USB communication

This tutorial demonstrates: using the **USB Full Speed Device interface** to simulate the serial port to send and receive data (OLED shows USB connection status, LED represents different connection status).

1、software-hardware

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

  USB: Chip internal peripheral

  LED: Onboard

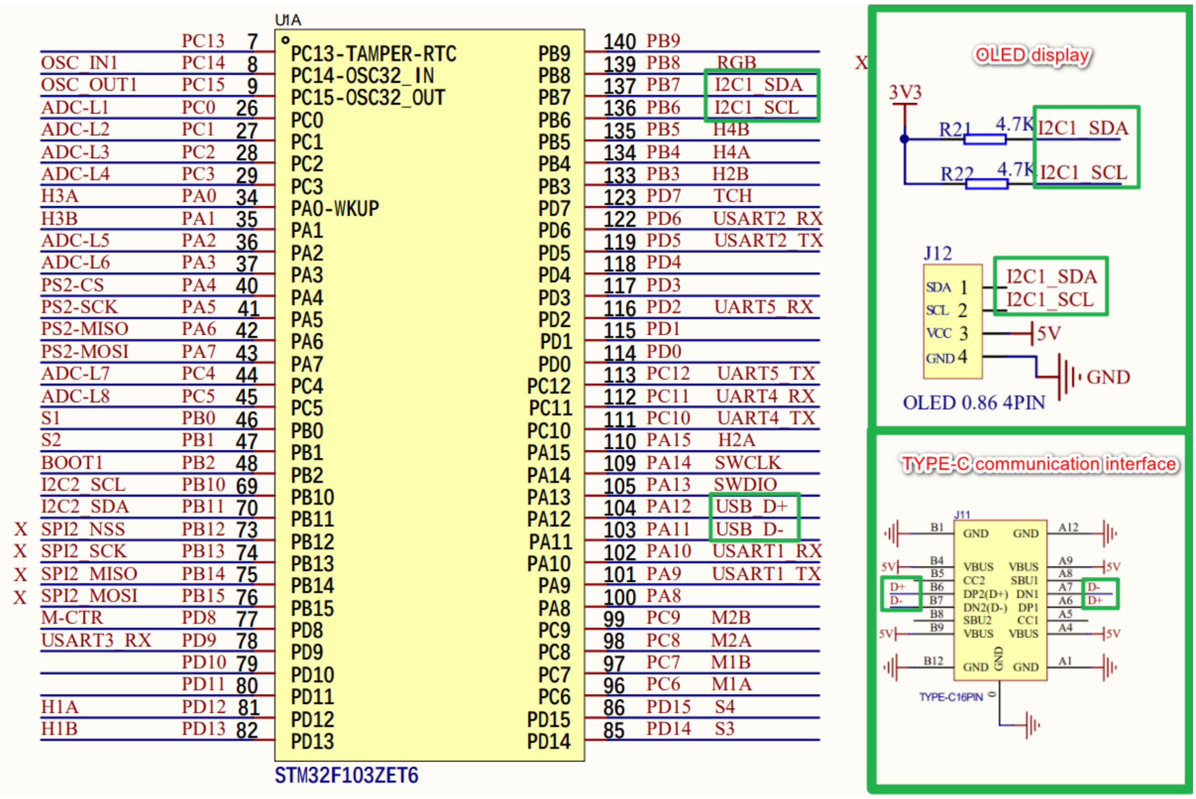
  OLED: External
- **Type-C cable or ST-Link**

  Download or simulate the program of the development board

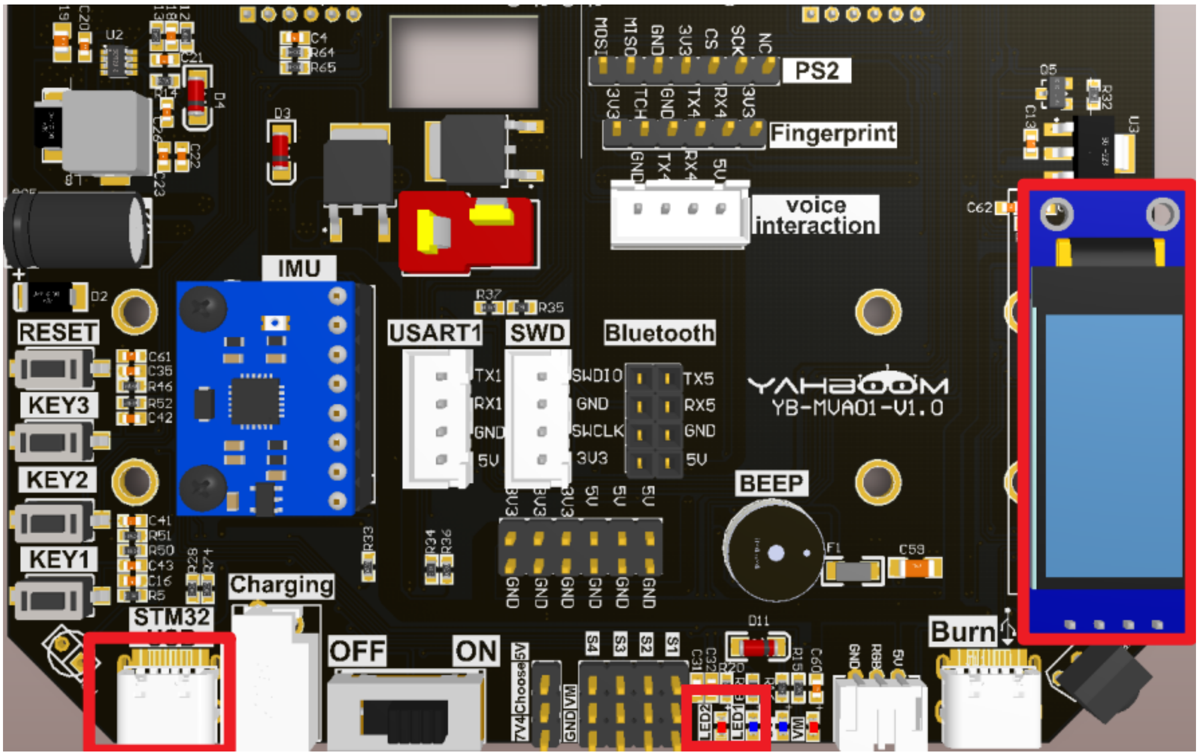
2、Brief principle

2.1、Hardware schematic diagram

The LED interface is not shown here, you can refer to the schematic diagram by yourself



2.2、Physical connection diagram



OLED	STM32 board
VCC	VCC
SCL	SCL
SDA	SDA
GND	GND

STM32 USB	PC
Type-C interface	USB interface

```
The Type-C data cable was used to connect the computer to the STM USB interface
```

2.3、Principle of control

- **USB**：(Universal Serial Bus)

USB	Characteristics
Class of speed	Low speed（USB1.0：1.5Mbps） Full speed（USB1.1：12Mbps） High speed（USB2.0：480Mbps） Super high speed（USB3.0：5Gbps、USB3.1：10Gbps） The development board is in full speed mode
Communication interface	Development board uses 16-pin Type-C interface (see schematic)
Data transfer	Control transmission Interrupt transmission Bulk transfer Isochronous transmission
Advantages	1. Support hotplugging and PNP (Plug-and-Play); 2. Provide USB peripherals with various speed levels to adapt to different requirements; 3. The port has very flexible expansibility; 4. Provides a single, easy-to-use standard connection type; 5. To meet the requirements of various types of peripheral devices, USB provides four different data transfer types.

- **USB Protocol**

The USB bus uses differential signal transmission, where D+ and D- represent the positive and negative data lines of the differential signal respectively;

USB1.1 and USB2.0 use a differential signal level of 3.3V and USB3.0 uses a differential signal level of 5V.

```
USB related knowledge can go to the "USB Encyclopedia of USB Chinese network" for
understanding and learning, the following only introduces the common concepts
```

- **Full speed/low speed USB bus device connection method**

High speed/full speed equipment：D+ connected with 1.5k pull-up resistor;

Low speed equipment：D-connected 1.5k pull-up resistor;

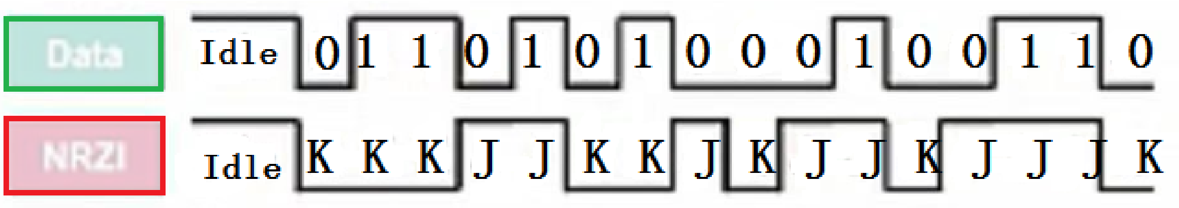
- **Bus status**

Level type	Scope
Difference 1	D+>2.8V，D-<0.3V
Difference 0	D+<0.3V，D-<0.3V
Single ended 0 （SE0）	D+<0.3V，D-<0.3V
Single ended 1 （SE1）	D+>2.8V，D->2.8V
J state： Low speed Full speed High speed	 Difference 0 Difference 1 Difference 1
K state Low speed Full speed High speed	 Difference 1 Difference 0 Difference 0
Restore status	K state
Packet start（SOF）	The USB data bus switches from the idle state to the K state
Packet end（EOP）	The SE0 lasts for two basic time units, and the J state lasts for one time unit

- **Encoding of USB signals**

The data in USB is encoded by NRZI (none-return to zero inverted), and the different states of D+ and D- lines are defined as J and K.

Encoding/decoding of data (reverse non-return-to-zero code)

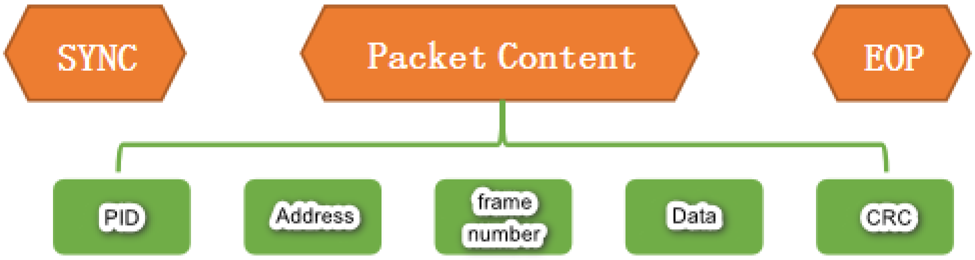

Data
Idle 0 1 1 0 1 0 1 0 0 0 1 0 0 1 1 0
NRZI
Idle K K K J J K K J K J J K J J J K

```
Note: The edge that meets 0 changes its state. If it is 1, the original state
remains
Before the data is NRZI encoded, a 0 signal is inserted after every six
consecutive 1 signals so as not to lose synchronization.
```

- **USB Protocol layer**

Packet is the basic unit of information transmission in USB system. All data is transmitted on the bus after being packaged.

Each packet consists of six parts: synchronization field (SYNC), packet identifier (PID), address field (ADDR), DATA field (including frame number), validation field (CRC), End of Packet (EOP).



- **Practical Applications**

The STM32F103ZET6 has an embedded device controller compatible with full speed USB, which complies with the standard for full speed USB devices (12 MBPS).

The USB-specific 48MHz clock is generated directly from the internal main PLL (the clock source must be a HSE crystal oscillator).

source file	function
usb_device.c	USB device initialization function
usbd_conf.c	USB protocol parameters, GPIO initialization, and other functions
usbd_cdc_if.c	Virtual serial port receiving and sending functions
usbd_desc.c	USB descriptor and USB enumeration processing

```
USB related knowledge does not do too much introduction, you can consult the
information on the Internet, mainly introduces the functions used in engineering
```

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

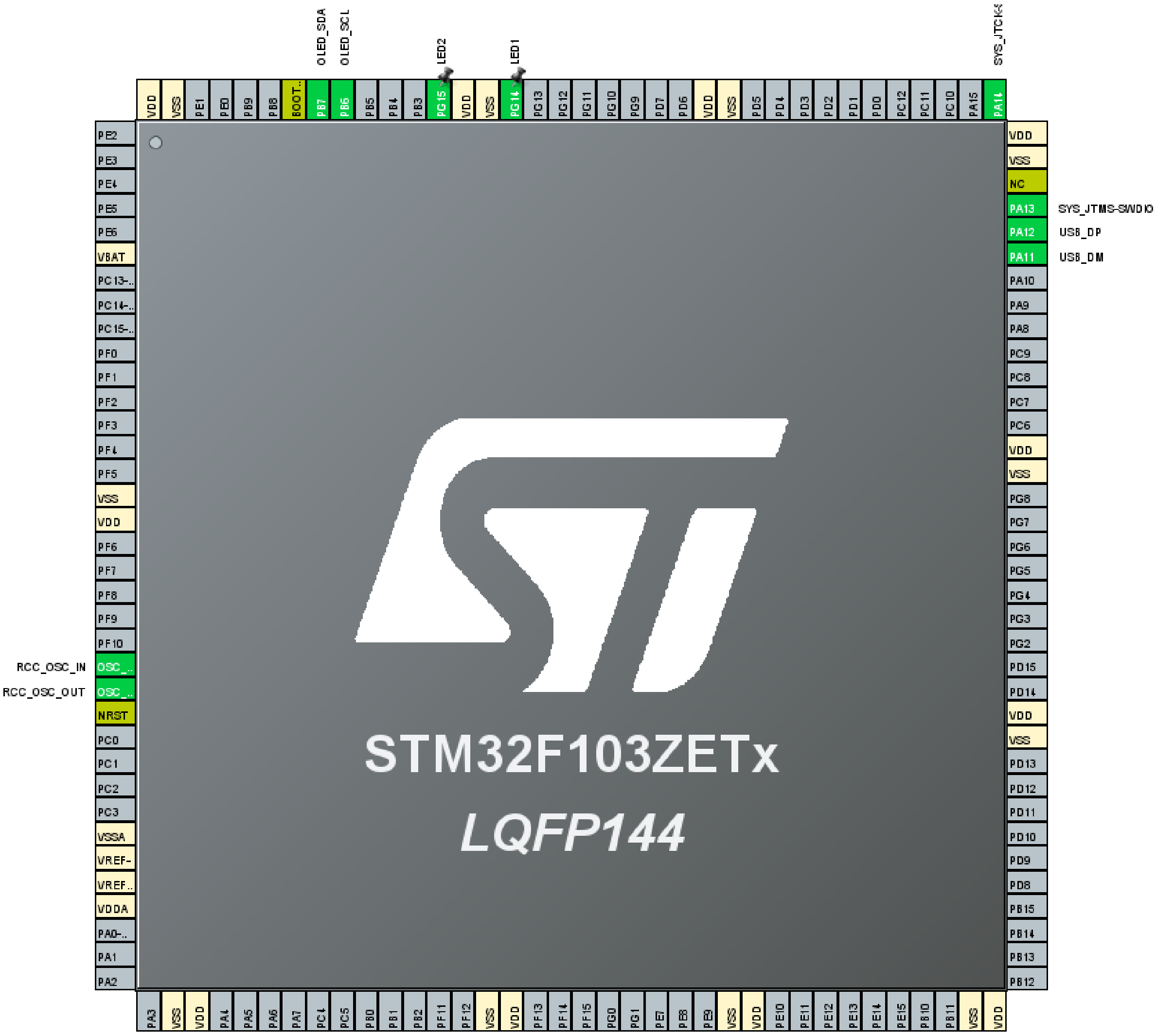
The project configuration part, which is not omitted, is the key point to configure in this tutorial.

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

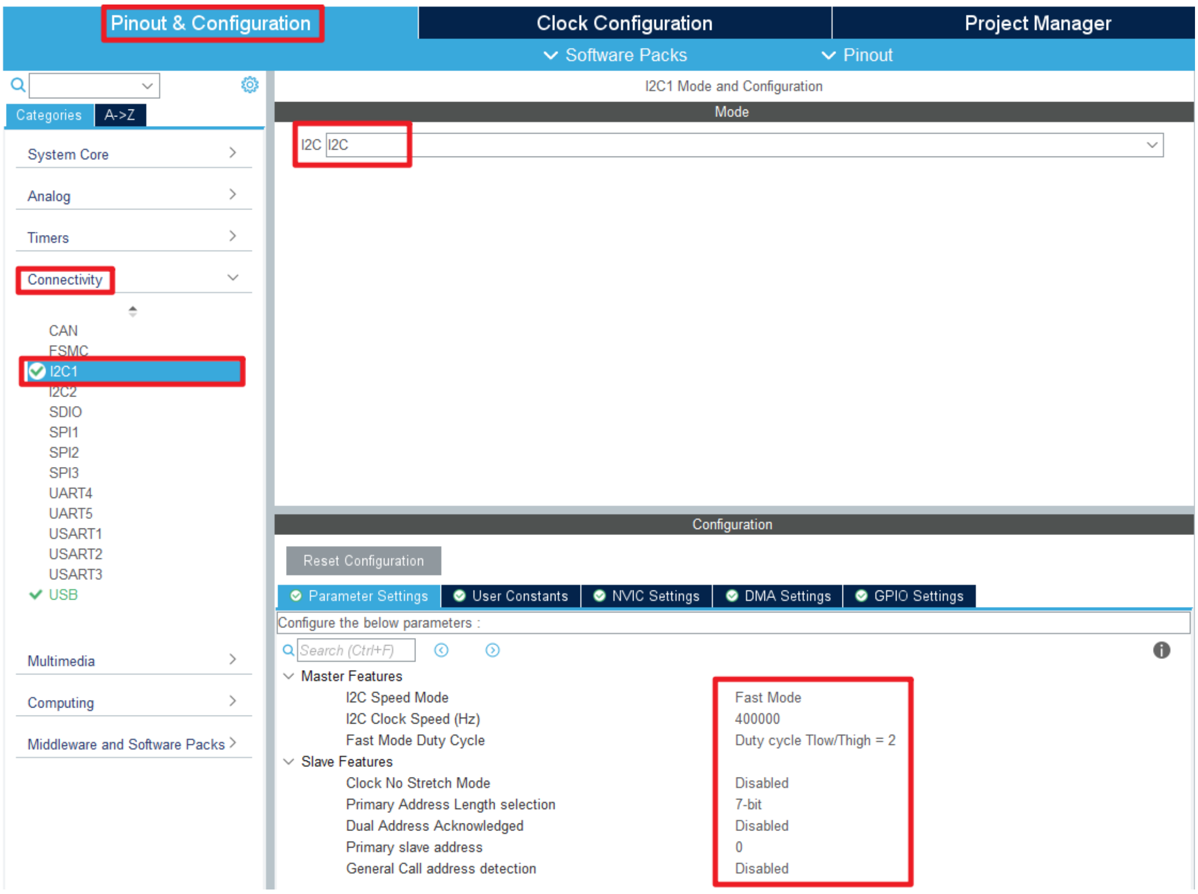
3.2、Pin configuration

- **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



- **I2C**



I2C1 Mode and Configuration

Mode

I2C I2C

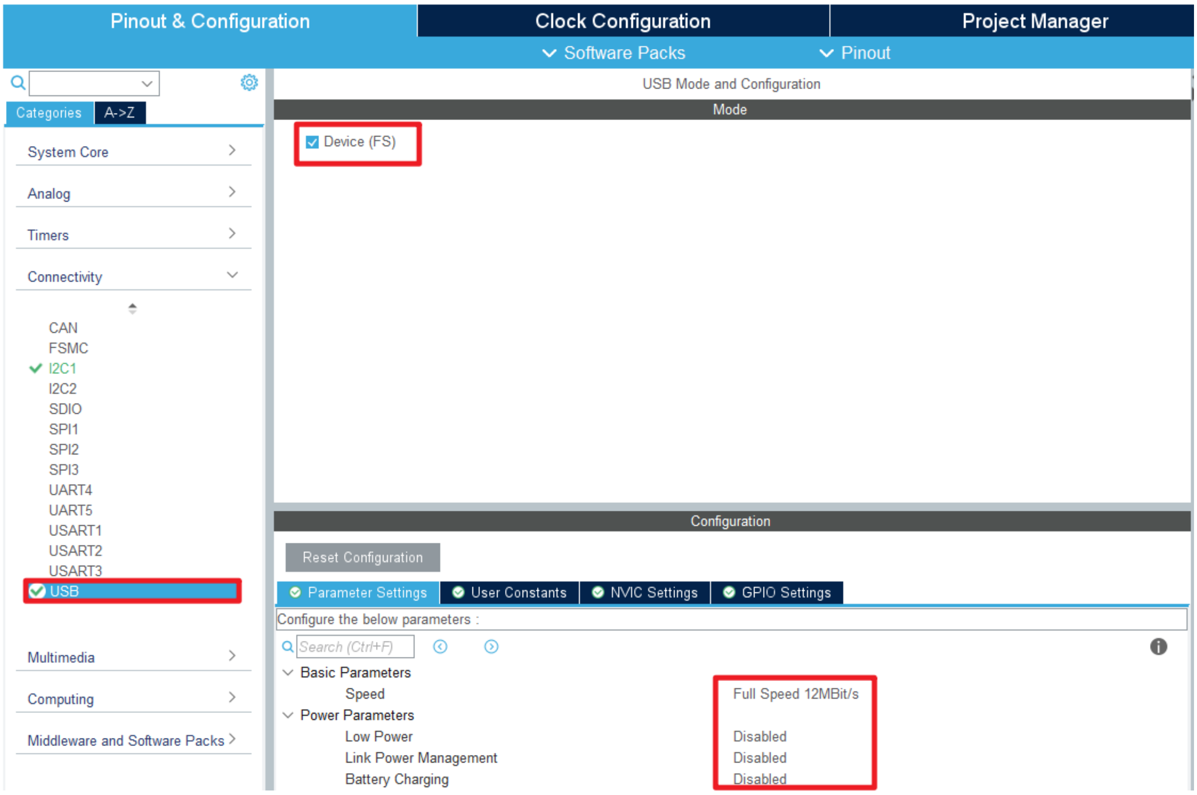
Configuration

Configure the below parameters :

Master Features
- I2C Speed Mode: Fast Mode
- I2C Clock Speed (Hz): 400000
- Fast Mode Duty Cycle: Duty cycle Tlow/Thigh = 2

Slave Features
- Clock No Stretch Mode: Disabled
- Primary Address Length selection: 7-bit
- Dual Address Acknowledged: Disabled
- Primary slave address: 0
- General Call address detection: Disabled

- **USB**



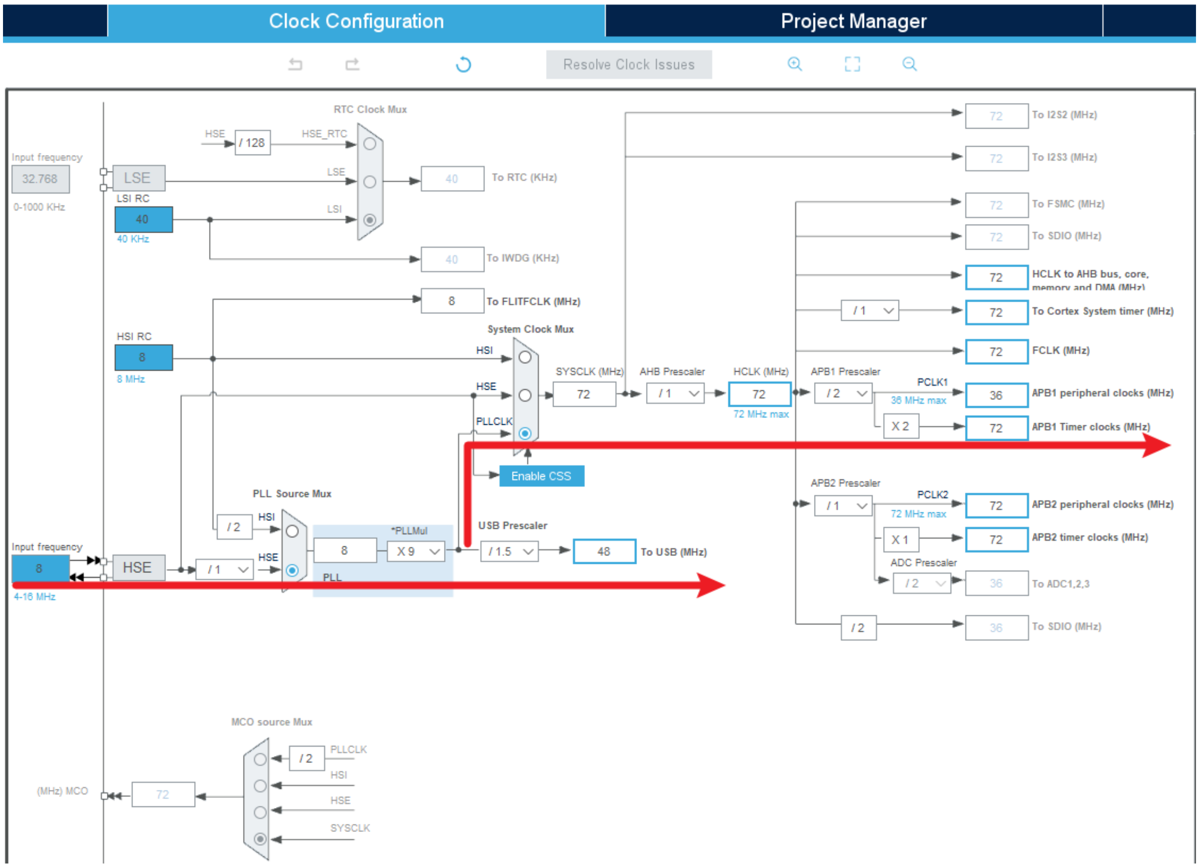
USB Mode and Configuration

☑ Device (FS)

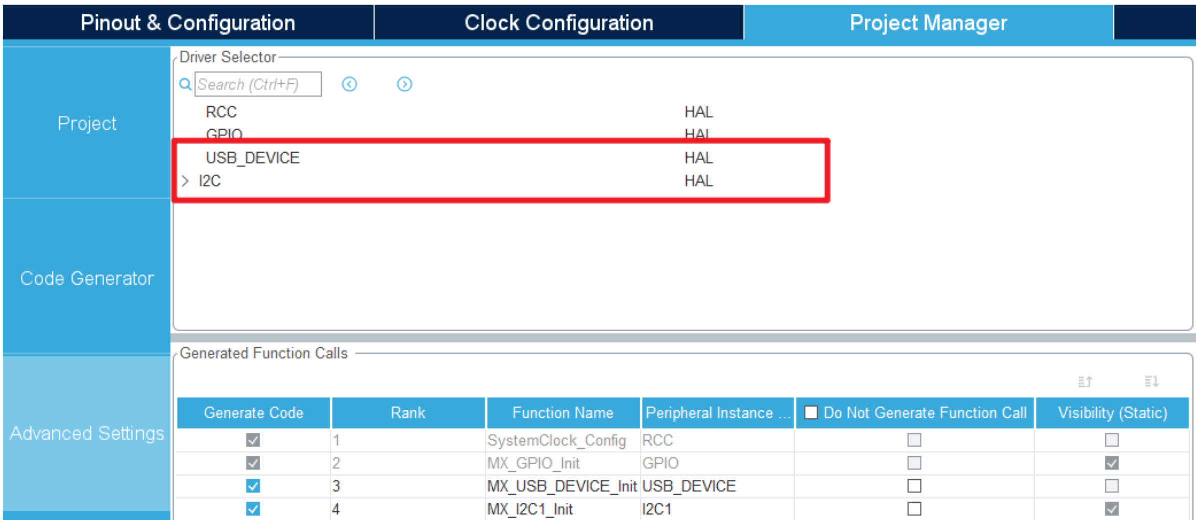
Configure the below parameters :

Basic Parameters
Speed Full Speed 12MBit/s
Power Parameters
Low Power Disabled
Link Power Management Disabled
Battery Charging Disabled

- **Clock Configuration**



- **Advanced Settings**



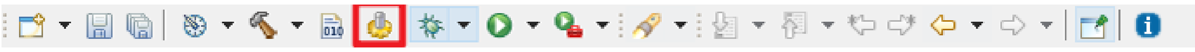
Driver Selector

RCC	HAL
GPIO	HAL
USB_DEVICE	HAL
> I2C	HAL

Generated Function Calls

Generate Code	Rank	Function Name	Peripheral Instance ...	□ Do Not Generate Function Call	Visibility (Static)
☑	1	SystemClock_Config	RCC	□	□
☑	2	MX_GPIO_Init	GPIO	□	☑
☑	3	MX_USB_DEVICE_Init	USB_DEVICE	□	□
☑	4	MX_I2C1_Init	I2C1	□	☑

- **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：USB_Init**

Function prototypes	**void USB_Init(void)**
Functional Description	USB pin initialization
Input parameters	None
Return value	None
Tips	Prevent the need to plug and unplug the USB port again after downloading the program

> **function：VCP_Status**

Function prototypes	**void VCP_Status(void)**
Functional Description	Check the USB connection status
Input parameters	None
Return value	None

> **function：usb_printf**

Function prototypes	void usb_printf(const char *format, ...)
Functional Description	printf redefined to usb serial port (send function)
Input parameters	Specifies the content and format to output
Return value	None

> **function：Deal_Recv**

Function prototypes	void Deal_Recv(void)
Functional Description	Process the received data
Input parameters	None
Return value	None
Tips	This routine prints the received data and displays the length of the data

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：MX_USB_DEVICE_Init**

Function prototypes	void MX_USB_DEVICE_Init(void)
Functional Description	Initialize the USB device
Input parameters	None
Return value	None

> **function：MX_USB_DEVICE_Init**

Function prototypes	USBD_StatusTypeDef USBD_Init (USBD_HandleTypeDef *pdev, USBD_DescriptorsTypeDef *pdesc, uint8_t id)
Functional Description	Initialize the configuration and parameters of the USB device
Input parameters1	**pdev**：Handle to a USB device
Input parameters2	**pdesc**：USB device descriptor information

Function prototypes	**USBD_StatusTypeDef USBD_Init (USBD_HandleTypeDef *pdev, USBD_DescriptorsTypeDef *pdesc, uint8_t id)**
Input parameters3	**id**：The identifier of the USB device
Return value	USB status：USBD_OK、USBD_BUSY、USBD_FAIL

> **function：USB_LP_CAN1_RX0_IRQHandler**

Function prototypes	**void USB_LP_CAN1_RX0_IRQHandler(void)**
Functional Description	Handle USB_LP_CAN1_RX0_IRQn interrupt (USB low priority and CAN1 receive 0 interrupt)
Input parameters	None
Return value	None

5、Experimental phenomenon

After downloading the program successfully, press the RESET button of the development board to observe the OLED display and LED display

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

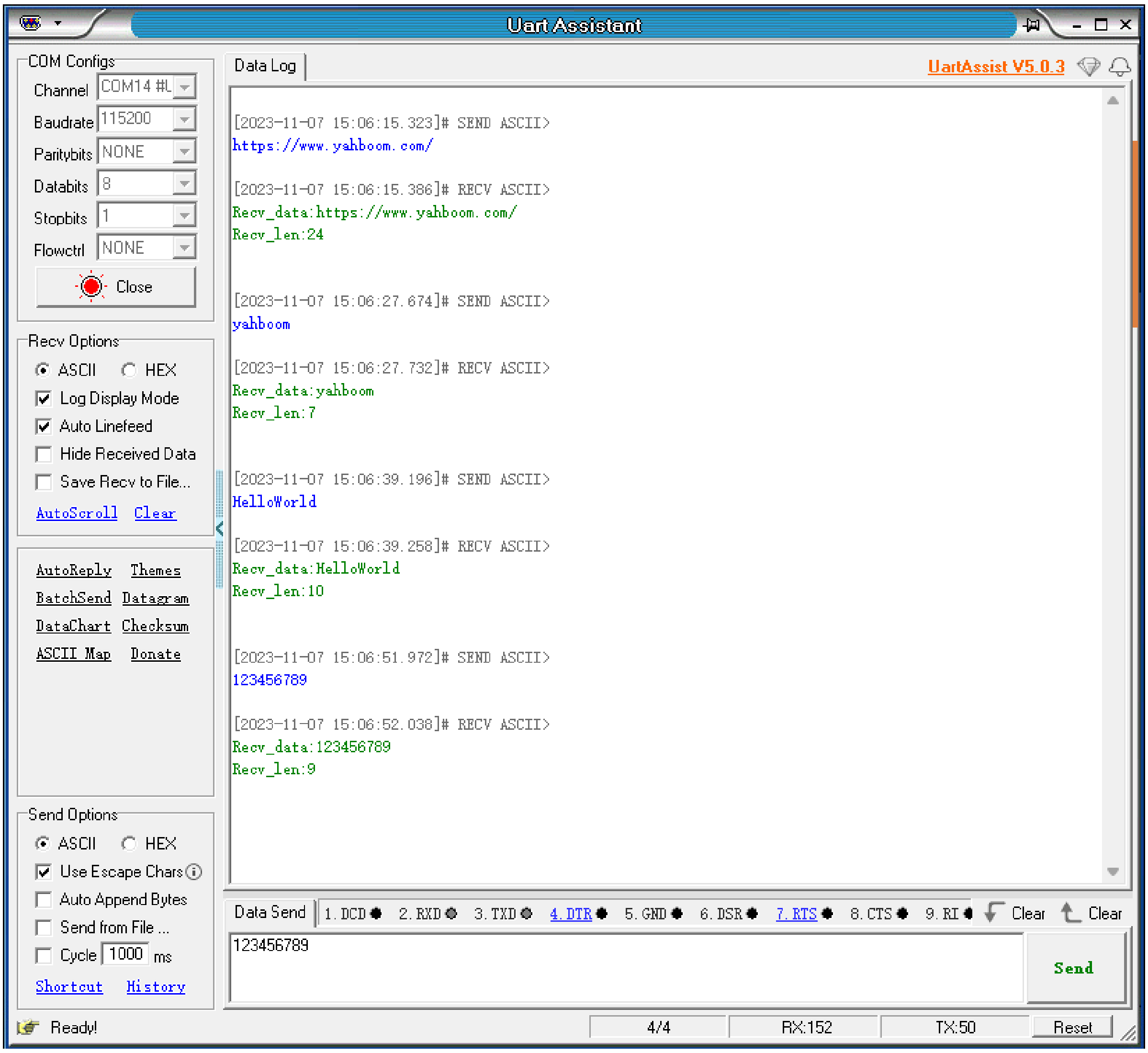
phenomenon：

The STM32 USB interface is not connected：LED1 off, OLED display "USB disconnect";

Connect the STM32 USB port：LED1 bright, OLED display "USB connect";

Use the serial debugging assistant to connect the virtual serial port of the USB interface. Sending data will return the same data.

If the virtual serial port shows an exception in the Device Manager：Install "STM32 USB virtual serial port Driver" in the "Project Source" folder



COM Configs
Channel COM14 #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...

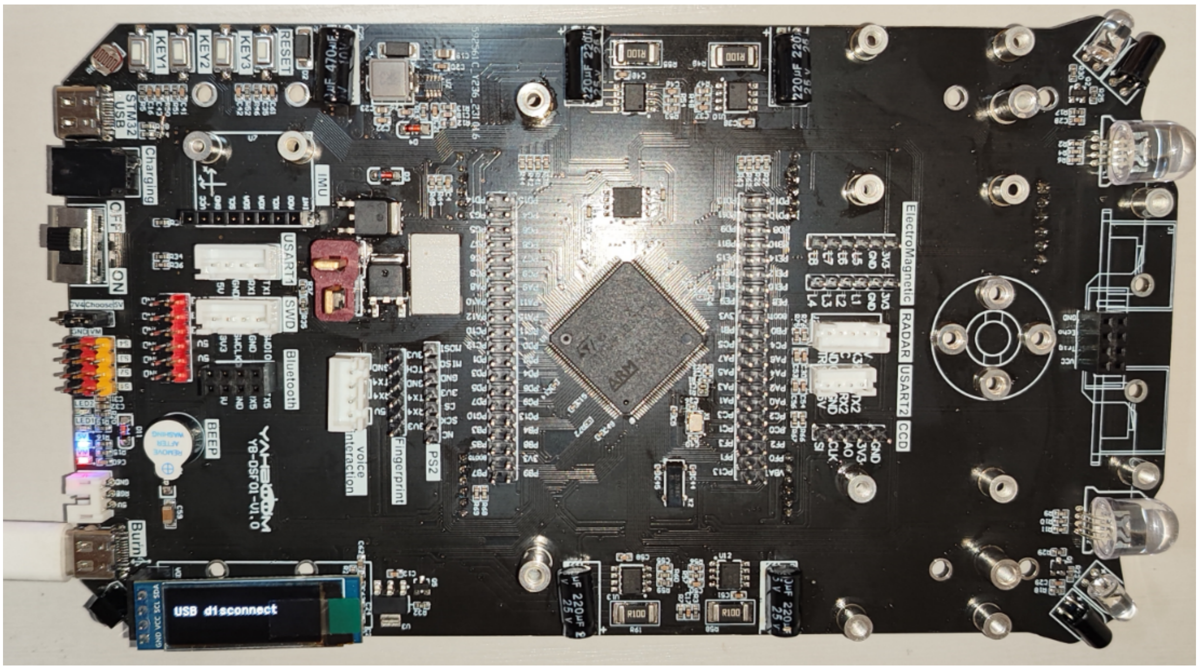
Send Options
● ASCII ○ HEX
☑ Use Escape Chars
□ Auto Append Bytes
□ Send from File ...
□ Cycle 1000 ms

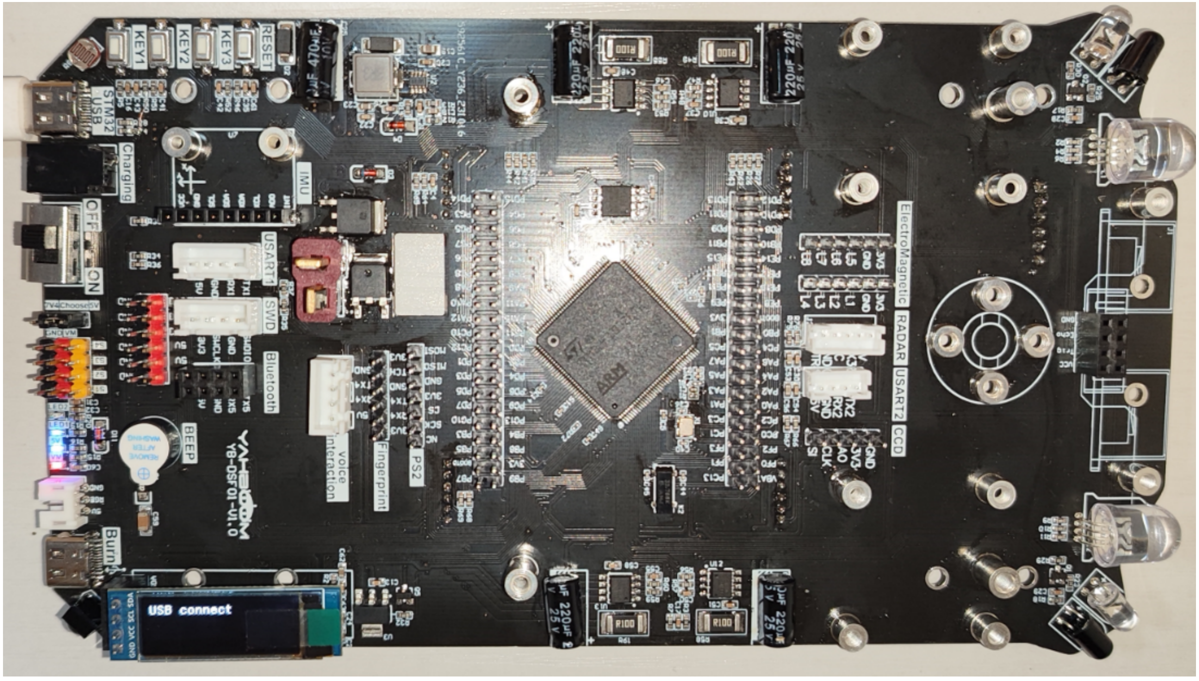
```
[2023-11-07 15:06:15.323]# SEND ASCII>
https://www.yahboom.com/

[2023-11-07 15:06:15.386]# RECV ASCII>
Recv_data:https://www.yahboom.com/
Recv_len:24


[2023-11-07 15:06:27.674]# SEND ASCII>
yahboom

[2023-11-07 15:06:27.732]# RECV ASCII>
Recv_data:yahboom
Recv_len:7


[2023-11-07 15:06:39.196]# SEND ASCII>
HelloWorld

[2023-11-07 15:06:39.258]# RECV ASCII>
Recv_data:HelloWorld
Recv_len:10


[2023-11-07 15:06:51.972]# SEND ASCII>
123456789

[2023-11-07 15:06:52.038]# RECV ASCII>
Recv_data:123456789
Recv_len:9
```

123456789

Ready! 4/4 RX:152 TX:50

Note the Type-C data line interface location for the following experimental phenomena




KEY1
KEY2
KEY3
RESET
STM32 USB
Charging
OFF
ON
IMU
USART1
SWD
Bluetooth
BEEP
YAHBOOM
YB-DSF01-V1.0
voice interaction
Fingerprint
PS2
ElectroMagnetic
RADAR
USART2
CCD
Burn
USB connect