On-chip Flash

This tutorial demonstrates how to control the **internal Flash** of the development board to read and write data through **keys** , and print the data through the serial port (USART1)

1、Software-Hardware

- **STM32F103CubeIDE**
- **STM32 robot expansion board**

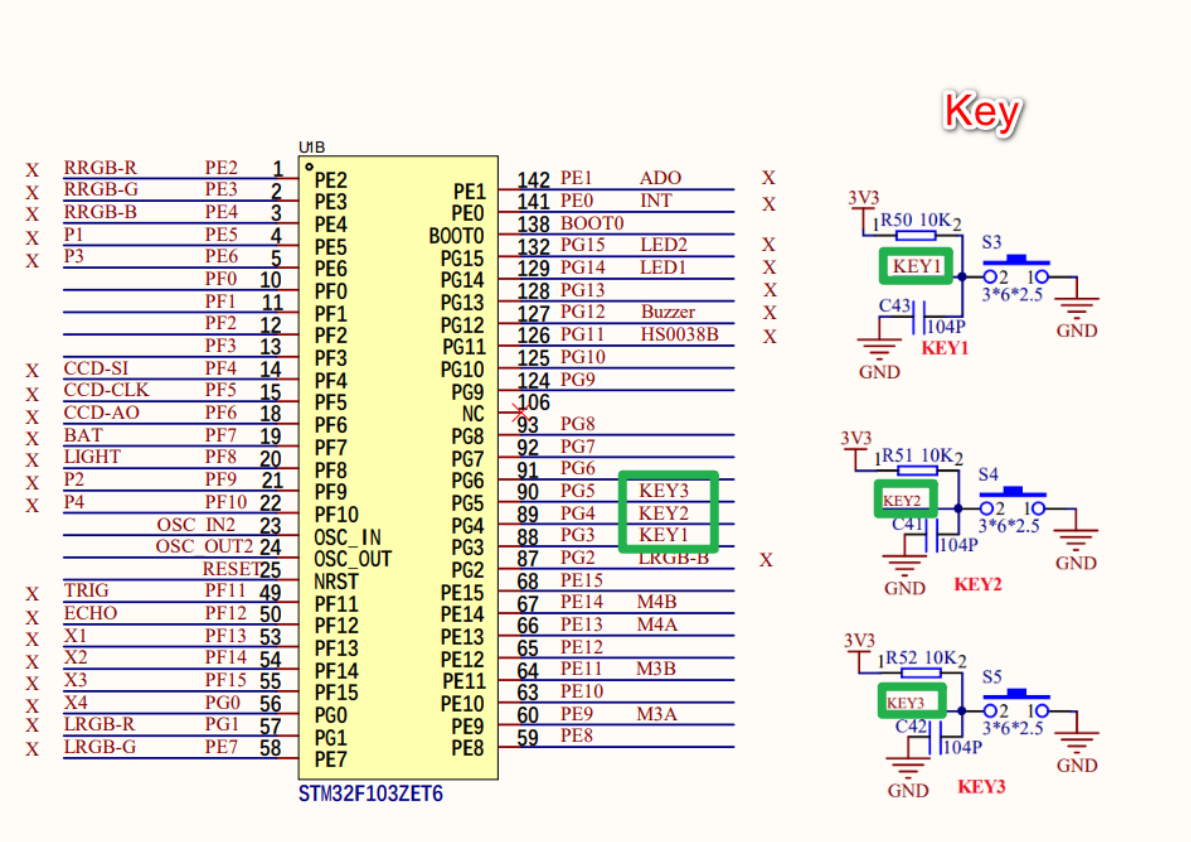
  USART, internal Flash: chip internal peripherals

  Key: Onboard
- **Type-C cable or ST-Link**

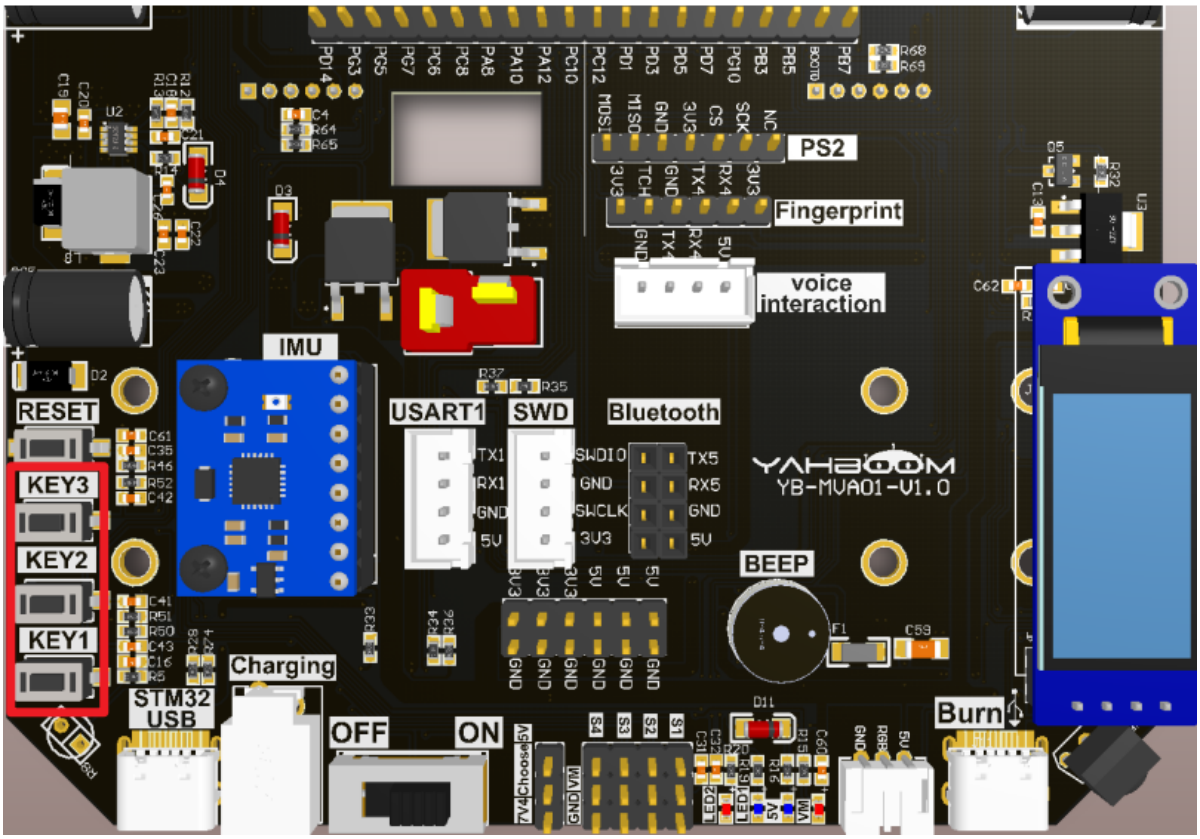
  Download or simulate the program of the development board

2、Brief principle

2.1、Hardware schematic diagram



2.2、Physical connection diagram



2.3、Principle of control

Use the keys to control the data written and read in flash, and print the data out through the serial port.

- **GPIO reading**

By defining different functions for the key, the data read and write operation in flash is realized.

KEY (Schematic name)	Control pin	Function
KEY1	PG3	Add 1 to the count
KEY2	PG4	The count is cleared to zero
KEY3	PG5	Reading count values

- **Flash**

STM32F103ZET6 Flash memory capacity is 512K bytes, a total of 256 2K byte pages;

Flash memory is mainly used to store program code and constant data in order to retain these data when restarted after power failure.

Programming and erasing flash memory	register	Function
The FPEC key register	FLASH_KEYR	Used to erase and program Flash memory unlocking and locking
Select the byte key register	FLASH_OPTKEYR	Option bytes used to unlock and lock the Flash memory
Flash memory control registers	FLASH_CR	Used to control Flash memory programming and erase operations
Flash memory status register	FLASH_SR	Used to report Flash memory status information
Flash address register	FLASH_AR	Used to specify the address of Flash memory to be programmed or erased
Select the byte register	FLASH_OBR	The value used to store the option bytes
Write the protected register	FLASH_WRPR	Used to specify that Flash memory sectors are write-protected to prevent accidental modification.

```
For detailed information, refer to "PM0042_STM32F10xxx Flash Memory Programming (Chinese)".
```

- **Flsah write**

  Unlock → erase → write data → lock

operation	function
Unlock	HAL_FLASH_Unlock
erase	HAL_FLASHEx_Erase
write data	HAL_FLASH_Program
lock	HAL_FLASH_Lock

3、Engineering configuration

Project Configuration: Prompts for configuration options in the STM32CubeIDE project configuration process

3.1、Notes

Omitted project configuration: **New project, chip selection, project configuration, SYS for pin configuration, RCC configuration, clock configuration, and project configuration** content

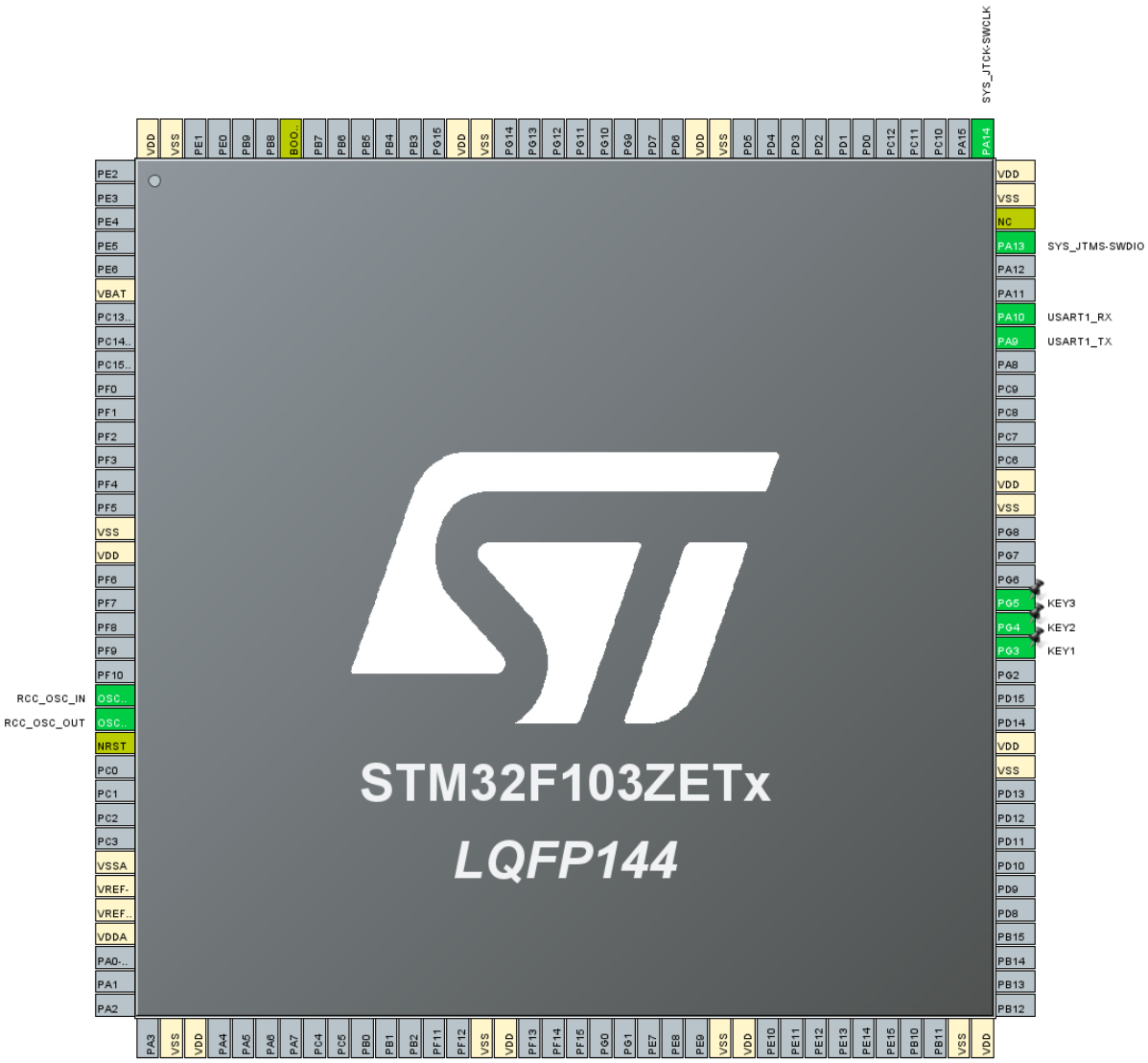
The project configuration part, which is not omitted, is the key point to configure in this tutorial.

```
Please refer to [2. Development environment construction and use: STM32CubeIDE
installation and use] to understand how to configure the omitted parts of the
project.
```

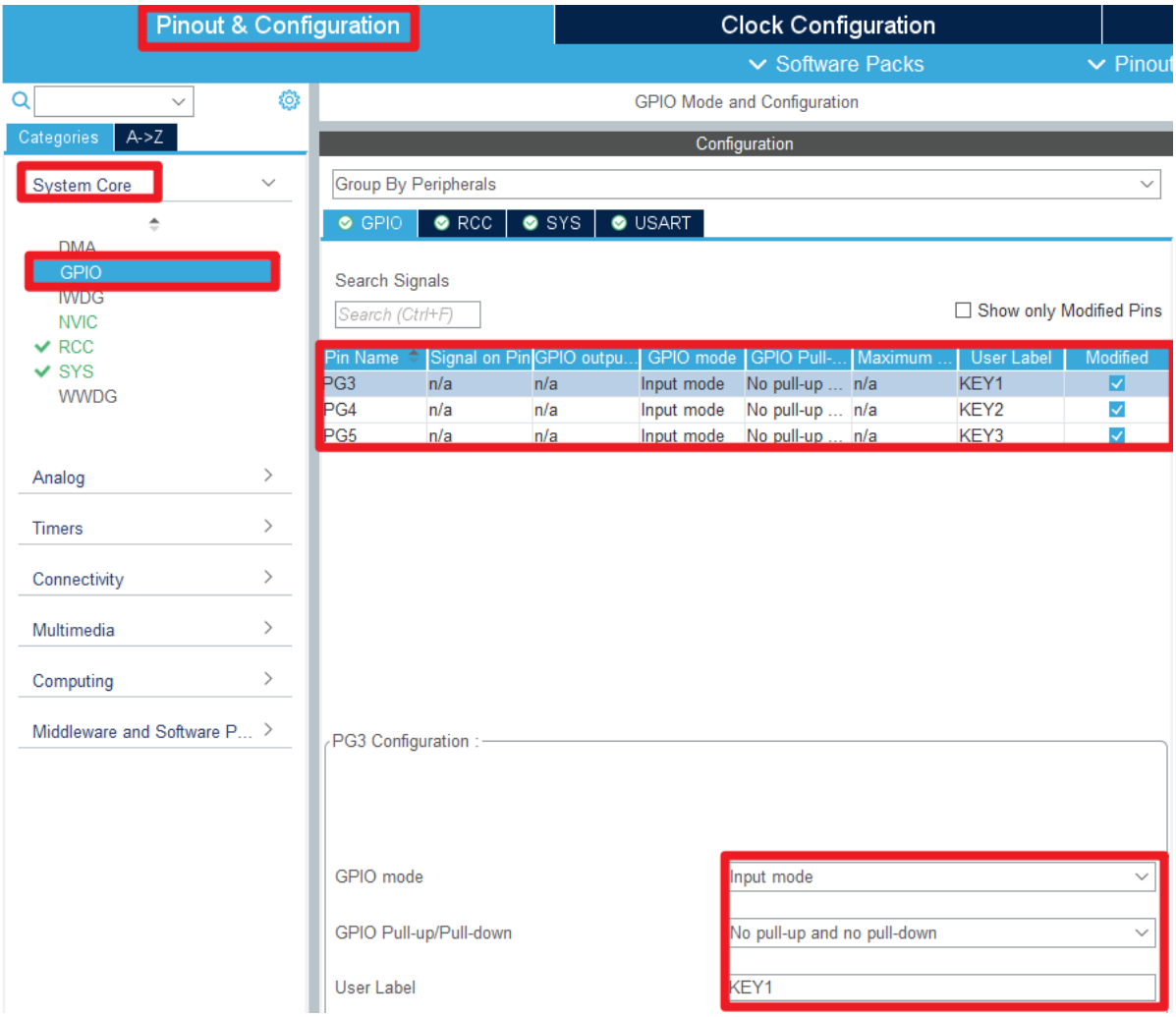
3.2、Pin configuration

- **Configure the specified pin function**

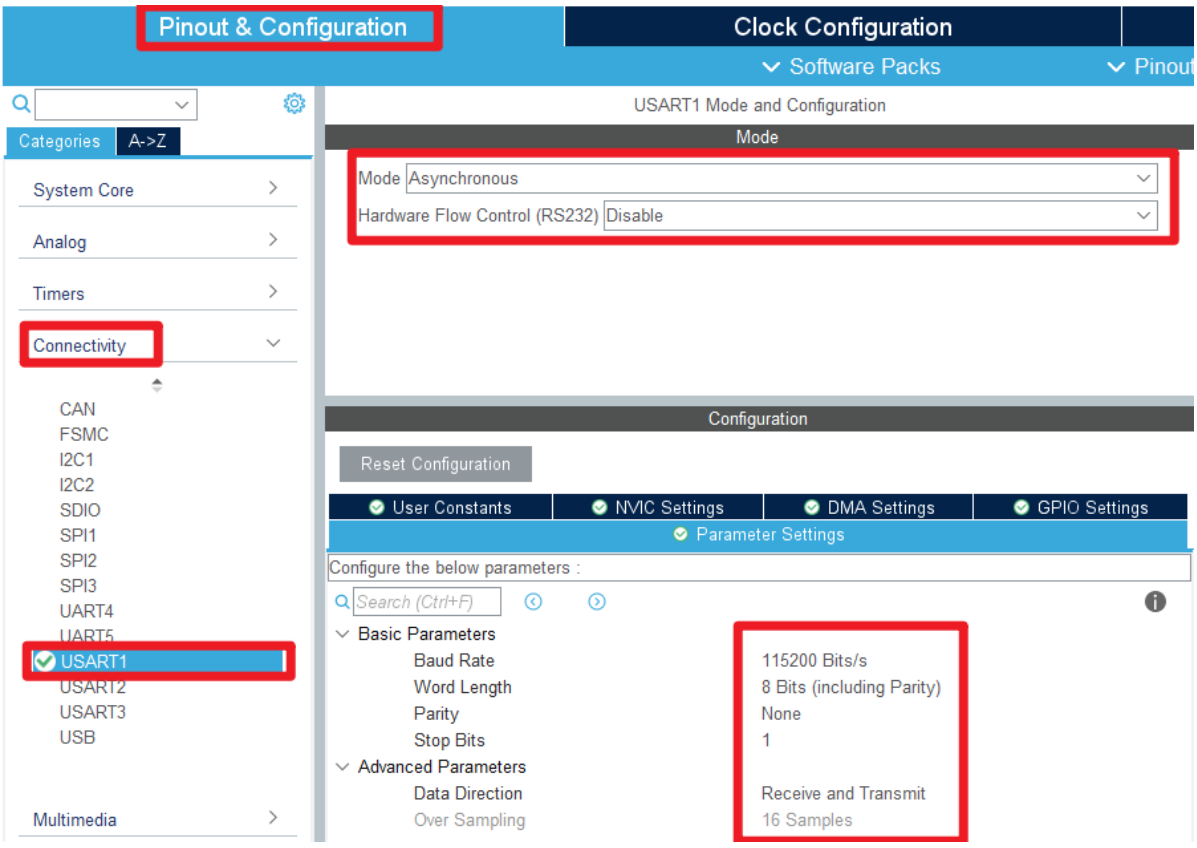
You can directly select the corresponding pin number in the pin view, and the corresponding option will appear when the mouse is left clicked

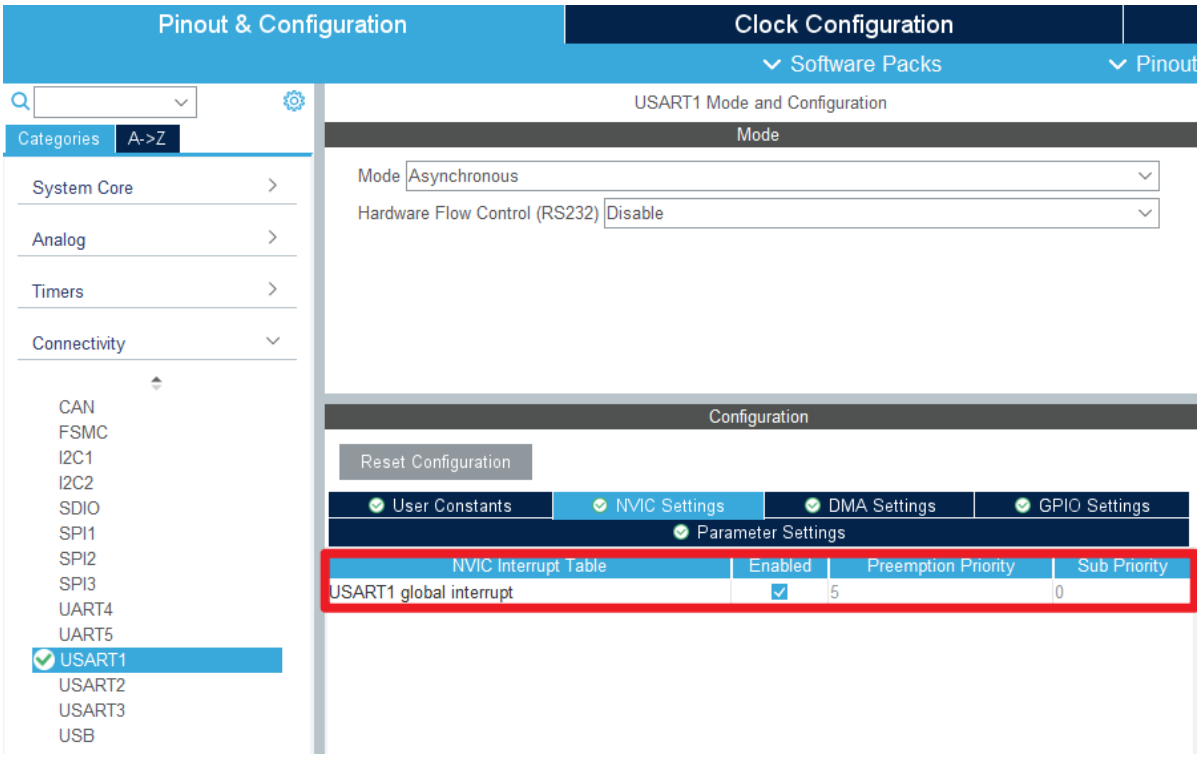


- **GPIO**


Pinout & Configuration
Clock Configuration
Software Packs
Pinout
GPIO Mode and Configuration
Categories
A->Z
System Core
DMA
GPIO
IWDG
NVIC
RCC
SYS
WWDG
Analog
Timers
Connectivity
Multimedia
Computing
Middleware and Software P...
Configuration
Group By Peripherals
GPIO
RCC
SYS
USART
Search Signals
Search (Ctrl+F)
Show only Modified Pins
Pin Name
Signal on Pin
GPIO outpu...
GPIO mode
GPIO Pull-...
Maximum ...
User Label
Modified
PG3
n/a
n/a
Input mode
No pull-up ...
n/a
KEY1
PG4
n/a
n/a
Input mode
No pull-up ...
n/a
KEY2
PG5
n/a
n/a
Input mode
No pull-up ...
n/a
KEY3
PG3 Configuration :
GPIO mode
Input mode
GPIO Pull-up/Pull-down
No pull-up and no pull-down
User Label
KEY1

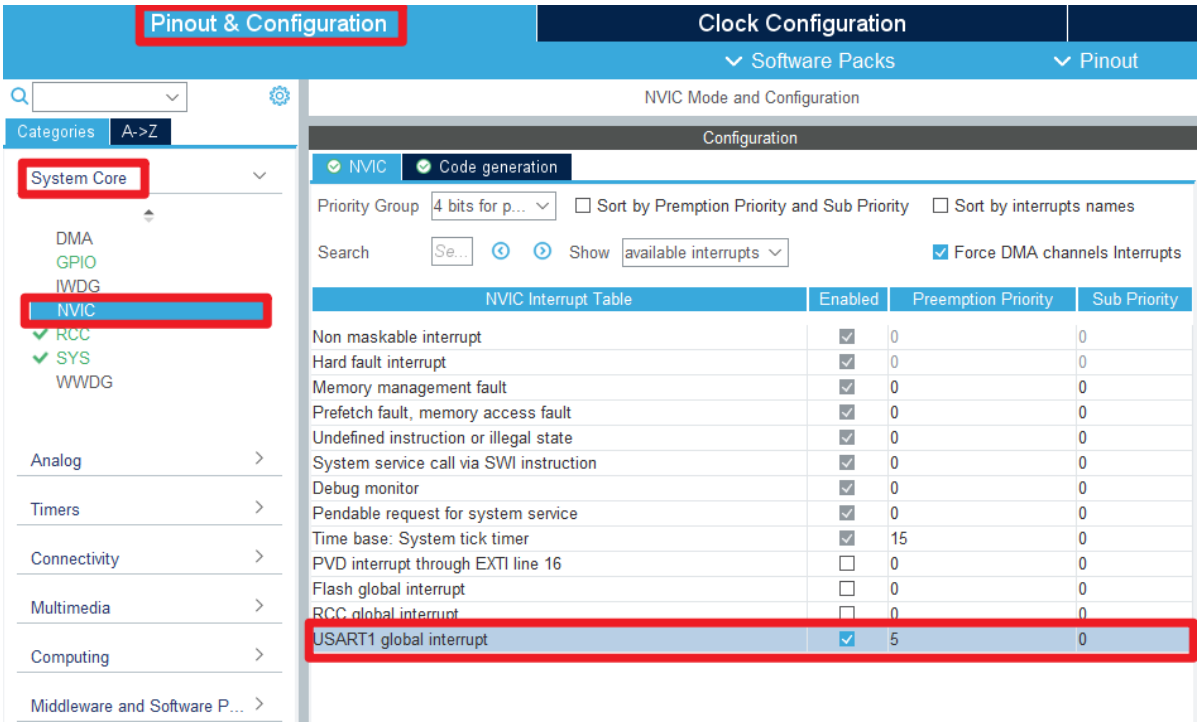
- **USART**


Pinout & Configuration
Clock Configuration
Software Packs
Pinout
USART1 Mode and Configuration
Categories
A->Z
System Core
Analog
Timers
Connectivity
CAN
FSMC
I2C1
I2C2
SDIO
SPI1
SPI2
SPI3
UART4
UART5
USART1
USART2
USART3
USB
Multimedia
Mode
Mode
Asynchronous
Hardware Flow Control (RS232)
Disable
Configuration
Reset Configuration
User Constants
NVIC Settings
DMA Settings
GPIO Settings
Parameter Settings
Configure the below parameters :
Search (Ctrl+F)
Basic Parameters
Baud Rate
Word Length
Parity
Stop Bits
Advanced Parameters
Data Direction
Over Sampling
115200 Bits/s
8 Bits (including Parity)
None
1
Receive and Transmit
16 Samples

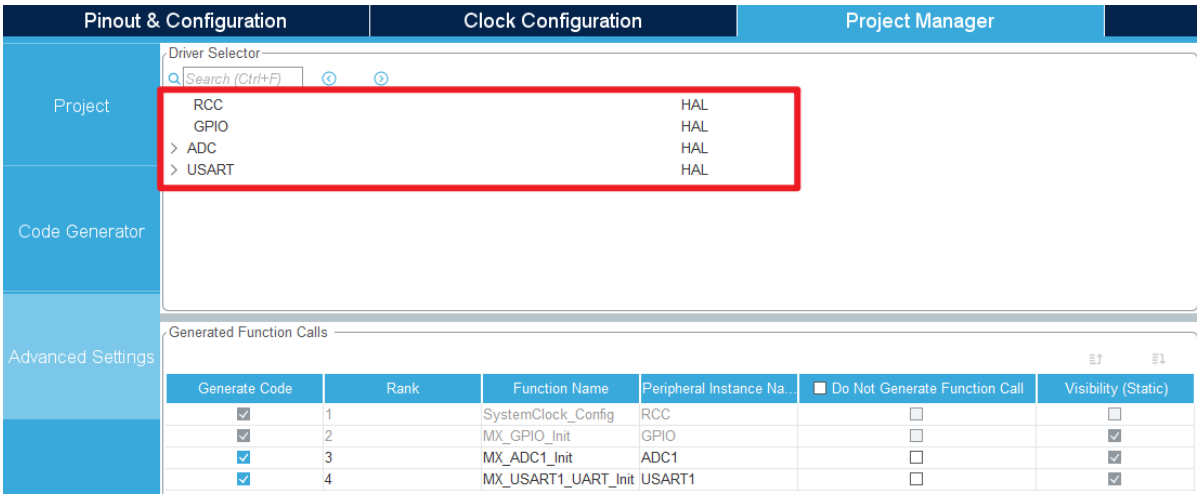


- **NVIC**

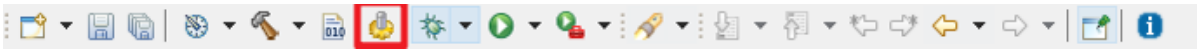
  NVIC can modify interrupt priority



- **Advanced Settings**



- **Generating code**



4、Main function

This section mainly introduces the functional code written by users. **Detailed code can be opened by yourself in the project file we provide, and enter the Bsp folder to view the source code.**

4.1、User function

> **function：STMFLASH_ReadHalfWord**

Function prototypes	**u16 STMFLASH_ReadHalfWord(u32 faddr)**
Functional Description	Read a half-word (16-bit data) at the specified address.
Input parameters	**faddr**：address
Return value	Data for the address

> **function：STMFLASH_Write_NoCheck**

Function prototypes	**void STMFLASH_Write_NoCheck(u32 WriteAddr,u16 *pBuffer,u16 NumToWrite)**
Functional Description	Writes data of the specified length from the specified address without checking
Input parameters1	**WriteAddr**：Start address
Input parameters2	**pBuffer**：Data pointer
Input parameters3	**NumToWrite**：Half-word (16 bit) number
Return value	None

> **function：STMFLASH_Write**

Function prototypes	void STMFLASH_Write(u32 WriteAddr,u16 *pBuffer,u16 NumToWrite)
Functional Description	Writes data of the specified length starting at the specified address
Input parameters1	**WriteAddr**：Start address
Input parameters2	**pBuffer**：Data pointer
Input parameters3	**NumToWrite**：Half-word (16 bit) number
Return value	None

> **function：STMFLASH_Read**

Function prototypes	void STMFLASH_Read(u32 ReadAddr,u16 *pBuffer,u16 NumToRead)
Functional Description	Read data of the specified length starting at the specified address
Input parameters1	**ReadAddr**：Start address
Input parameters2	**pBuffer**：Data pointer
Input parameters3	**NumToRead**：Half-word (16 bit) number
Return value	None

4.2、HAL library function parsing

The HAL library functions that were covered in the previous tutorial will not be covered

```
If you want to find the HAL library and LL library function analysis involved in the entire tutorial, you can view the documents in the folder [8. STM32 Manual: STM32F1_HAL Library and LL Library_User Manual]
```

> **function：HAL_FLASH_Unlock**

Function prototypes	HAL_StatusTypeDef HAL_FLASH_Unlock(void)
Functional Description	Unlock access to the flash control register
Input parameters	None
Return value	**HAL status value**：HAL_OK、HAL_ERROR、HAL_BUSY、HAL_TIMEOUT

> **function：HAL_FLASH_Lock**

Function prototypes	**HAL_StatusTypeDef HAL_FLASH_Lock(void)**
Functional Description	Lock flash control register access
Input parameters	None
Return value	**HAL status value**：HAL_OK、HAL_ERROR、HAL_BUSY、HAL_TIMEOUT

> **function：HAL_FLASH_Program**

Function prototypes	**HAL_StatusTypeDef HAL_FLASH_Program (uint32_t TypeProgram, uint32_t Address, uint64_t Data)**
Functional Description	Used for writing data to Flash
Input parameters1	**TypeProgram**：Type of data written: half-word, byte, or double-word
Input parameters2	**Address**：The address where Flash writes data
Input parameters3	**Data**：To write data
Return value	**HAL status value**：HAL_OK、HAL_ERROR、HAL_BUSY、HAL_TIMEOUT

> **function：HAL_FLASHEx_Erase**

Function prototypes	**HAL_StatusTypeDef HAL_FLASHEx_Erase (FLASH_EraseInitTypeDef *pEraseInit, uint32_t *PageError)**
Functional Description	A pointer to a FLASH_EraseInitTypeDef structure used to erase a sector in Flash memory
Input parameters1	**pEraseInit**：Configure the parameters for the erase operation
Input parameters2	**PageError**：Used to get the sector number of the error in the erase operation
Return value	**HAL status value**：HAL_OK、HAL_ERROR、HAL_BUSY、HAL_TIMEOUT

5、Experimental phenomenon

After downloading the program successfully, press the RESET button of the development board to open the serial debugging assistant to observe the phenomenon

```
For program download, please refer to [2. Development environment construction and use: program download and simulation]
```

phenomenon：

Press KEY1: The serial debugging assistant will output the prompt of adding 1 to the count;

Press KEY2: The serial debugging assistant will output the prompt of zero count;

Press KEY3: Serial debugging assistant will read the current count value;

After the development board power off, you can also read the previous count value.