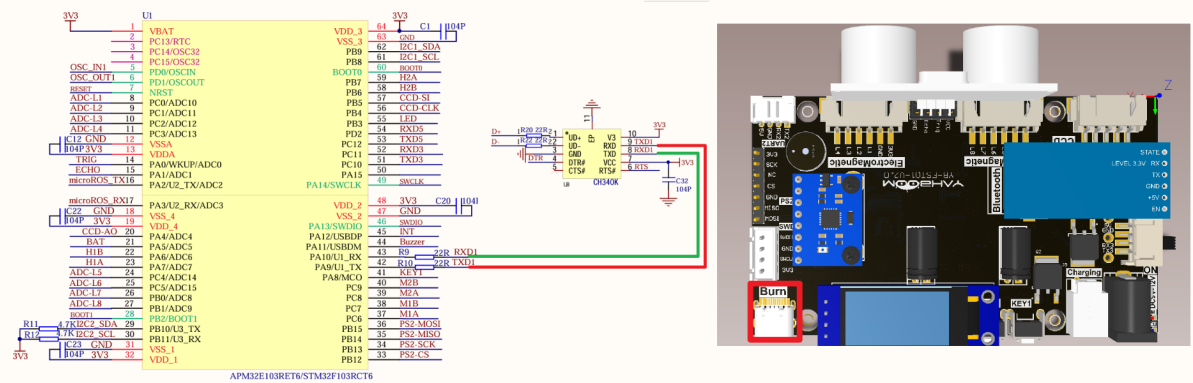
Serial communication (USART)

The tutorial demonstrates the serial port (USART1) interruption to send and receive data.

```
The tutorial only introduces the standard library project code
```

Hardware connection



Peripherals	Development board	Description
USART1_TX	PA9	USART1 sends data
USART1_RX	PA10	USART1 receives data

Control principle

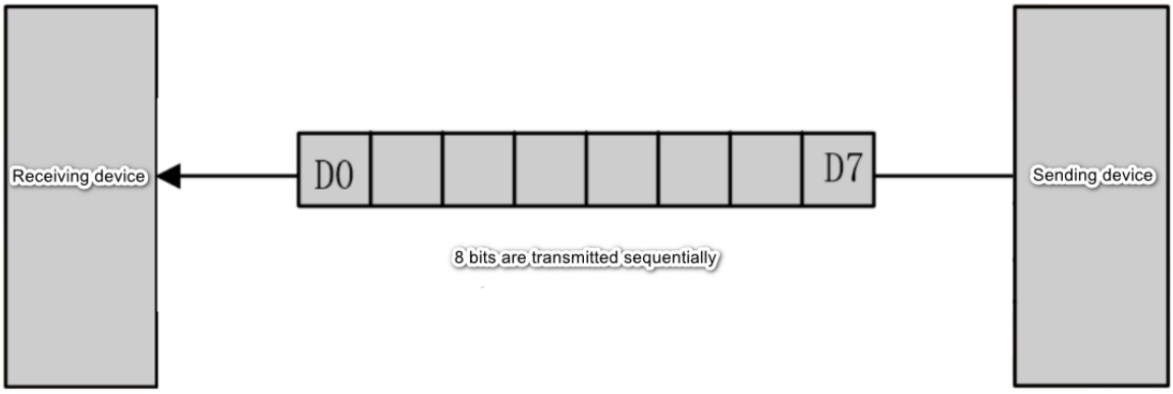
Sending data to the serial port (USART1) through the serial port debugging assistant triggers the serial port interrupt, and the data is printed out through the serial port interrupt callback function.

```
The onboard Type-C interface is connected to USART1, so we can use the Type-C
interface to complete program burning and serial communication
```

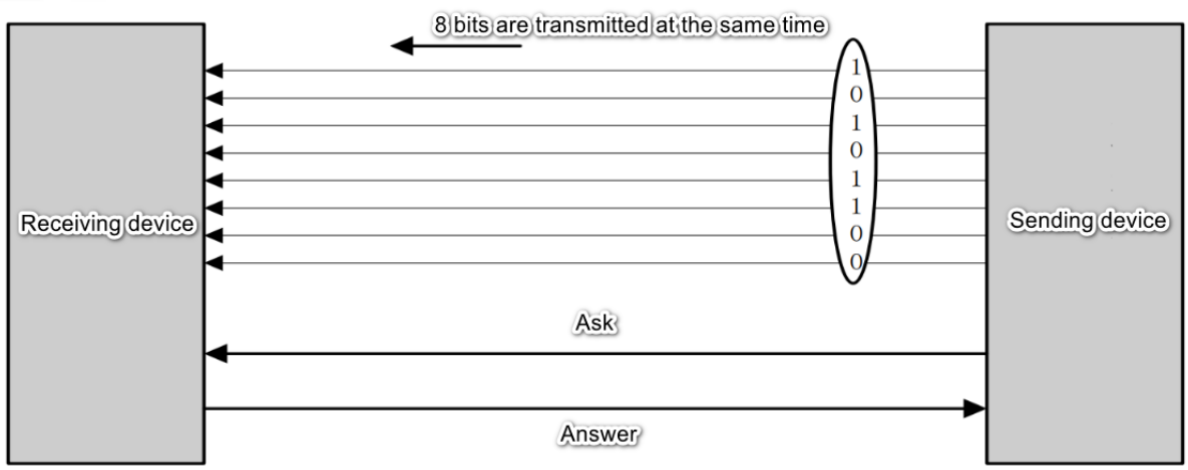
- **Computer communication**

Classified by data transmission method: serial communication and parallel communication

- **Serial communication**: data is transmitted bit by bit



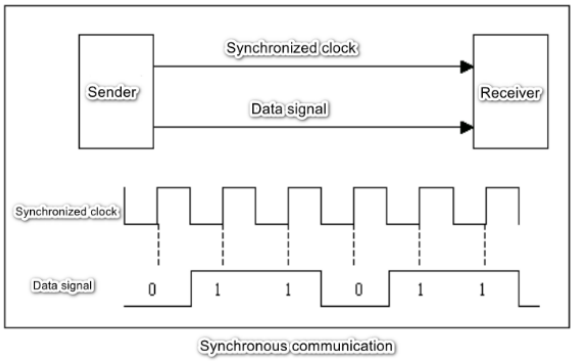
- **Parallel communication**: multi-bit data is transmitted simultaneously

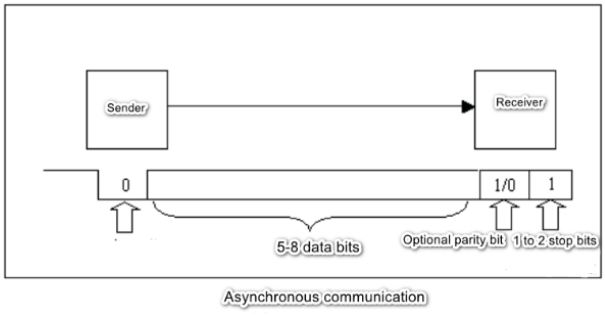


- **Serial communication**
- **Serial communication data synchronization method**: synchronous serial communication and asynchronous serial communication

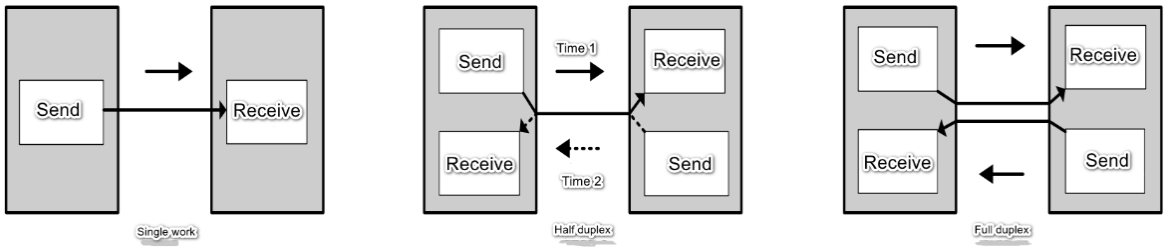
Synchronous serial communication: The sender and receiver rely on independent clock lines to synchronize signals

Asynchronous serial communication: The sender and receiver complete the communication according to the agreed character format and communication rate





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



Simple: 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)

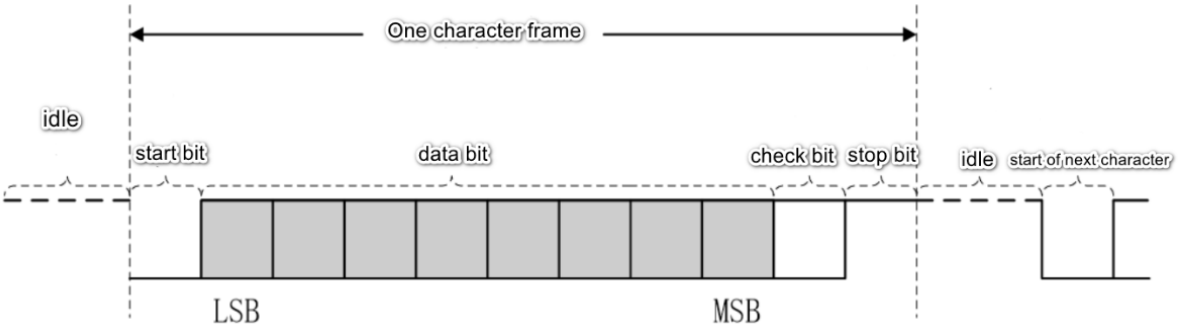
USART supports synchronous and asynchronous communication. Our tutorial uses UART (Universal Serial Asynchronous Receiver/Transmitter).

- **Asynchronous Serial Communication**
- **Communication Rate**

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

```
Common baud rates: 9600, 57600 and 115200
Baud rate 115200: 115200 bits are transmitted per second
```

- **Character Format**




```
When transmitting data, the low bit is in front and the high bit is in the back
```

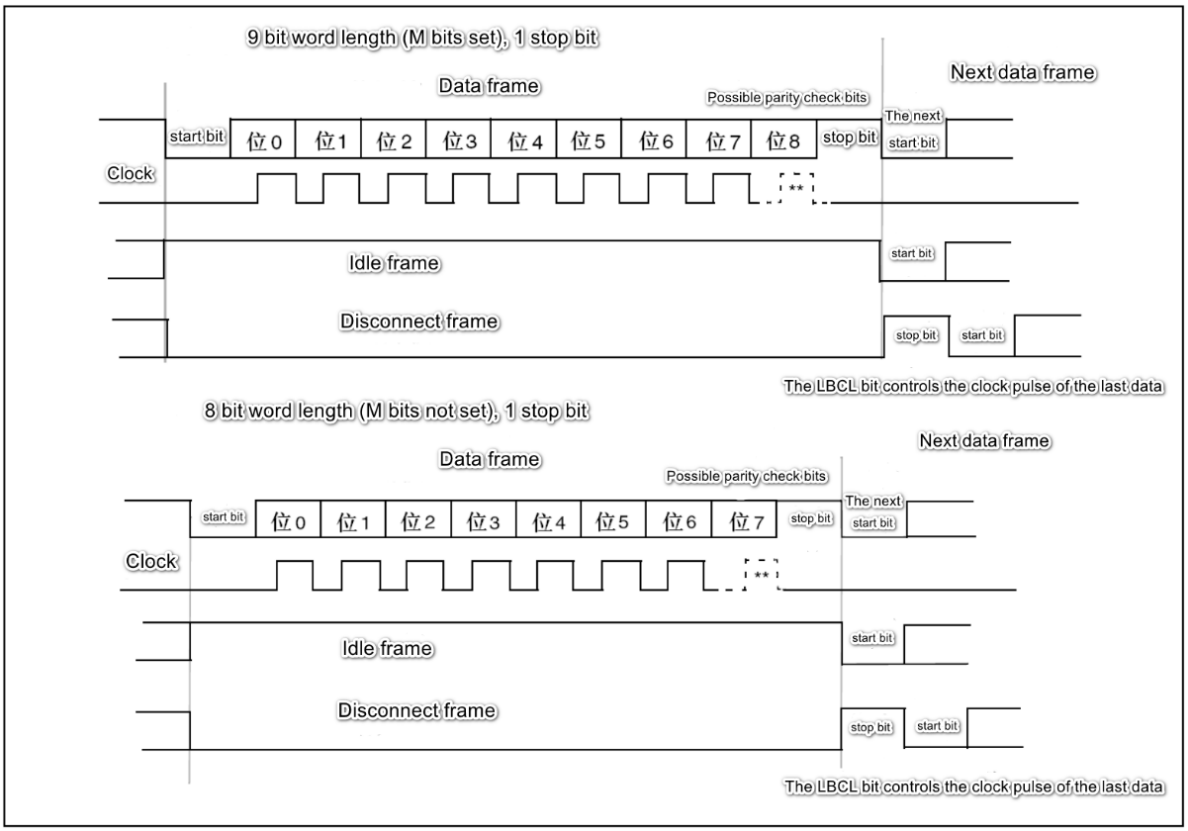
Data frame format: **Start bit + data bit + check bit + Stop bit**

Start bit (start of data frame): 1 bit

Data bit: 5~8 bits

Parity bit: odd or even (is the number of "1" in the data odd or even)

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



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

> **Onboard serial port interface description**

Serial port	Peripherals	Onboard peripheral connection
USART1_TX (pin function: default multiplexing function)	PA9	Type-C interface use
USART1_RX (pin function: default multiplexing function)	PA10	Type-C interface use
USART2_TX (pin function: default multiplexing function)	PA2	K210 vision module use
USART2_RX (pin function: default multiplexing function)	PA3	K210 vision module use
USART3_TX (pin function: redefined function)	PC10	Radar use
USART3_RX (pin function: redefined function)	PC11	Radar use
UART5_TX (pin function: default multiplexing function)	PC12	Bluetooth module use
UART5_RX (pin function: default multiplexing function)	PD2	Bluetooth module use

Software configuration

Pin definition

Main control chip	Pin	Main function (after reset)	Default multiplexing function	Redefined function
STM32F103RCT6	PA9	PA9	USART1_TX/TIM1_CH2	
STM32F103RCT6	PA10	PA10	USART1_RX/TIM1_CH3	

Software code

Since the default function of the PA9/10 pin is a normal IO pin function, we need to use the multiplexing function.

```
Product supporting materials source code path: Attachment → Source code summary → 1.Base_Course → 5.USART
```

Control function

The tutorial only briefly introduces the code, you can open the project source code to read the details.

> **uart_init**

```
void uart_init(u32 bound)
{
    GPIO_InitTypeDef GPIO_InitStructure;
    USART_InitTypeDef USART_InitStructure;
    NVIC_InitTypeDef NVIC_InitStructure;

    RCC_APB2PeriphClockCmd(RCC_APB2Periph_USART1|RCC_APB2Periph_GPIOA, ENABLE);

    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_9;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
    GPIO_Init(GPIOA, &GPIO_InitStructure);

    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_10;//PA10
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING;
    GPIO_Init(GPIOA, &GPIO_InitStructure);

    USART_InitStructure.USART_BaudRate = bound;
    USART_InitStructure.USART_WordLength = USART_WordLength_8b;
    USART_InitStructure.USART_StopBits = USART_StopBits_1;
    USART_InitStructure.USART_Parity = USART_Parity_No;
    USART_InitStructure.USART_HardwareFlowControl =
USART_HardwareFlowControl_None;
    USART_InitStructure.USART_Mode = USART_Mode_Rx | USART_Mode_Tx;

    USART_Init(USART1, &USART_InitStructure);
    USART_ITConfig(USART1, USART_IT_RXNE, ENABLE);
```

```
    USART_Cmd(USART1, ENABLE);

    NVIC_InitStructure.NVIC_IRQChannel = USART1_IRQn;
    NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
    NVIC_InitStructure.NVIC_IRQChannelSubPriority = 1;
    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
    NVIC_Init(&NVIC_InitStructure);
}
```

> **USART1_IRQHandler**

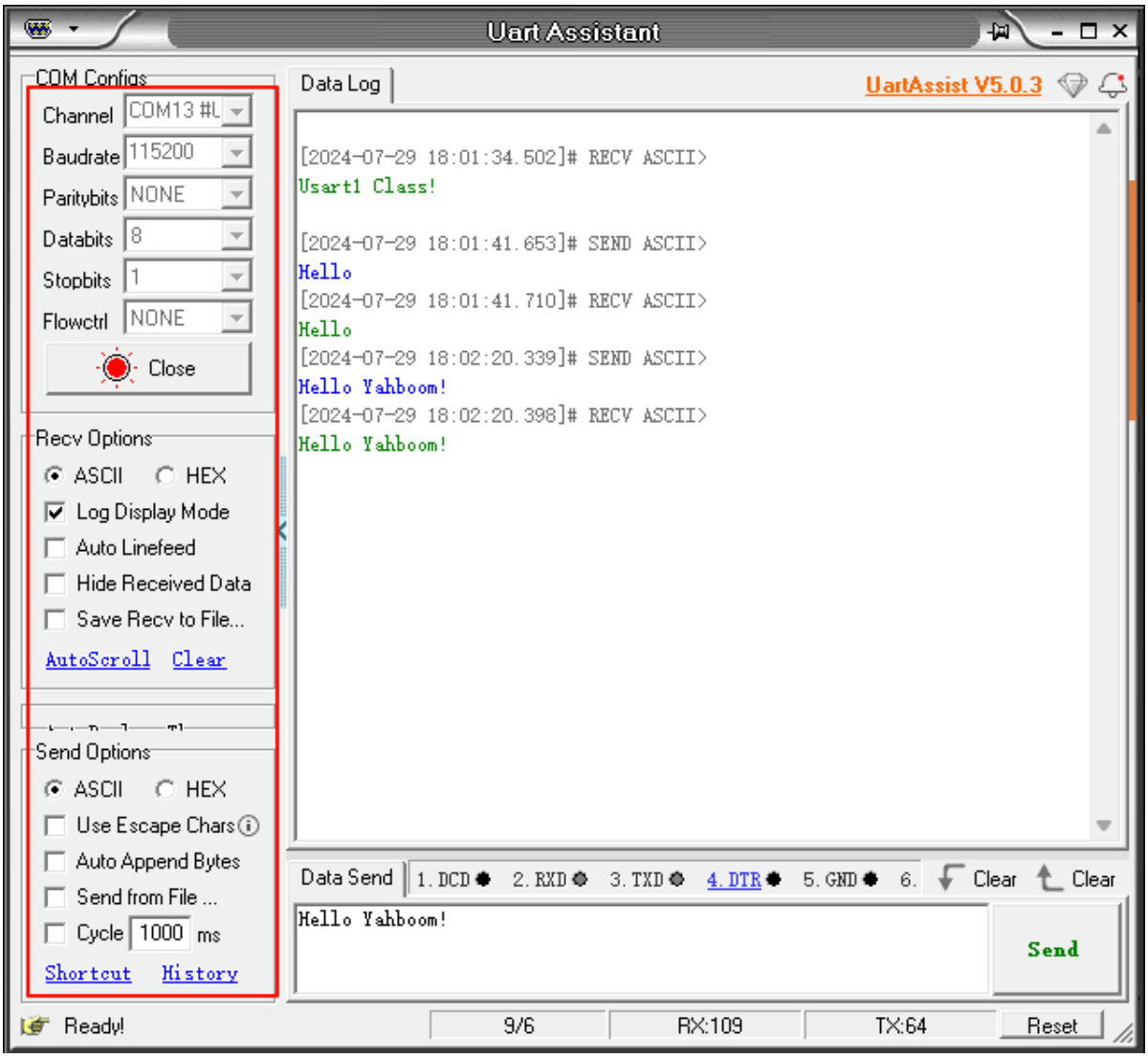
```
void USART1_IRQHandler(void)
{
    uint8_t Rx1_Temp = 0;
    if (USART_GetITStatus(USART1, USART_IT_RXNE) != RESET)
    {
        Rx1_Temp = USART_ReceiveData(USART1);
        USART1_Send_U8(Rx1_Temp);
    }
}
```

Experimental phenomenon

The USART.hex file generated by the project compilation is located in the OBJ folder of the USART project. Find the USART.hex file corresponding to the project and use the FlyMcu software to download the program into the development board.

After the program is successfully downloaded: the LED switches on and off every 500ms; use the serial port debugging assistant to send information, and you can see that the serial port returns the same information.

```
When using the serial port debugging assistant, you need to pay attention to the
serial port settings. Incorrect settings may cause inconsistent phenomena
```


Uart Assistant
COM Configs
Channel
COM13 #L
Baudrate
115200
Paritybits
NONE
Databits
8
Stopbits
1
Flowctrl
NONE
Close
Recv Options
ASCII
HEX
Log Display Mode
Auto Linefeed
Hide Received Data
Save Recv to File...
AutoScroll
Clear
Send Options
ASCII
HEX
Use Escape Chars
Auto Append Bytes
Send from File ...
Cycle
1000
ms
Shortcut
History
Data Log
UartAssist V5.0.3
[2024-07-29 18:01:34.502]# RECV ASCII>
Usart1 Class!
[2024-07-29 18:01:41.653]# SEND ASCII>
Hello
[2024-07-29 18:01:41.710]# RECV ASCII>
Hello
[2024-07-29 18:02:20.339]# SEND ASCII>
Hello Yahboom!
[2024-07-29 18:02:20.398]# RECV ASCII>
Hello Yahboom!
Data Send
1. DCD
2. RXD
3. TXD
4. DTR
5. GND
6.
Clear
Clear
Hello Yahboom!
Send
Ready!
9/6
RX:109
TX:64
Reset