Serial communication

This tutorial demonstrates: sending and printing serial port information through **serial port (USART1) interrupt**.

1. Software-Hardware

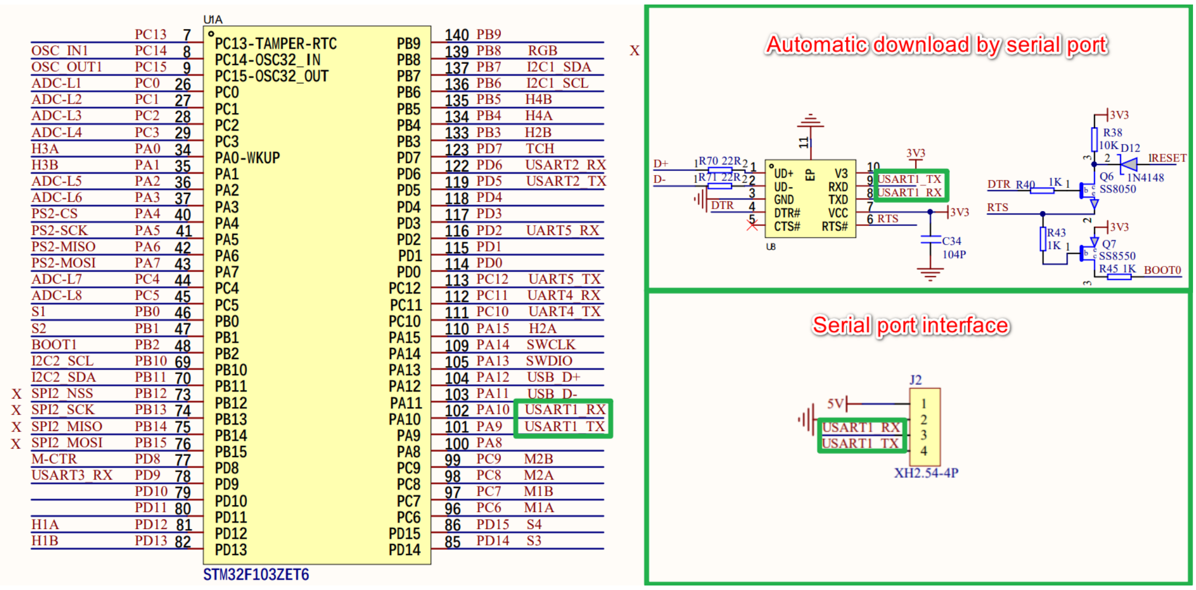
- **STM32F103CubeIDE**
- **STM32 Robot Development Board**

  USART: chip internal peripherals
- **Type-C data cable or ST-Link**

  Download programs or simulate the development board

2. Brief principle

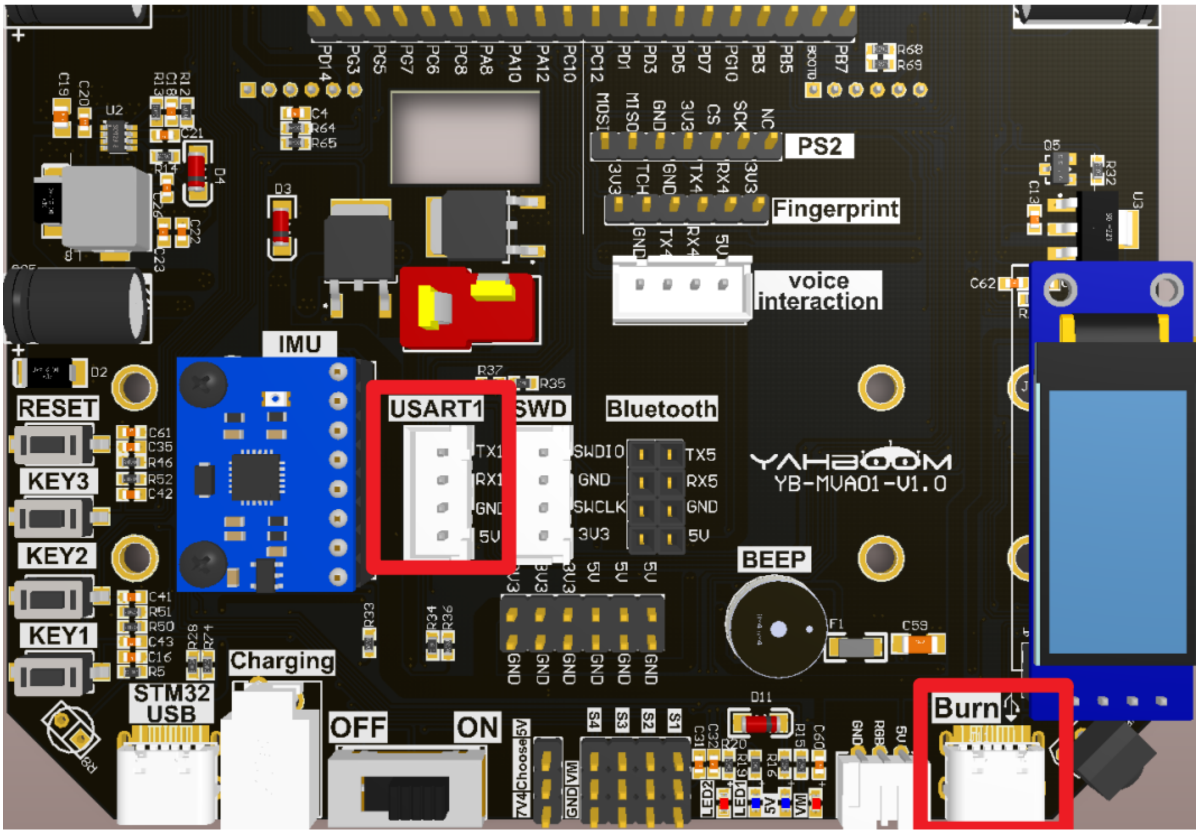
1. Hardware schematic diagram



2. Physical connection diagram

USART1 on the development board leads to two interfaces: Type-C and USART1;

It is recommended to use the Type-C burning interface for testing and sending and receiving serial port information.



USART (schematic name)	pin	Function
USART1_TX	PA9	USART1 sends data
USART1_RX	PA10	USART1 receives data

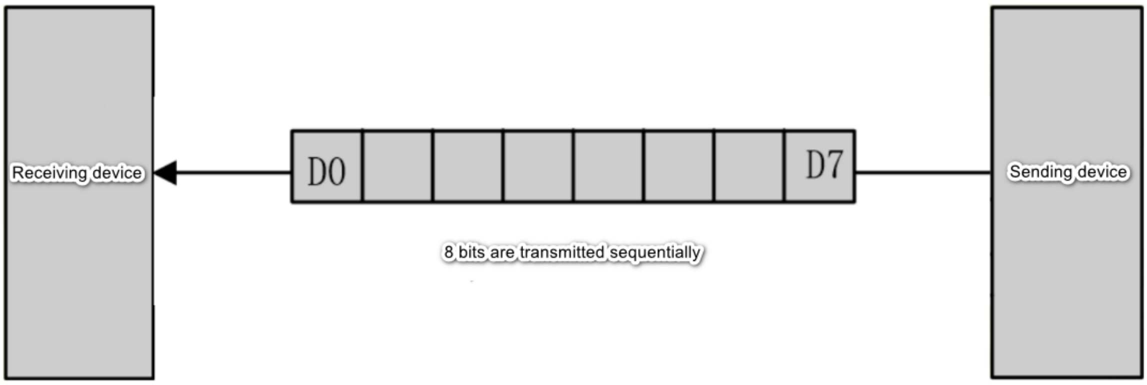
3. Control principle

Send data to the serial port (USART1) through the serial port debugging assistant to trigger the serial port interrupt, and print out the data through the serial port interrupt callback function.

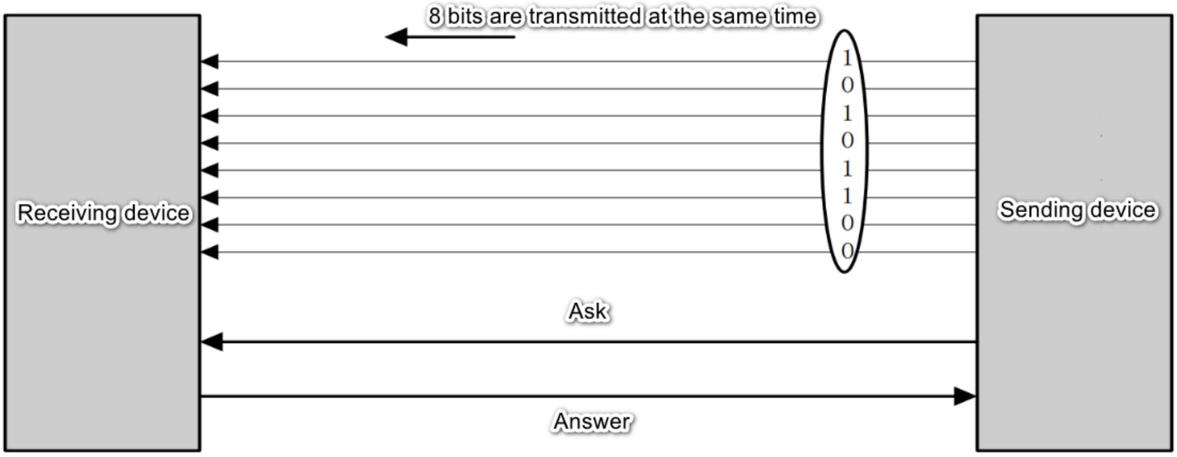
- **Computer Communications**

  Classified by data transmission method: serial communication and parallel communication

  - **Serial Communication**: Data is transferred bit by bit



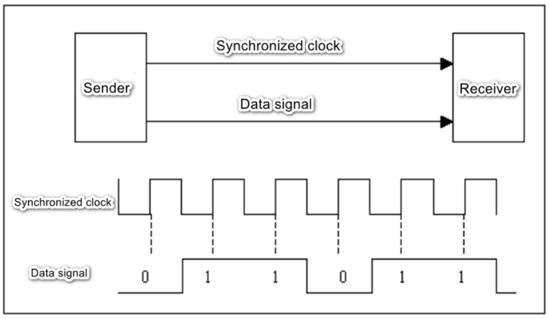
- **Parallel communication**: Multiple bits of data are transmitted simultaneously



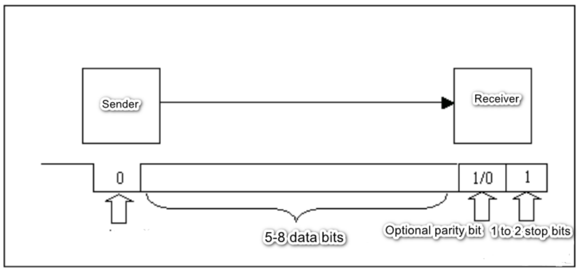
- **Serial Communication**
  - **Serial communication data synchronization mode**: synchronous serial communication and asynchronous serial communication

  Synchronous serial communication: The sender and receiver rely on independent clock lines to synchronize signals.

  Asynchronous serial communication: The sending and receiving parties complete communication according to the agreed character format and communication rate.

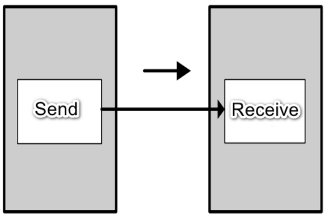


Synchronous communication

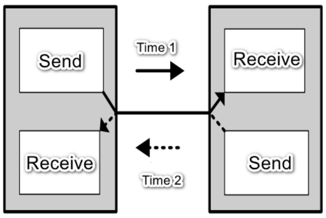


Asynchronous communication

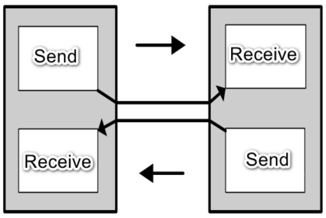
  - **Serial communication data transmission direction**: single duplex, half duplex and full duplex.



Single work



Half duplex



Full duplex

Simplex: Data can only be transmitted in one direction, and cannot be sent and received at the same time;

Half-duplex: Data is transmitted in both directions, but cannot be sent and received at the same time;

Full Duplex: Data can be transmitted in both directions at the same time, and can be sent and received at the same time.

- **USART**(Universal Synchronous/Asynchronous Receiver/Transmitter)

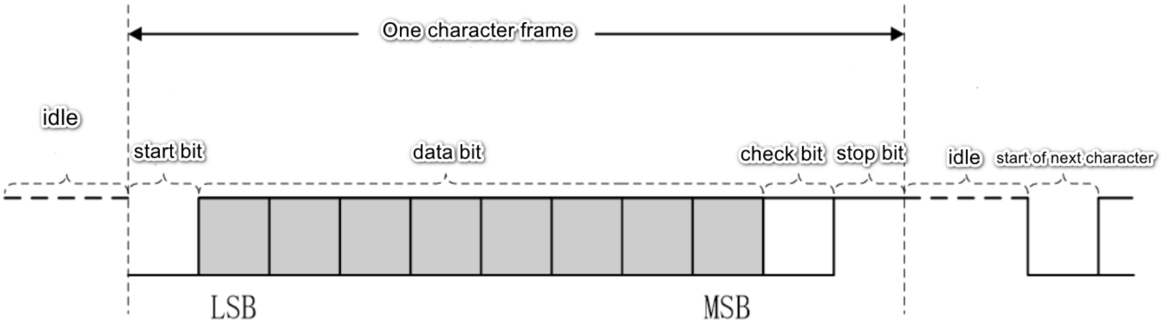
USART supports synchronous and asynchronous communication. Our tutorial uses UART (Universal Serial Asynchronous Receiver-Transmitter).

- **Asynchronous serial communication**
  - **Communication Rate**

    Baud rate: The number of binary digits transmitted per second, in bit/s (bps).

```
Commonly used baud rates: 9600, 57600 and 115200
Baud rate 115200: 115200 bits transmitted per second
```

- **Character format**



```
When transmitting data, the low bit comes first and the high bit comes last.
```

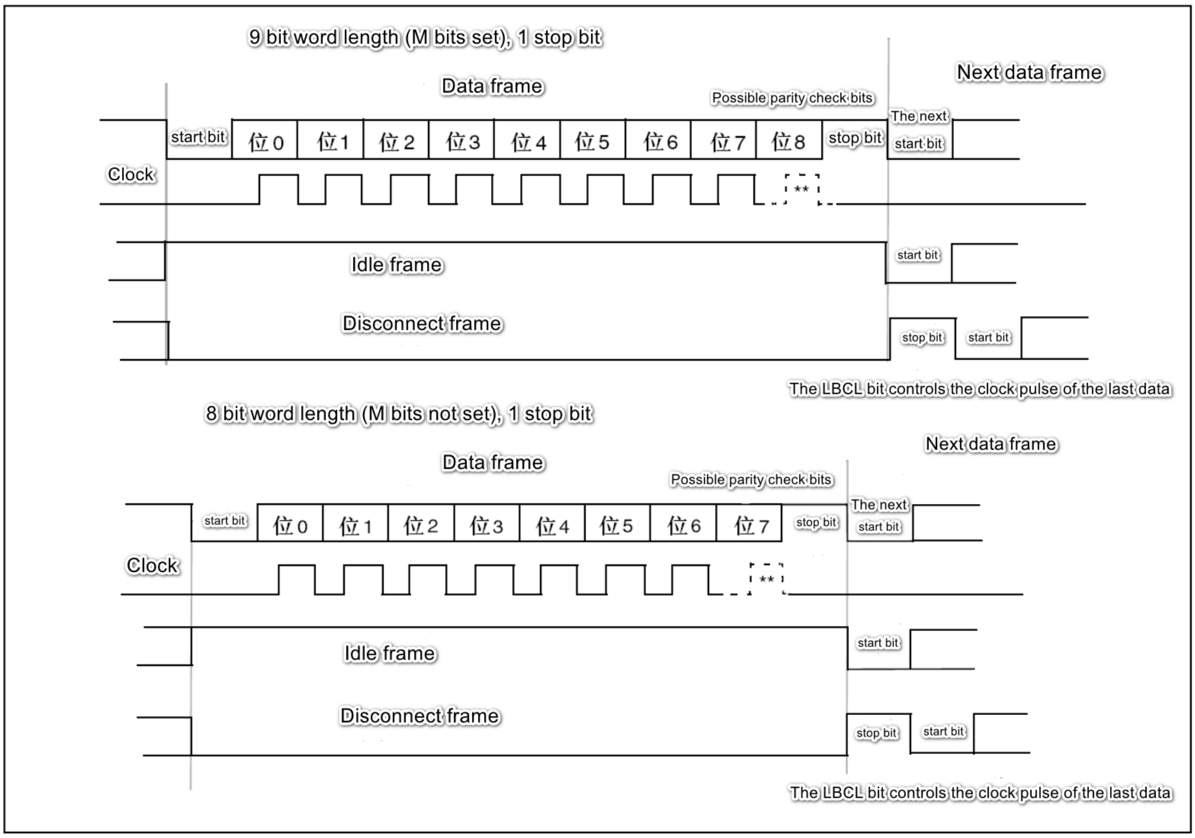
Data frame format: **Start bit + data bit + check bit + stop bit**

Start bit (beginning of data frame): 1 bit

Data bits: 5~8 bits

Check digit: odd or even parity (whether the number of "1"s in the data is odd or even)

Stop bit (end of data frame): 0.5, 1.5, 2 stop bits



- **USART two-way communication**

USART (development board)	**Peripheral/USB to TTL module**
TXD	RXD
RXD	TXD
GND	GND
VCC	VCC
Precautions	1.TXD (transmitting line): pin used to transmit data 2.RXD (receiving line): pin used to receive data 3.GND (ground wire): used to connect to ground pins to ensure that the system level is consistent; 4.VCC (power line): If the peripherals do not have independent power supply, the VCC pin needs to be correctly connected to the appropriate power supply voltage; 5.USART two-way communication requires at least two pins: TXD (transmitting line) and RXD (receiving line).

Common serial port character format configuration: 1 start bit, 8 data bits, no parity, 1 stop bit

3. Project configuration

Project configuration: Prompt configuration options during STM32CubeIDE project configuration

1. Description

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

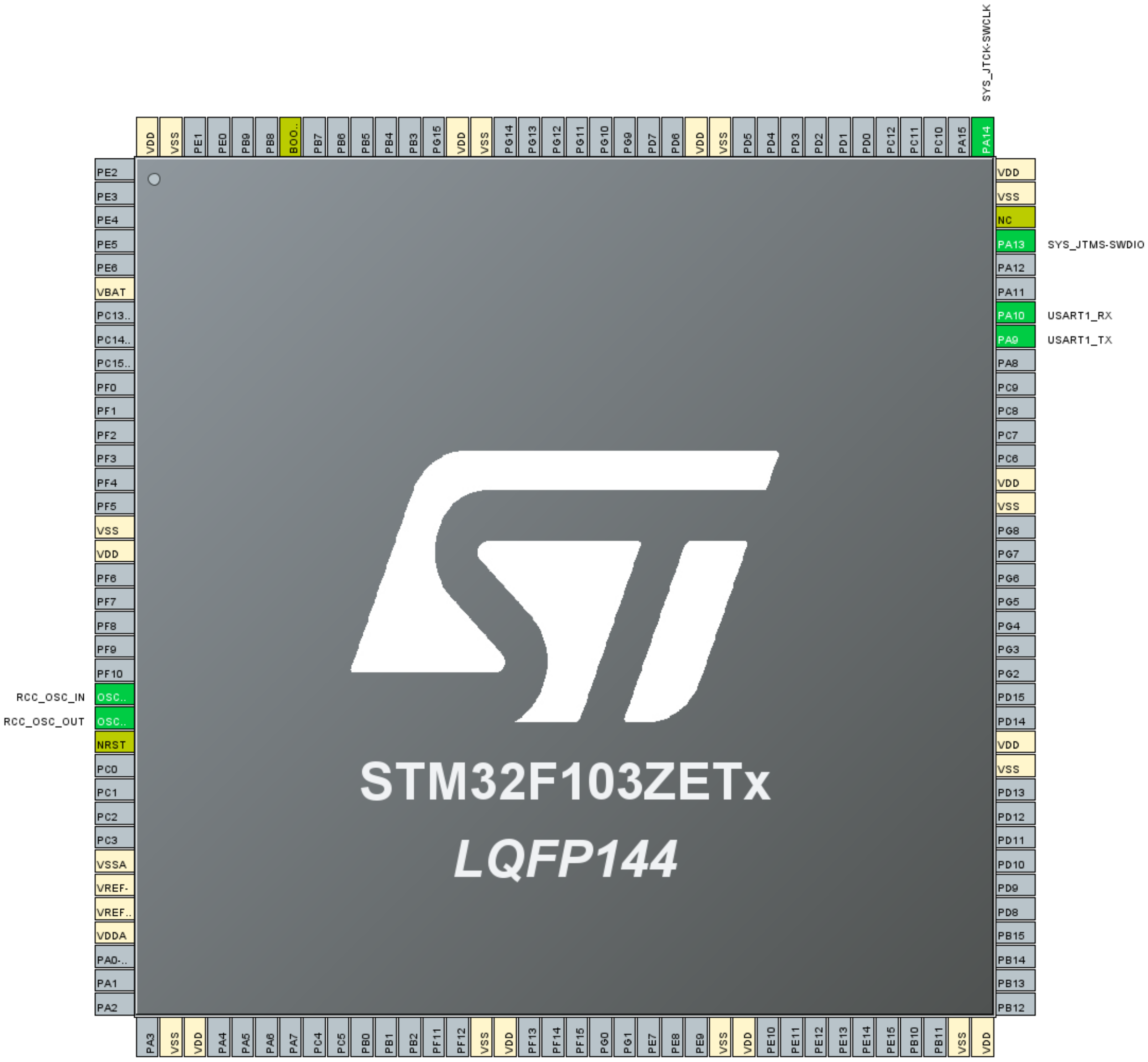
The project configuration part that is not omitted is the key point that needs to be configured in this tutorial.

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

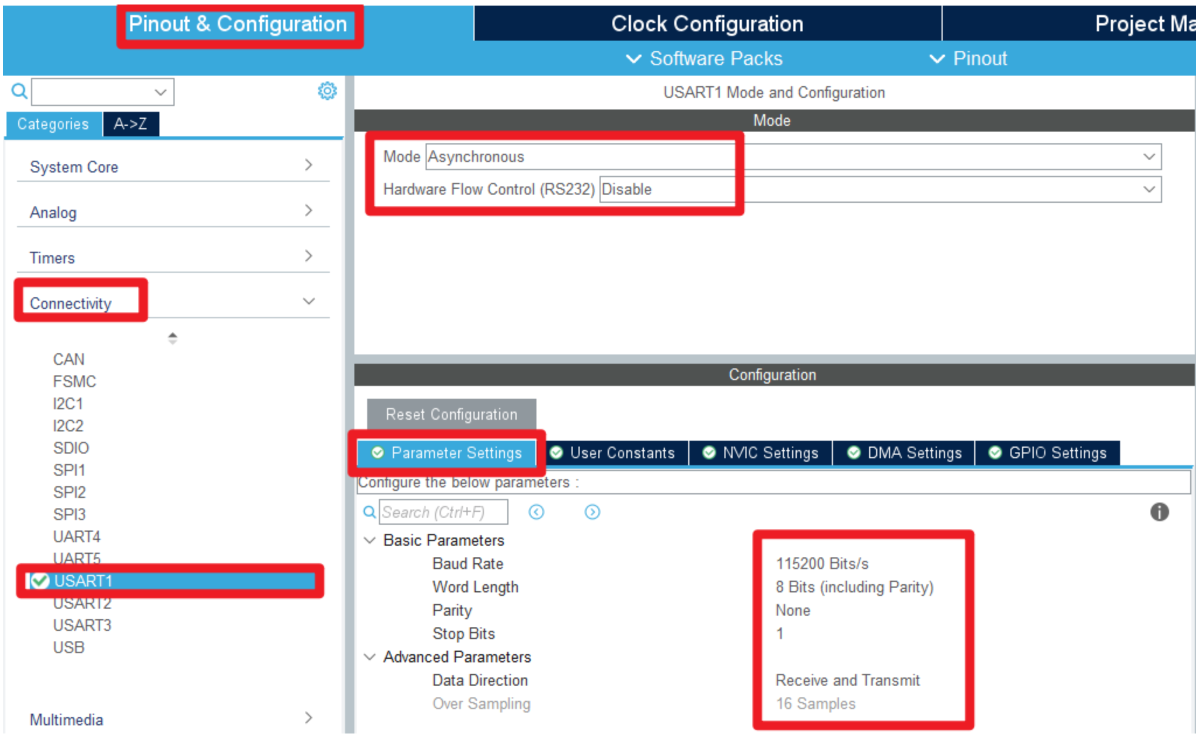
2. Pin configuration

- **Configure specified pin function**

You can directly select the corresponding pin number in the pin view, and the corresponding options will appear when you left-click the mouse.



- **USART**



USART1 Mode and Configuration

Mode Asynchronous

Hardware Flow Control (RS232) Disable

Configure the below parameters :

Basic Parameters

Baud Rate 115200 Bits/s

Word Length 8 Bits (including Parity)

Parity None

Stop Bits 1

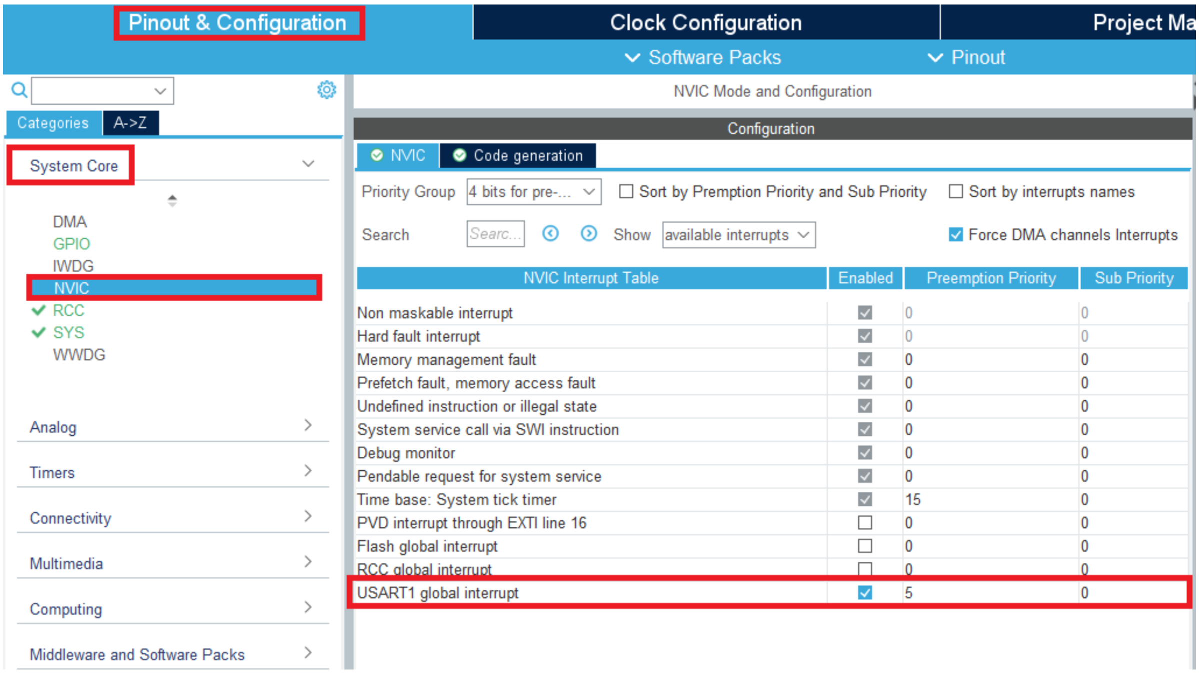
Advanced Parameters

Data Direction Receive and Transmit

Over Sampling 16 Samples

- **NVIC**

Later, when it comes to modifying the interrupt priority, you need to enter the NVIC interface settings. The default priority is used here.

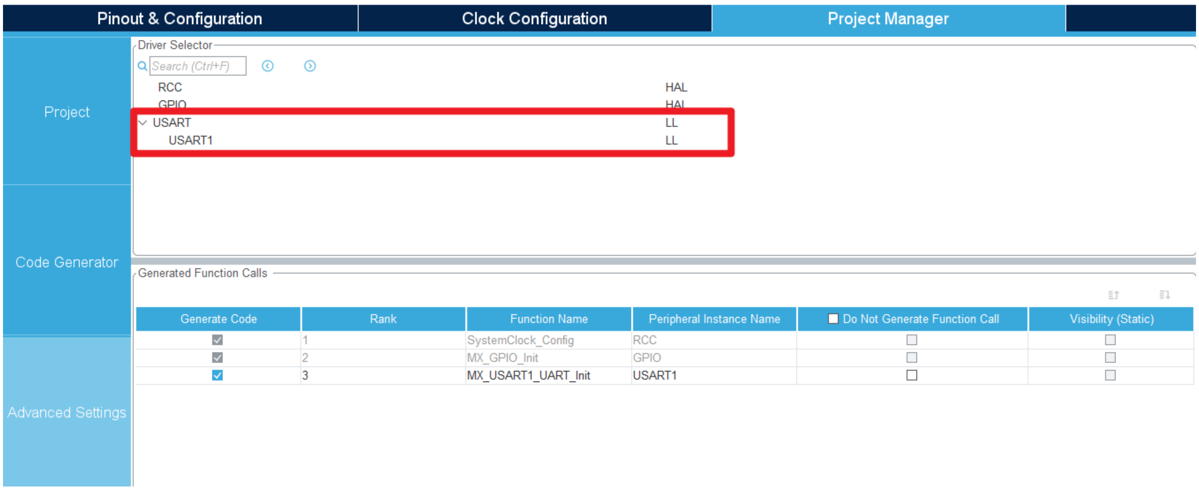


Priority Group: 4 bits for pre-... ☐ Sort by Premption Priority and Sub Priority ☐ Sort by interrupts names

Search: Searc... Show: available interrupts ☑ Force DMA channels Interrupts

NVIC Interrupt Table	Enabled	Preemption Priority	Sub Priority
Non maskable interrupt	☑	0	0
Hard fault interrupt	☑	0	0
Memory management fault	☑	0	0
Prefetch fault, memory access fault	☑	0	0
Undefined instruction or illegal state	☑	0	0
System service call via SWI instruction	☑	0	0
Debug monitor	☑	0	0
Pendable request for system service	☑	0	0
Time base: System tick timer	☑	15	0
PVD interrupt through EXTI line 16	☐	0	0
Flash global interrupt	☐	0	0
RCC global interrupt	☐	0	0
USART1 global interrupt	☑	5	0

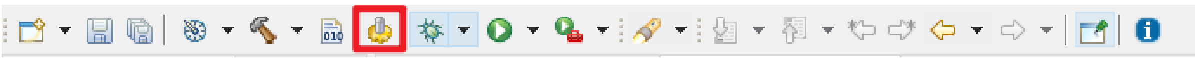
- **Advanced Settings**



Driver Selector	
RCC	HAL
GPIO	HAL
USART	LL
USART1	LL

Generate Code	Rank	Function Name	Peripheral Instance Name	☐ Do Not Generate Function Call	Visibility (Static)
☑	1	SystemClock_Config	RCC	☐	☐
☑	2	MX_GPIO_Init	GPIO	☐	☐
☑	3	MX_USART1_UART_Init	USART1	☐	☐

- **Generate code**



4. Main functions

It mainly introduces the functional code written by the user. For detailed code, you can open the project file provided by us yourself and enter the Bsp folder to view the source code.

1. User function

> **function：USART1_DataByte**

function prototype	**void USART1_DataByte(uint8_t data_byte)**
function description	Send a byte
Input parameters	**data_byte**：data

function prototype	**void USART1_DataByte(uint8_t data_byte)**
return value	none

> **function：USART1_DataString**

function prototype	**void USART1_DataString(uint8_t *data_str, uint16_t datasize)**
function description	send a string
Input parameters 1	**data_str**: data first address
Input parameters 2	**datasize**: Number of bytes of data
return value	none

2. LL library function analysis

The HAL library functions that have been introduced in the previous tutorial will not be introduced again in the tutorial!

```
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: LL_USART_Init**

function prototype	**ErrorStatus LL_USART_Init(USART_TypeDef *USARTx, const LL_USART_InitTypeDef *USART_InitStruct)**
function description	**Initialize USART peripheral**
Input parameters 1	**USARTx**：USARTx handle address
Input parameters 2	**USART_InitStruct**: USART initialization structure, containing configuration information of the specified USART peripherals
return value	**ErrorStatus**：SUCCESS、ERROR

> **function：LL_USART_ConfigAsyncMode**

function prototype	**void LL_USART_ConfigAsyncMode(USART_TypeDef *USARTx)**
function description	**Configure the specified USART peripheral to asynchronous mode**
Input parameters	**USARTx**：USARTx handle address
return value	none

> **function：LL_USART_Enable**

function prototype	void LL_USART_Enable(USART_TypeDef *USARTx)
function description	**Enable specified USART peripheral**
Input parameters	**USARTx**：USARTx handle address
return value	none

> **function：LL_USART_ReceiveData8**

function prototype	uint8_t LL_USART_ReceiveData8(const USART_TypeDef *USARTx)
function description	**Specify the USART peripheral to receive 8-bit data**
Input parameters	**USARTx**：USARTx handle address
return value	received data

> **function：LL_USART_TransmitData8**

function prototype	void LL_USART_TransmitData8(USART_TypeDef *USARTx, uint8_t Value)
function description	**Specify the USART peripheral to send 8-bit data**
Input parameters 1	**USARTx**：USARTx handle address
Input parameters 2	**Value**: sent data
return value	none

> **function：LL_USART_IsActiveFlag_TXE**

function prototype	uint32_t LL_USART_IsActiveFlag_TXE(const USART_TypeDef *USARTx)
function description	**Check the TXE flag status of the specified USART peripheral**
Input parameters	**USARTx**：USARTx handle address
return value	0 (inactive), 1 (active)

> **function：LL_USART_EnableIT_RXNE**

function prototype	void LL_USART_EnableIT_RXNE(USART_TypeDef *USARTx)
function description	**Enable specified USART peripheral receive interrupt**
Input parameters	**USARTx**：USARTx handle address
return value	none

> **function：LL_USART_IsEnabledIT_RXNE**

function prototype	**uint32_t LL_USART_IsEnabledIT_RXNE(const USART_TypeDef *USARTx)**
function description	**Check whether the receive data register non-empty interrupt of the specified USART peripheral is enabled**
Input parameters	**USARTx**：USARTx handle address
return value	0 (disabled), 1 (enabled)

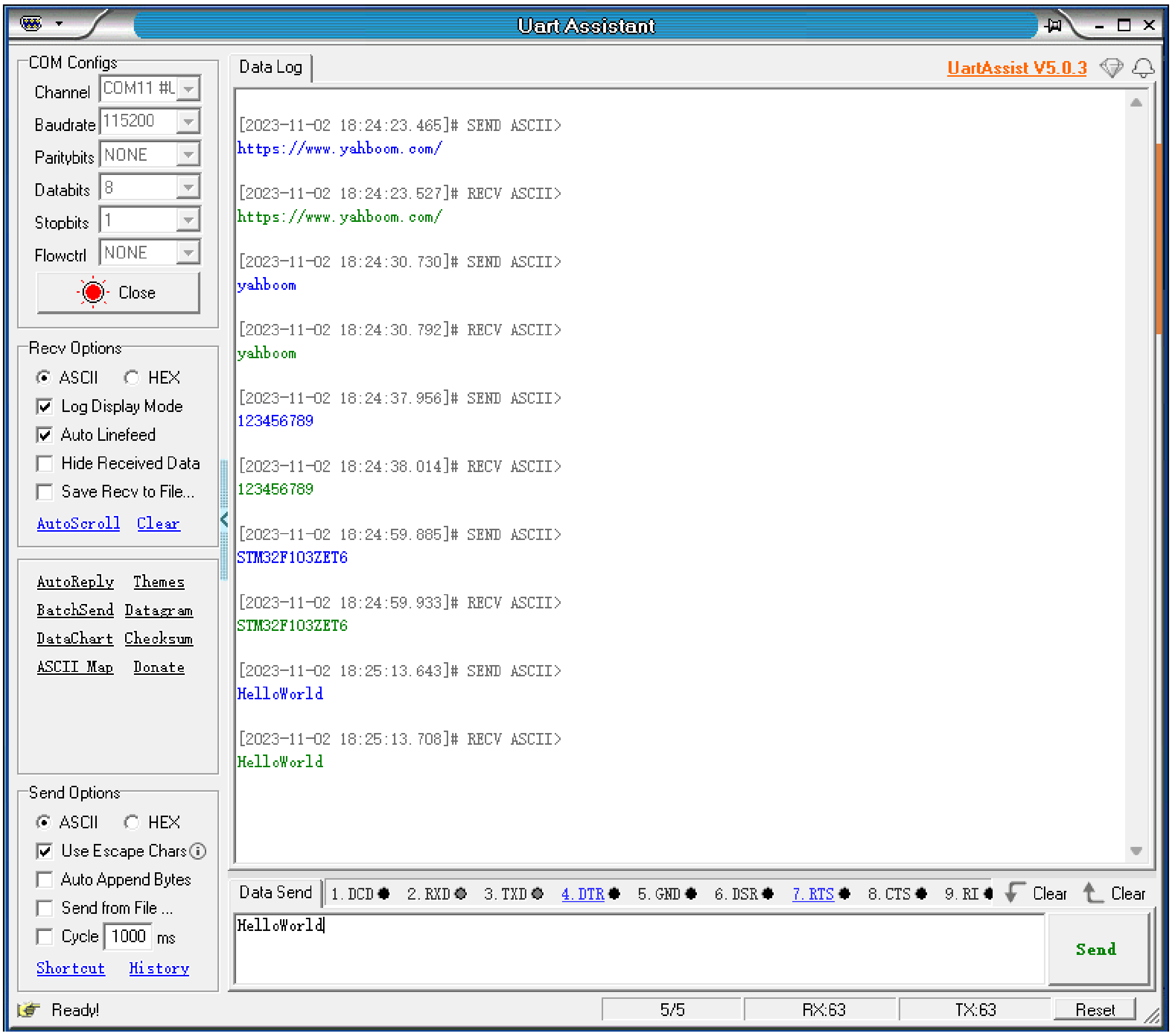
5. Experimental phenomena

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

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

Phenomenon:

Use the serial port debugging assistant to send information, and you can see that the serial port returns the same information.



COM Configs
Channel COM11 #L
Baudrate 115200
Paritybits NONE
Databits 8
Stopbits 1
Flowctrl NONE

Recv Options
● ASCII ○ HEX
☑ Log Display Mode
☑ Auto Linefeed
☐ Hide Received Data
☐ Save Recv to File...

Send Options
● ASCII ○ HEX
☑ Use Escape Chars
☐ Auto Append Bytes
☐ Send from File ...
☐ Cycle 1000 ms

```
[2023-11-02 18:24:23.465]# SEND ASCII>
https://www.yahboom.com/

[2023-11-02 18:24:23.527]# RECV ASCII>
https://www.yahboom.com/

[2023-11-02 18:24:30.730]# SEND ASCII>
yahboom

[2023-11-02 18:24:30.792]# RECV ASCII>
yahboom

[2023-11-02 18:24:37.956]# SEND ASCII>
123456789

[2023-11-02 18:24:38.014]# RECV ASCII>
123456789

[2023-11-02 18:24:59.885]# SEND ASCII>
STM32F103ZET6

[2023-11-02 18:24:59.933]# RECV ASCII>
STM32F103ZET6

[2023-11-02 18:25:13.643]# SEND ASCII>
HelloWorld

[2023-11-02 18:25:13.708]# RECV ASCII>
HelloWorld
```

Data Send: HelloWorld

Ready! 5/5 RX:63 TX:63