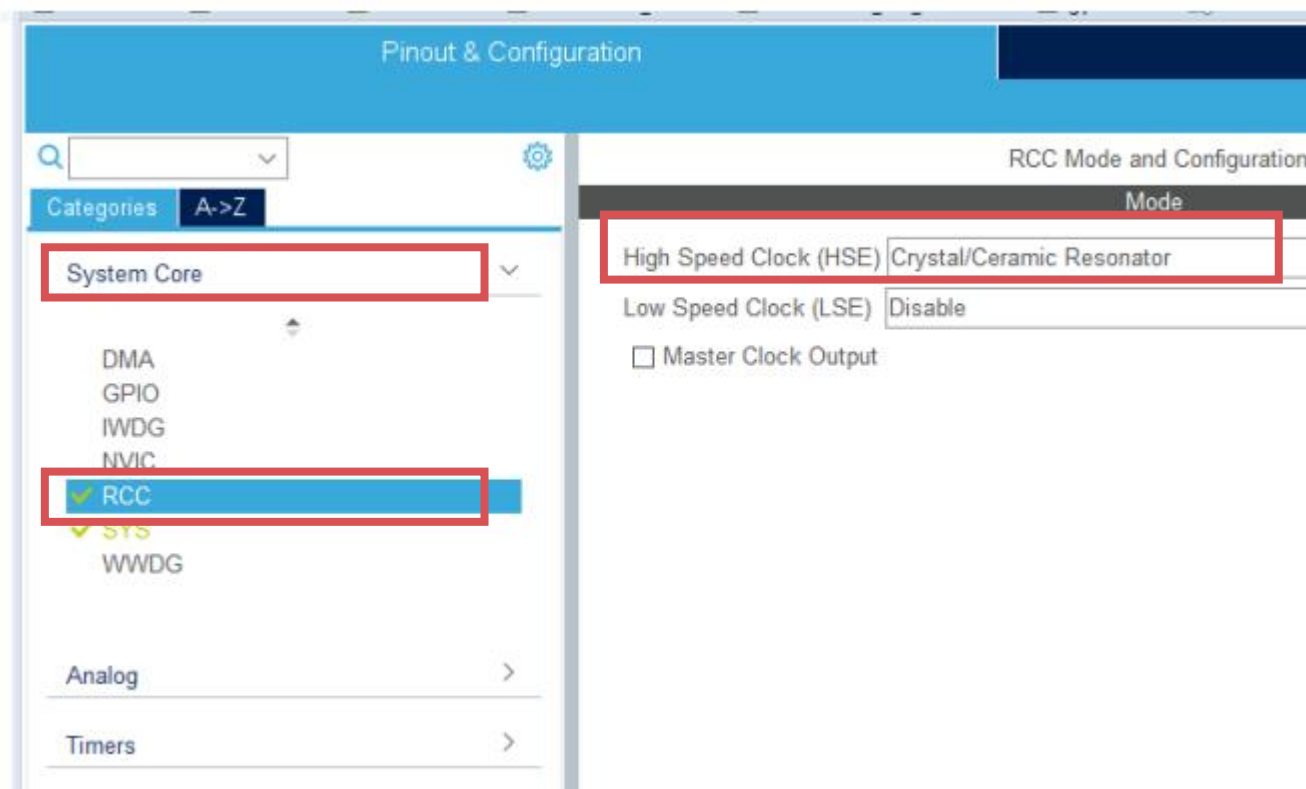
- Download software packages





2. How to create a new project

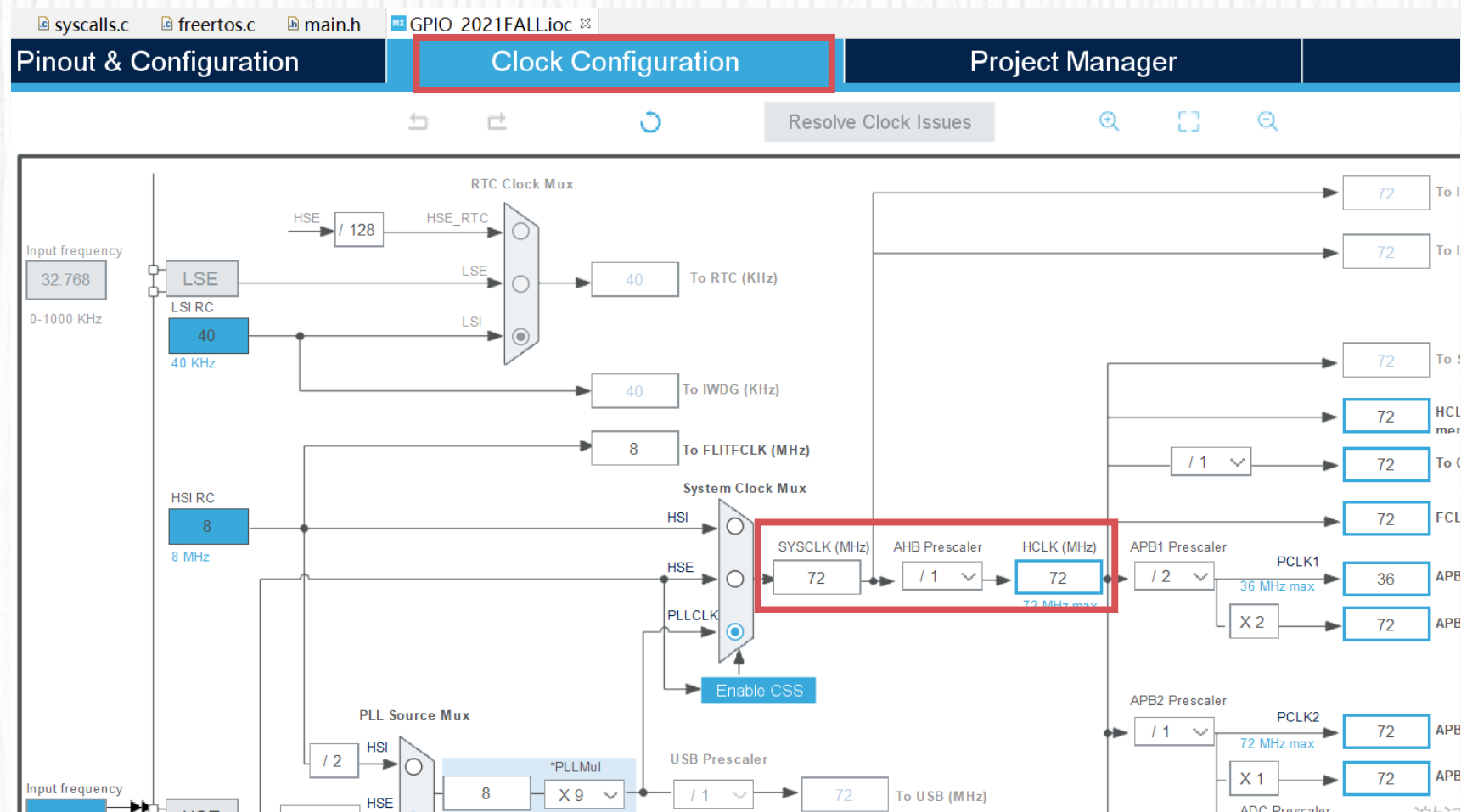
- RCC and Clock Configuration
- There are four kinds of clock sources in STM32: HSE clock, HSI clock, LSE clock and LSI clock (HS for high speed, LS for low speed, E for external and I for internal)
- Only HSE and HSI clock can be used to drive the SYSCLK
- Most of the time we use an external crystal oscillator or ceramic resonator (HSE) to drive the SYSCLK because it is more accurate





2. How to create a new project

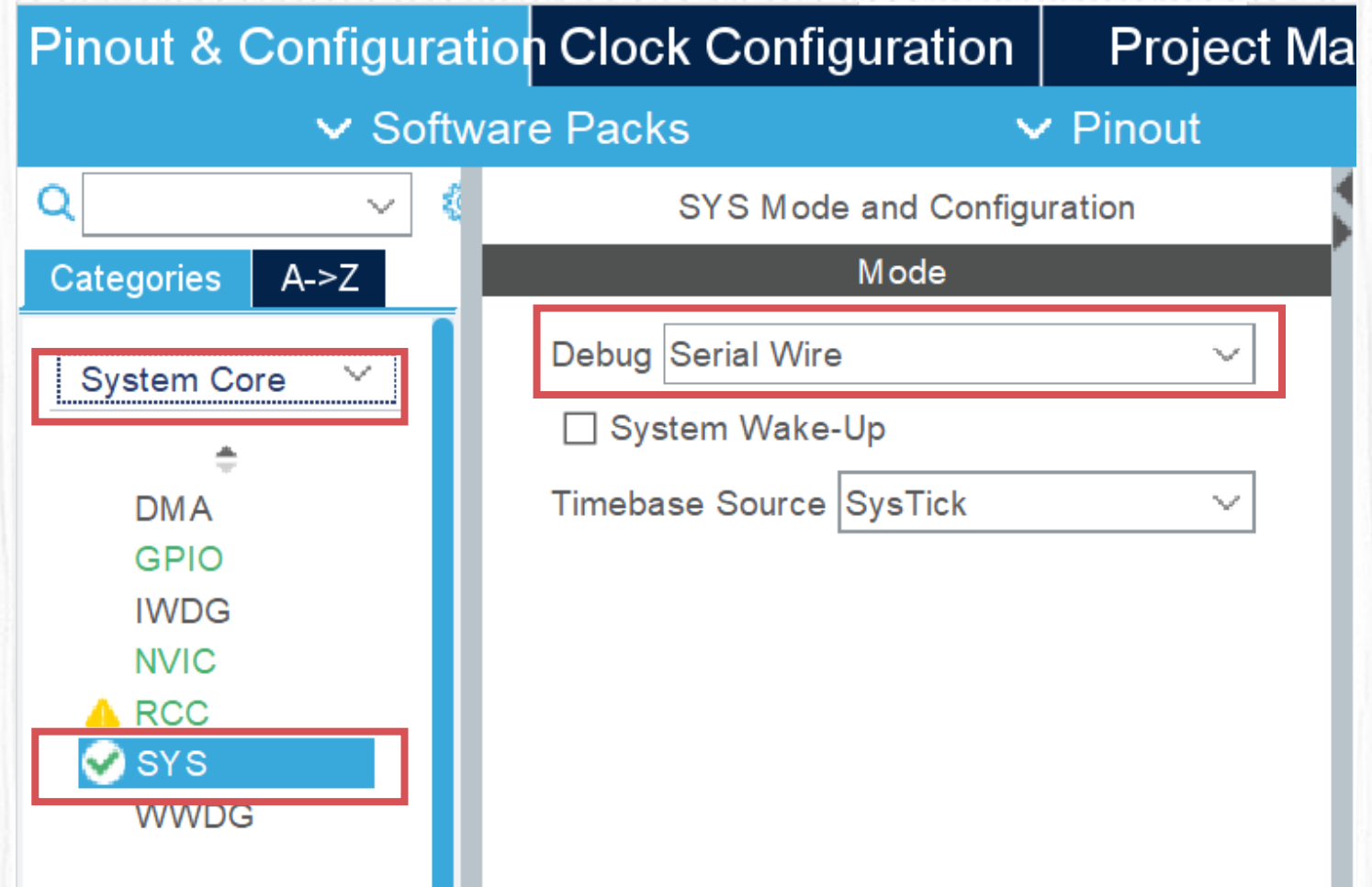
- Clock Configuration
- Change to **clock configuration**
- Set SYSCLK as 72M (maximum)





2. How to create a new project

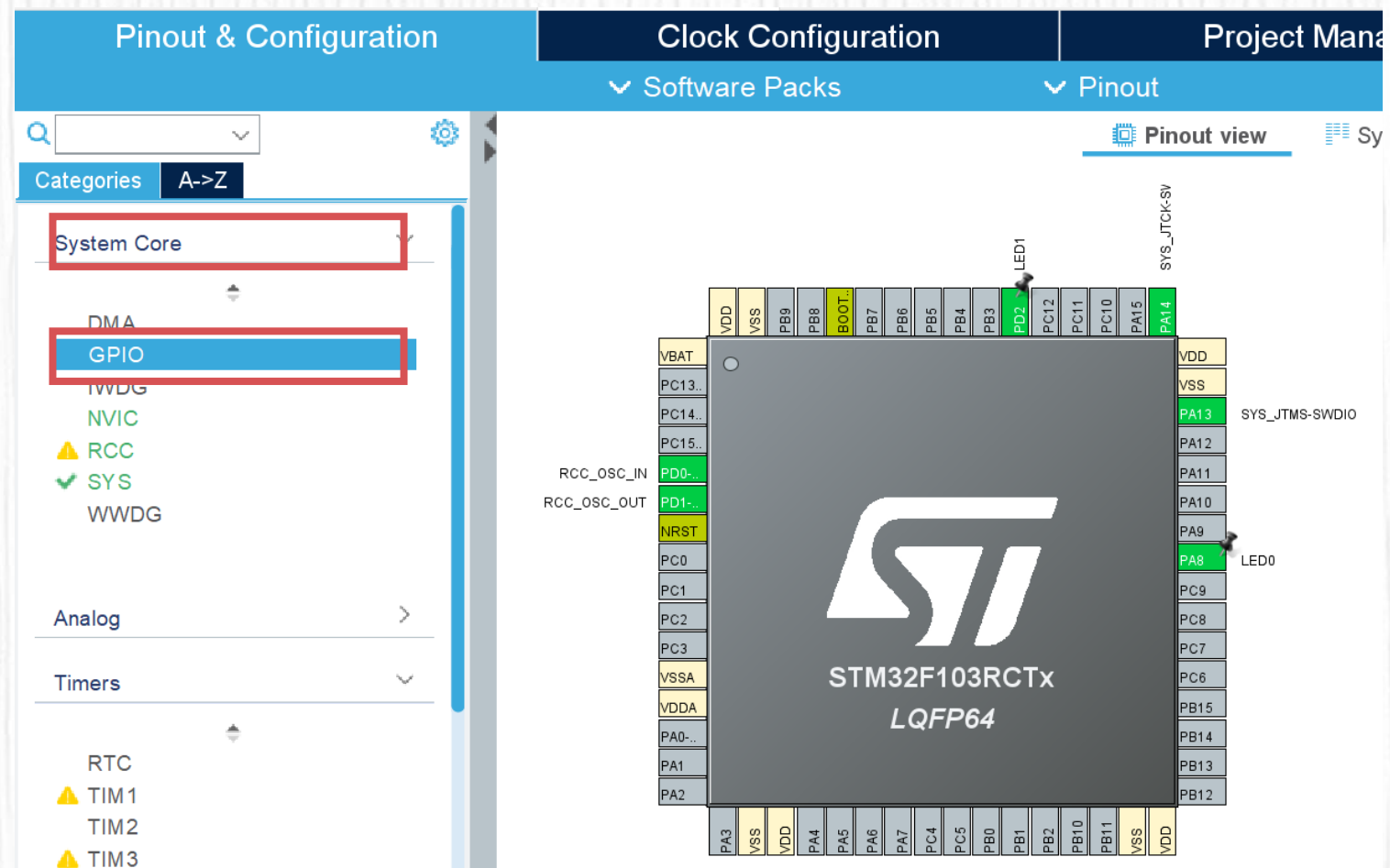
- SYS Mode and Configuration
- Back to **Pinout & Configuration** -> **Categories** -> **System Core** -> **SYS**
- Use Serial Wire(SW) as debug wire





2. How to create a new project

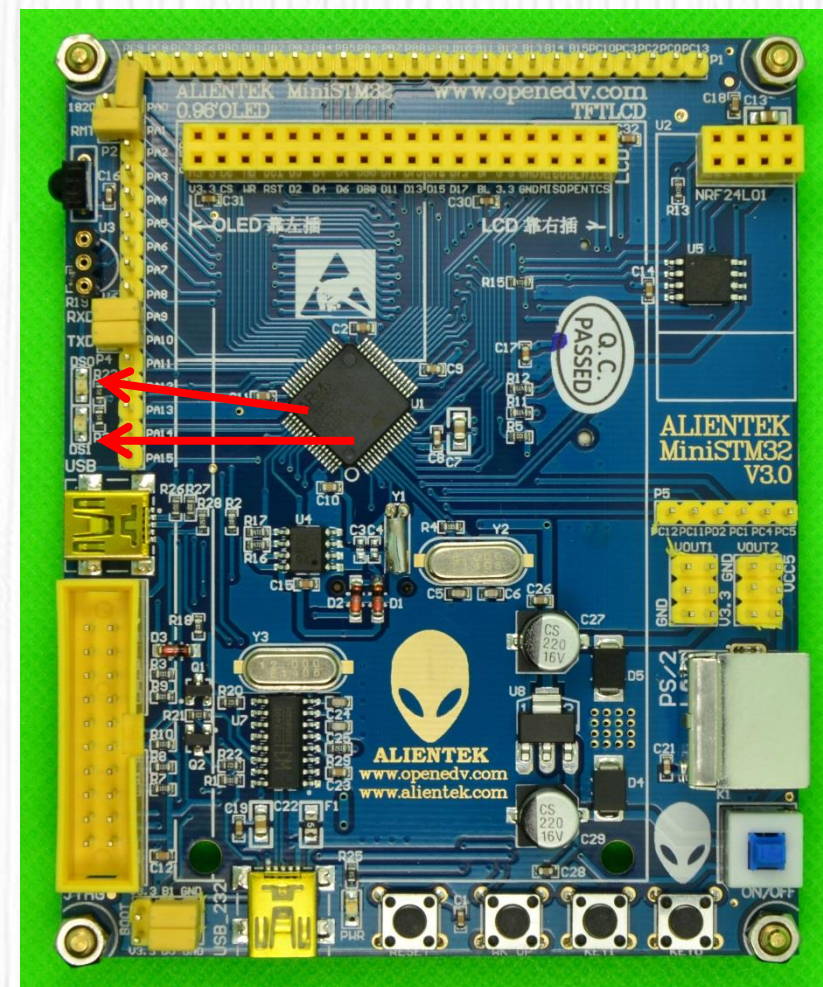
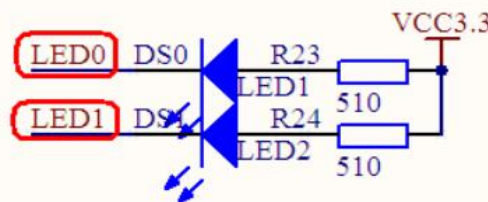
- GPIO Configuration
- Still in **Pinout & Configuration** -> **Categories** -> **System Core** -> **GPIO**
- Set PA8 as GPIO_Output, and rename as LED0
- Set PD2 as GPIO_Output, and rename as LED1



2. How to create a new project

- GPIO Configuration

PB3/JTDO/SPI3_SCK/I2S3_CK	53	PC12	IIC SCL
D2/TIM3_ETR/U5_RX/SDIO_CMD	52	PC11	IIC SDA
PC12/U5_TX/SDIO_CK	51	PC10	LCD BL
PC11/U4_RX/SDIO_D3	50	PA15	JTDI P
CH1/TIM8_ETR/U4_TX/SDIO_D2	49	PA14	JTCK
PA15/JTDI/SPI3_NSS/I2S3_WS	48	VCC3.3	
PA14/JTCK/SWCLK	47	GND	
VDD	46	PA13	JTMS
VSS	45	PA12	USB D+
PA13/JTMS/SWDIO	44	PA11	USB D-
PA12/CAN_TX/USBDP/TIM1_ETR	43	PA10	U1_RXD
PA11/CAN_RX/USBDM/TIM1_CH4	42	PA9	U1_TXD
KIN PA10/U1_RX/TIM1_CH3	41	PA8	LED0
H1N PA9/U1_TX/TIM1_CH2	40	PC9	LCD CS
PA8/TIM1_CH1/MCO			
PC9/TIM1_CH1/SDIO_D1			





2. How to create a new project

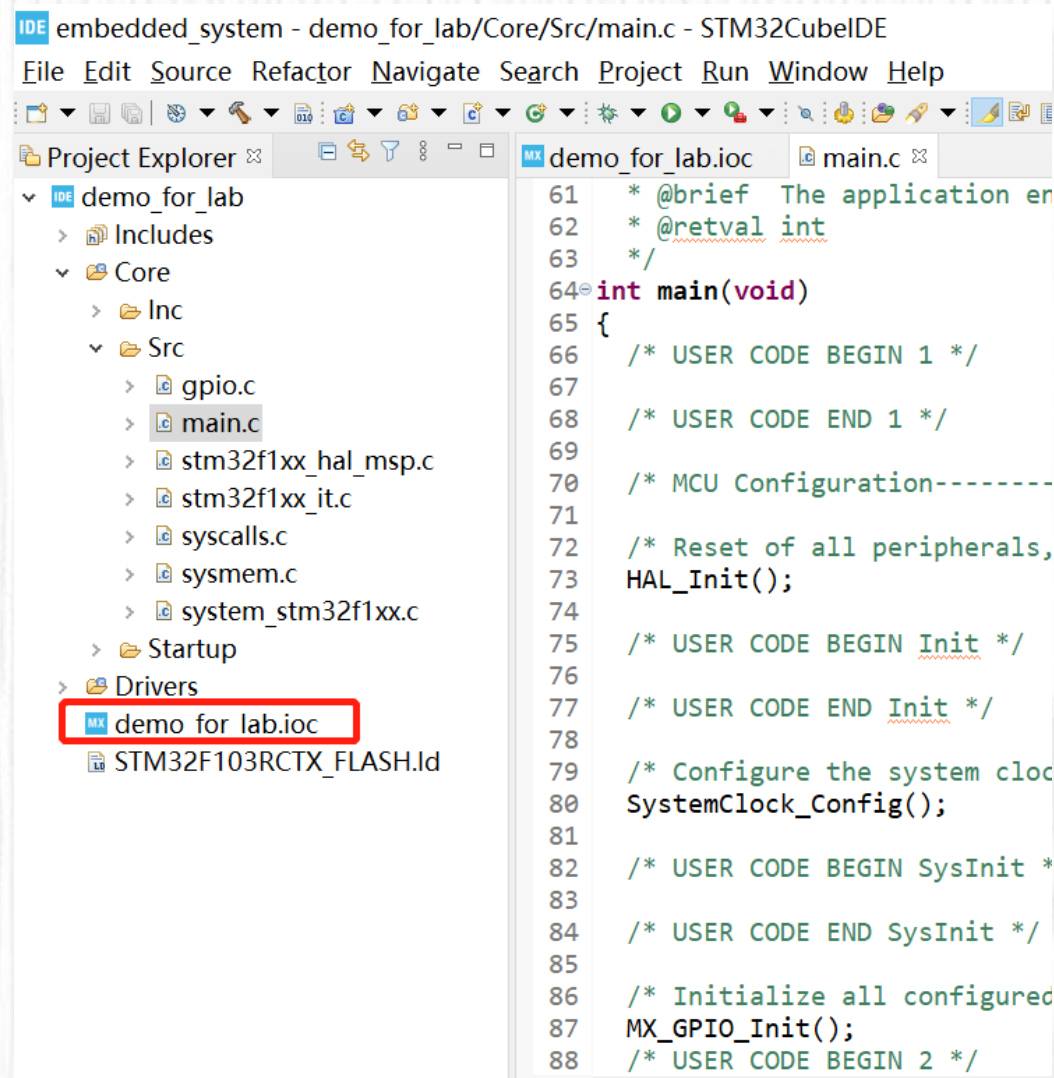
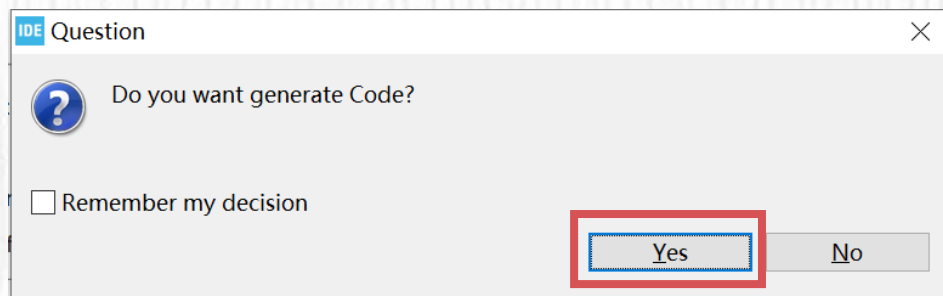
- Project Management
- Change to **Project Manager -> Code Generator**
- Check up ***Generate peripheral initialization as a pair of '.c/.h' files per peripheral***, which will make the codes more modular

Pinout & Configuration	Clock Configuration	Project Manager
Project	STM32Cube MCU packages and embedded software packs	
	<input type="radio"/> Copy all used libraries into the project folder	
	<input checked="" type="radio"/> Copy only the necessary library files	
Code Generator	<input type="radio"/> Add necessary library files as reference in the toolchain project configuration file	
	Generated files	
	<input checked="" type="checkbox"/> Generate peripheral initialization as a pair of '.c/.h' files per peripheral	
	<input type="checkbox"/> Backup previously generated files when re-generating	
	<input checked="" type="checkbox"/> Keep User Code when re-generating	
	<input checked="" type="checkbox"/> Delete previously generated files when not re-generated	
	HAL Settings	
	<input type="checkbox"/> Set all free pins as analog (to optimize the power consumption)	
	<input type="checkbox"/> Enable Full Assert	



2. How to create a new project

- Project Management
- Save the configuration information and start a new project





03

How to programming



3. How to programming

- HAL(Hardware Abstract Layer) library
- Keep updating, bugs will be changed at the next version
- Rapid development, work with cube tool and generate code with one click
- It's convenient to replace the chip and transplant it. You don't have to think about what special registers this chip has. The manufacturer has made it for you
- Learn more about HAL
 - https://bbs.21ic.com/icview-2512392-1-1.html?_dsign=133c8287
 - <https://www.jianshu.com/p/c6809c2bcb4f?from=timeline>



3. How to programming

- STM32CubeIDE has generated codes for us according our configuration. The last thing we need to do is flash the LED
- Remember to put our own codes **into the USER CODE comment block**, otherwise, STM32CubeIDE will overwrite them.

```
int main(void)
{
    /* USER CODE BEGIN 1 */
    /* USER CODE END 1 */

    /* MCU Configuration-----*/

    /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
    HAL_Init();

    /* USER CODE BEGIN Init */
    /* USER CODE END Init */

    /* Configure the system clock */
    SystemClock_Config();

    /* USER CODE BEGIN SysInit */
    /* USER CODE END SysInit */

    /* Initialize all configured peripherals */
    MX_GPIO_Init();
    /* USER CODE BEGIN 2 */
    /* USER CODE END 2 */

    /* Infinite loop */
    /* USER CODE BEGIN WHILE */
    while (1)
    {
        /* USER CODE END WHILE */

        /* USER CODE BEGIN 3 */
        HAL_Delay(1000);
        HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_8);
        HAL_GPIO_TogglePin(GPIOD, GPIO_PIN_2);
        /* USER CODE END 3 */
    }
}
```




3. How to programming

- Functions used frequently

HAL_GPIO_Init

Function name	void HAL_GPIO_Init (GPIO_TypeDef * GPIOx, GPIO_InitTypeDef * GPIO_Init)
Function description	Initializes the GPIOx peripheral according to the specified parameters in the GPIO_Init.
Parameters	<ul style="list-style-type: none">GPIOx: where x can be (A..G depending on device used) to select the GPIO peripheralGPIO_Init: pointer to a GPIO_InitTypeDef structure that contains the configuration information for the specified GPIO peripheral.
Return values	<ul style="list-style-type: none">None:

HAL_GPIO_DeInit

Function name	void HAL_GPIO_DeInit (GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin)
Function description	De-initializes the GPIOx peripheral registers to their default reset values.
Parameters	<ul style="list-style-type: none">GPIOx: where x can be (A..G depending on device used) to select the GPIO peripheralGPIO_Pin: specifies the port bit to be written. This parameter can be one of GPIO_PIN_x where x can be (0..15).
Return values	<ul style="list-style-type: none">None:

HAL_GPIO_ReadPin

Function name	GPIO_PinState HAL_GPIO_ReadPin (GPIO_TypeDef * GPIOx, uint16_t GPIO_Pin)
Function description	Reads the specified input port pin.
Parameters	<ul style="list-style-type: none">GPIOx: where x can be (A..G depending on device used) to select the GPIO peripheralGPIO_Pin: specifies the port bit to read. This parameter can be GPIO_PIN_x where x can be (0..15).
Return values	<ul style="list-style-type: none">The: input port pin value.

HAL_GPIO_WritePin

Function name	void HAL_GPIO_WritePin (GPIO_TypeDef * GPIOx, uint16_t GPIO_Pin, GPIO_PinState PinState)
Function description	Sets or clears the selected data port bit.
Parameters	<ul style="list-style-type: none">GPIOx: where x can be (A..G depending on device used) to select the GPIO peripheralGPIO_Pin: specifies the port bit to be written. This parameter can be one of GPIO_PIN_x where x can be (0..15).PinState: specifies the value to be written to the selected bit. This parameter can be one of the GPIO_PinState enum values:<ul style="list-style-type: none">GPIO_BIT_RESET: to clear the port pinGPIO_BIT_SET: to set the port pin
Return values	<ul style="list-style-type: none">None:
Notes	<ul style="list-style-type: none">This function uses GPIOx_BSRR register to allow atomic read/modify accesses. In this way, there is no risk of an IRQ occurring between the read and the modify access.



3. How to programming

- Functions used frequently

HAL_GPIO_TogglePin

Function name	<code>void HAL_GPIO_TogglePin (GPIO_TypeDef * GPIOx, uint16_t GPIO_Pin)</code>
Function description	Toggles the specified GPIO pin.
Parameters	<ul style="list-style-type: none">GPIOx: where x can be (A..G depending on device used) to select the GPIO peripheralGPIO_Pin: Specifies the pins to be toggled.
Return values	<ul style="list-style-type: none">None:

HAL_GPIO_LockPin

Function name	<code>HAL_StatusTypeDef HAL_GPIO_LockPin (GPIO_TypeDef * GPIOx, uint16_t GPIO_Pin)</code>
Function description	Locks GPIO Pins configuration registers.
Parameters	<ul style="list-style-type: none">GPIOx: where x can be (A..G depending on device used) to select the GPIO peripheralGPIO_Pin: specifies the port bit to be locked. This parameter can be any combination of GPIO_Pin_x where x can be (0..15).
Return values	<ul style="list-style-type: none">None:
Notes	<ul style="list-style-type: none">The locking mechanism allows the IO configuration to be frozen. When the LOCK sequence has been applied on a port bit, it is no longer possible to modify the value of the port bit until the next reset.

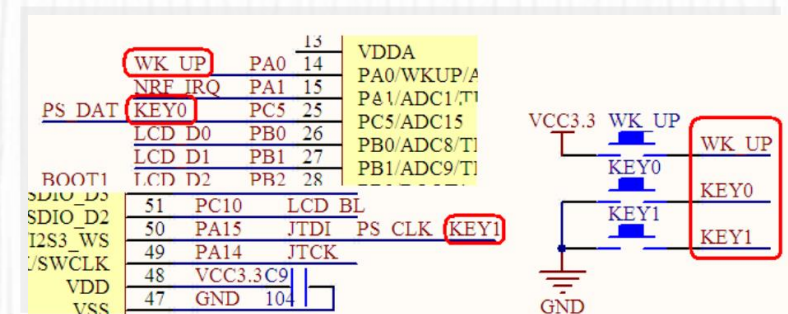
HAL_GPIO_EXTI_IRQHandler

Function name	<code>void HAL_GPIO_EXTI_IRQHandler (uint16_t GPIO_Pin)</code>
Function description	This function handles EXTI interrupt request.
Parameters	<ul style="list-style-type: none">GPIO_Pin: Specifies the pins connected EXTI line
Return values	<ul style="list-style-type: none">None:

HAL_GPIO_EXTI_Callback

Function name	<code>void HAL_GPIO_EXTI_Callback (uint16_t GPIO_Pin)</code>
Function description	EXTI line detection callbacks.

- On the ALIENTEK MiniSTM32 board, use KEY0 and KEY1 to control two LED LED0 and LED1 individually



3. How to programming

- GPIO Configuration

- Find the pins connected to KEY0, KEY1, LED0 and LED1, which are PC5, PA15, PA8 and PD2

GPIO Mode and Configuration

Configuration

Group By Peripherals

☒ GPIO ☒ RCC ☒ SYS

Search Signals
Search (Ctrl+F) ☐ Show only Modified Pins

Pin ...	Signal o...	GPIO...	GPIO mode	GPIO ...	Maximu...	User L...	Modified
PA8	n/a	High	Output Push Pull	No pull...	Low	LED0	<input checked="" type="checkbox"/>
PA15	n/a	n/a	Input mode	Pull-up	n/a	KEY 1	<input checked="" type="checkbox"/>
PC5	n/a	n/a	Input mode	Pull-up	n/a	KEY 0	<input checked="" type="checkbox"/>
PD2	n/a	High	Output Push Pull	No pull...	Low	LED1	<input checked="" type="checkbox"/>

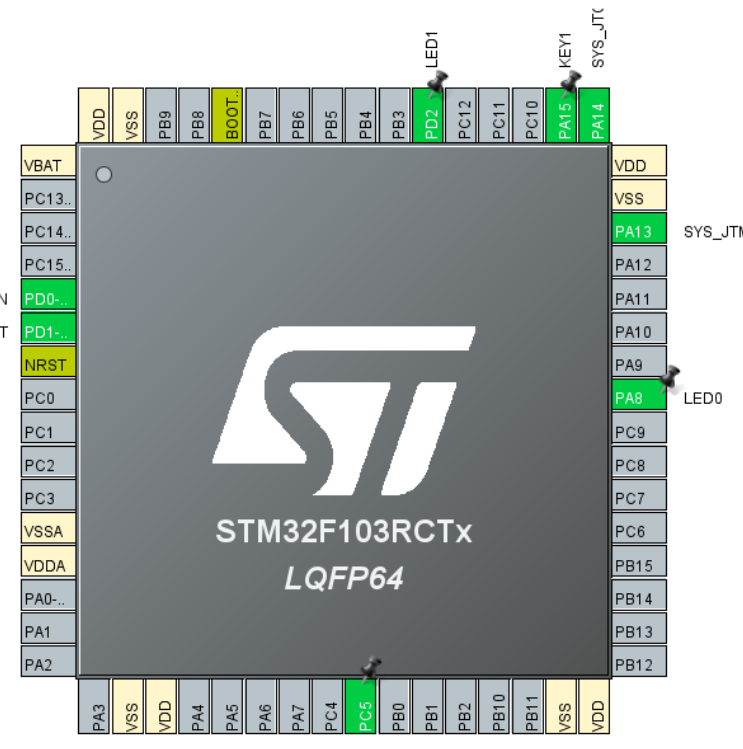
PD2 Configuration :

GPIO output level

GPIO mode

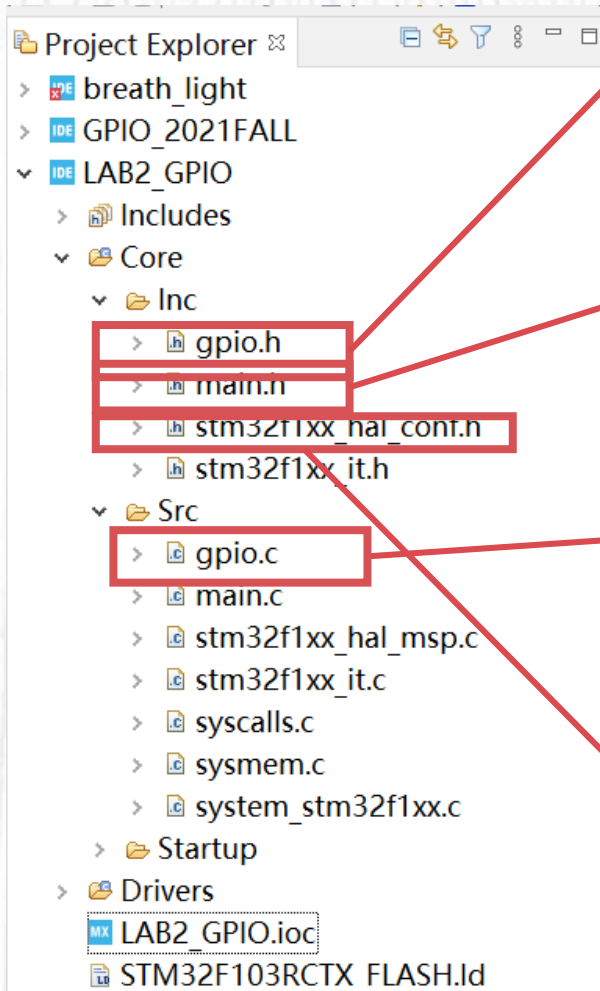
Pinout view

System view



3. How to programming

- The generated files and codes



```
void MX_GPIO_Init(void);
```

```
#define KEY0_Pin GPIO_PIN_5
#define KEY0_GPIO_Port GPIOC
#define LED0_Pin GPIO_PIN_8
#define LED0_GPIO_Port GPIOA
#define KEY1_Pin GPIO_PIN_15
#define KEY1_GPIO_Port GPIOA
#define LED1_Pin GPIO_PIN_2
#define LED1_GPIO_Port GPIOD
```

```
stm32f1xx_hal_conf.h
Legacy/stm32f1xx_hal_conf.h
stm32f1xx_hal_adc.h
stm32f1xx_hal_can.h
stm32f1xx_hal_cec.h
stm32f1xx_hal_cortex.h
stm32f1xx_hal_crc.h
stm32f1xx_hal_dac.h
stm32f1xx_hal_dma.h
stm32f1xx_hal_eth.h
stm32f1xx_hal_exti.h
stm32f1xx_hal_flash.h
stm32f1xx_hal_gpio.h
```

```
void MX_GPIO_Init(void){
    GPIO_InitTypeDef GPIO_InitStructure = {0};
    /* GPIO Ports Clock Enable */
    __HAL_RCC_GPIOC_CLK_ENABLE();
    __HAL_RCC_GPIOA_CLK_ENABLE();

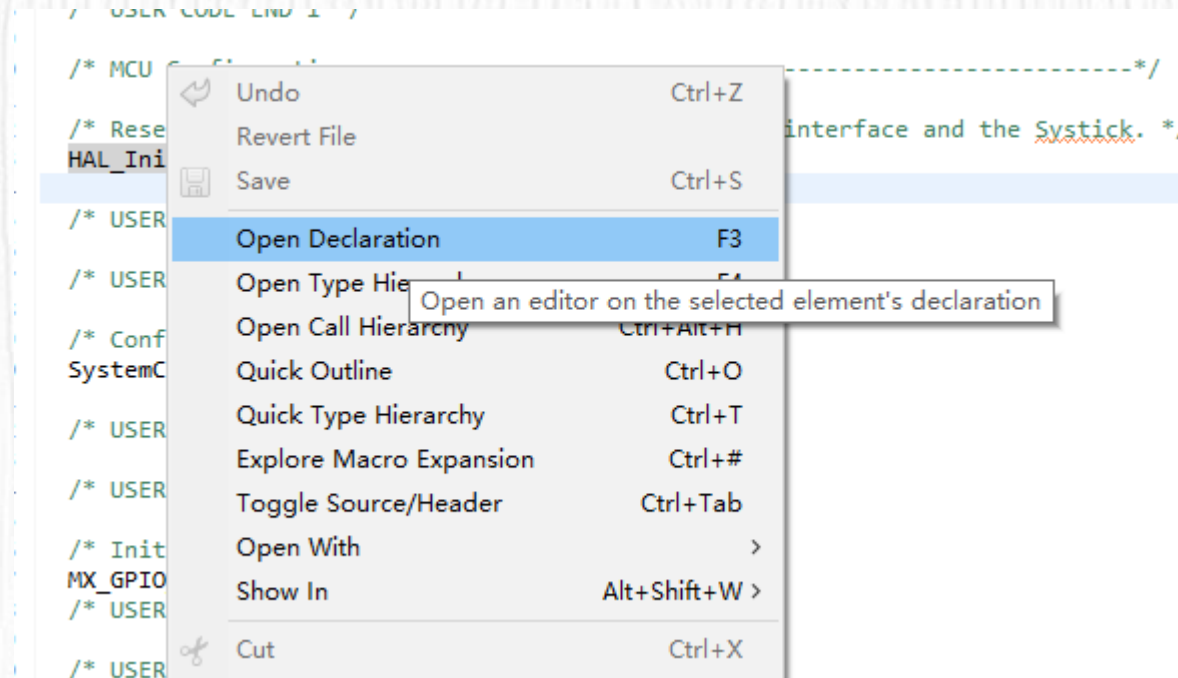
    /*Configure GPIO pin Output Level */
    HAL_GPIO_WritePin(LED0_GPIO_Port, LED0_Pin, GPIO_PIN_SET);
    /*Configure GPIO pin Output Level */
    HAL_GPIO_WritePin(LED1_GPIO_Port, LED1_Pin, GPIO_PIN_SET);

    /*Configure GPIO pin : PtPin */
    GPIO_InitStructure.Pin = KEY0_Pin;
    GPIO_InitStructure.Mode = GPIO_MODE_INPUT;
    GPIO_InitStructure.Pull = GPIO_PULLUP;
    HAL_GPIO_Init(KEY0_GPIO_Port, &GPIO_InitStructure);
    /*Configure GPIO pin : PtPin */
    GPIO_InitStructure.Pin = LED0_Pin;
    GPIO_InitStructure.Mode = GPIO_MODE_OUTPUT_PP;
    GPIO_InitStructure.Pull = GPIO_NOPULL;
    GPIO_InitStructure.Speed = GPIO_SPEED_FREQ_LOW;
    HAL_GPIO_Init(LED0_GPIO_Port, &GPIO_InitStructure);
    /*Configure GPIO pin : PtPin */
    GPIO_InitStructure.Pin = KEY1_Pin;
    GPIO_InitStructure.Mode = GPIO_MODE_INPUT;
    GPIO_InitStructure.Pull = GPIO_PULLUP;
    HAL_GPIO_Init(KEY1_GPIO_Port, &GPIO_InitStructure);
    /*Configure GPIO pin : PtPin */
    GPIO_InitStructure.Pin = LED1_Pin;
    GPIO_InitStructure.Mode = GPIO_MODE_OUTPUT_PP;
    GPIO_InitStructure.Pull = GPIO_NOPULL;
    GPIO_InitStructure.Speed = GPIO_SPEED_FREQ_LOW;
    HAL_GPIO_Init(LED1_GPIO_Port, &GPIO_InitStructure);
}
```




3. How to programming

- Use right click menu **“Open Declaration”** to check the definition of functions or structures





3. How to programming

- The structure used frequently

■ stm32f1xx_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_define */

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

    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_define */

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



3. How to programming

- Add our codes in main.c

```
while (1)
{
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
    if (HAL_GPIO_ReadPin(KEY0_GPIO_Port, KEY0_Pin) == GPIO_PIN_RESET) {
        HAL_Delay(100);
        HAL_GPIO_TogglePin(LED0_GPIO_Port, LED0_Pin);
    }
    else
        HAL_GPIO_WritePin(LED0_GPIO_Port, LED0_Pin, HAL_GPIO_ReadPin(LED0_GPIO_Port, LED0_Pin));

    if (HAL_GPIO_ReadPin(KEY1_GPIO_Port, KEY1_Pin) == GPIO_PIN_RESET) {
        HAL_Delay(100);
        HAL_GPIO_TogglePin(LED1_GPIO_Port, LED1_Pin);
    }
    //else
        //HAL_GPIO_WritePin(LED1_GPIO_Port, LED1_Pin, HAL_GPIO_ReadPin(LED1_GPIO_Port, LED1_Pin));
}
/* USER CODE END 3 */
```



04

Practice

4. Practice



- Run the demo on MiniSTM32 board