5. Serial communication

5.1. Purpose of the experiment

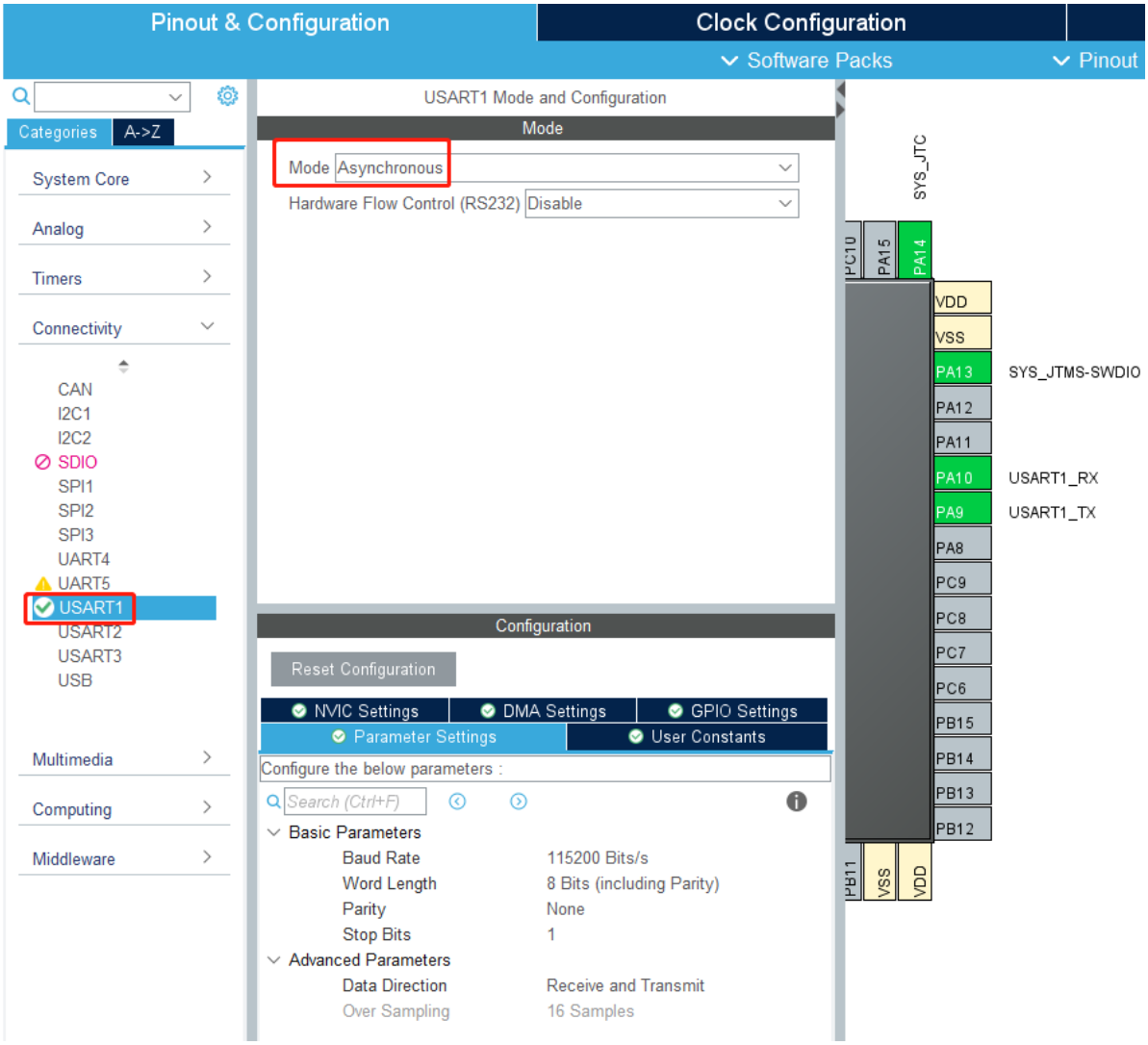
Use the serial port communication of STM32, receive the data of the serial port assistant, and return the received data to the serial port, redefine the printf function.

5.2. Configuration pin information

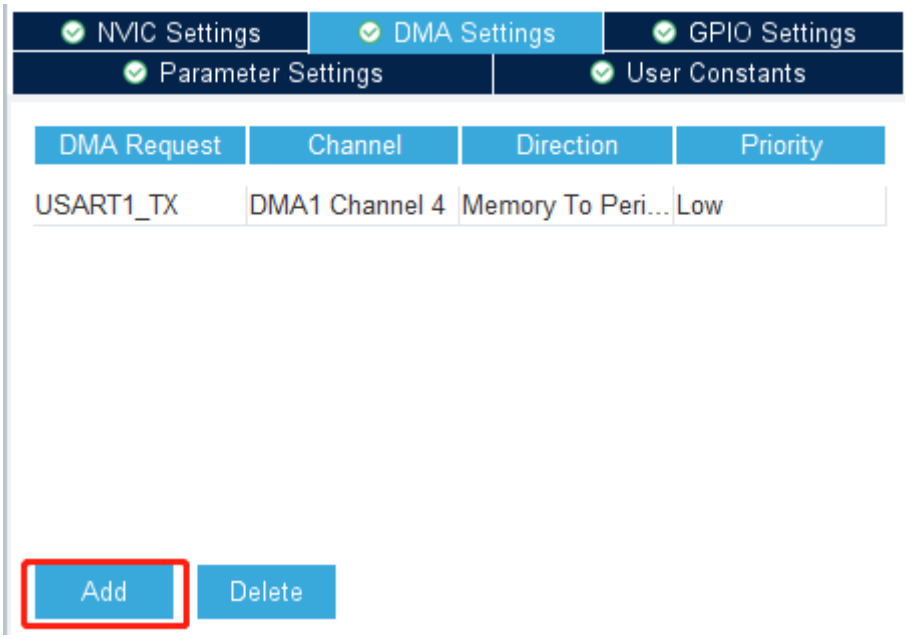
Since each new project needs configuration information, it is more troublesome. Fortunately, STM32CubeIDE provides the function of importing .ioc files, which can help us save time.

1.Import the ioc file from the BEEP project and name it Serial.

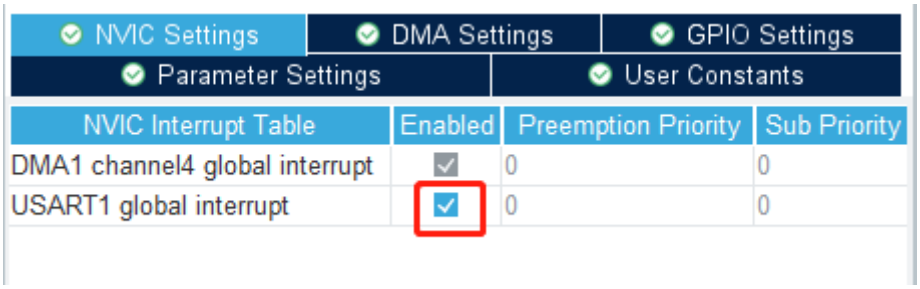
Change the mode of serial port 1 to Asynchronous synchronous communication, the baud rate is 115200, the data width: 8 bits, the test: None, and the stop bit: 1 bit.



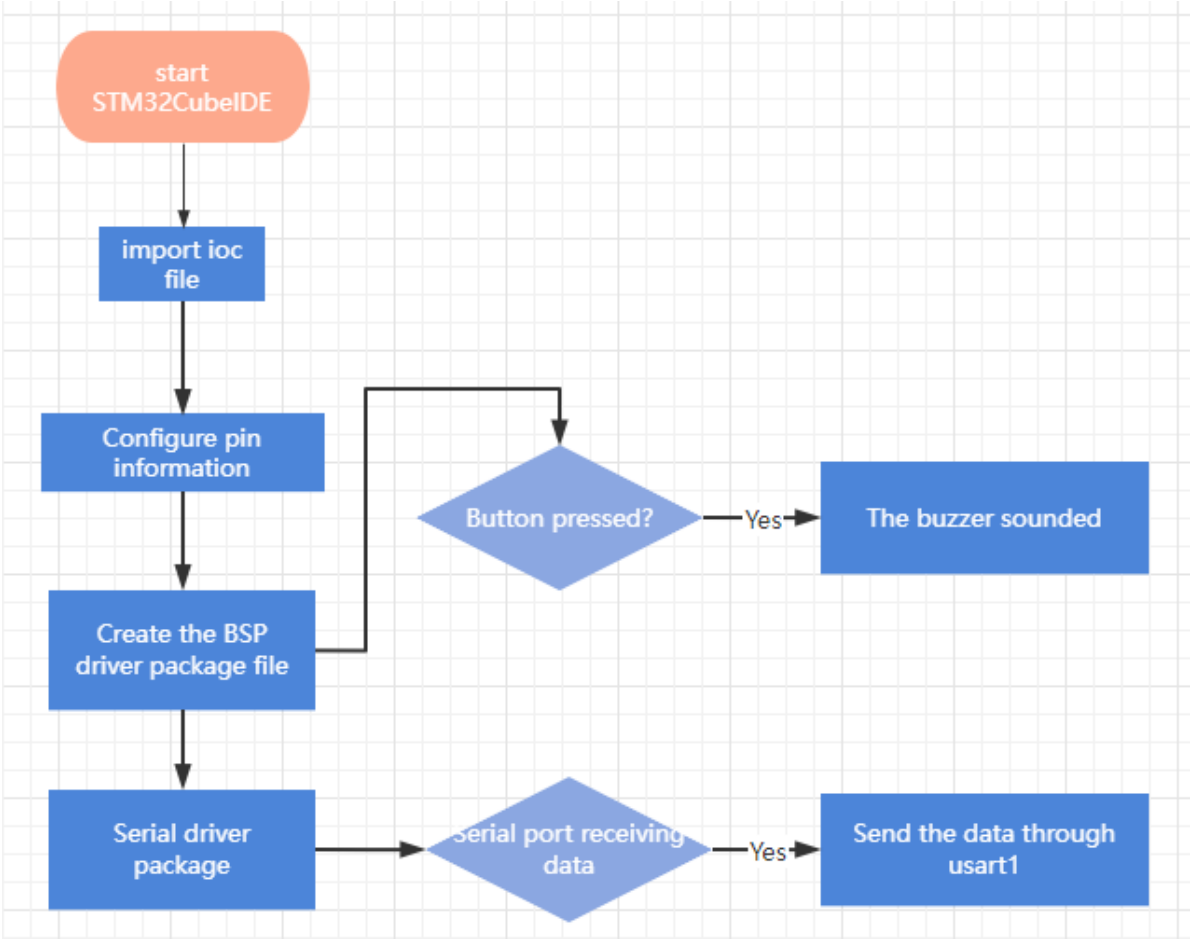
2.Add DMA sending channel



3. Open serial port 1 interrupt settings.



5.3. Analysis of the experimental flow chart



5.4. core code explanation

1. Create new buzzer driver library bsp_uart.h and bsp_uart.c files in BSP. Add the following to bsp_uart.h:

```
void USART1_Init(void);
void USART1_Send_U8(uint8_t ch);
void USART1_Send_ArrayU8(uint8_t *BufferPtr, uint16_t Length);
```

2. Add the following in bsp_uart.c:

USART1_Init(): Initialize the related content of the serial port, open the serial port to receive 1 data.

```
// Initialize USART1  初始化串口1
void USART1_Init(void)
{
    HAL_UART_Receive_IT(&huart1, (uint8_t *)&RxTemp, 1);
}
```

USART1_Send_U8(ch): Serial port 1 sends a byte.

```
// The serial port sends one byte  串口发送一个字节
void USART1_Send_U8(uint8_t ch)
{
    HAL_UART_Transmit(&huart1, (uint8_t *)&ch, 1, 0xFFFF);
}
```

USART1_Send_ArrayU8(BufferPtr,Length): Serial port 1 sends a string of data, BufferPtr is the first address of the data, and Length is the length of the data. ENABLE_UART_DMA is the switch of serial port 1 DMA.

```
// The serial port sends a string of data  串口发送一串数据
void USART1_Send_ArrayU8(uint8_t *BufferPtr, uint16_t Length)
{
    #if ENABLE_UART_DMA
    HAL_UART_Transmit_DMA(&huart1, BufferPtr, Length);
    #else
    while (Length--)
    {
        USART1_Send_U8(*BufferPtr);
        BufferPtr++;
    }
    #endif
}
```

3. Since the serial port 1 receive interrupt is only performed once, it is necessary to call the receive data again after receiving the data. In order to facilitate the test, the serial port is called in an interrupt to send data. In practical applications, data should not be sent in an interrupt. It is time-consuming to send data through the serial port, which may cause problems such as packet loss or even serial port errors.

```
// The serial port receiving is interrupted. Procedure  串口接收完成中断
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
{
    /* Prevent unused argument(s) compilation warning */
    UNUSED(huart);
    /* NOTE : This function should not be modified, when the callback is needed,
              the HAL_UART_RxCpltCallback can be implemented in the user file
     */
    // 测试发送数据，实际应用中不应该在中断中发送数据
    // Test sending data. In practice, data should not be sent during interrupts
    USART1_Send_U8(RxTemp);

    // Continue receiving data  继续接收数据
    HAL_UART_Receive_IT(&huart1, (uint8_t *)&RxTemp, 1);
}
```

4. Redefine printf to use serial port 1 to send data.

```
#ifdef __GNUC__
#define PUTCHAR_PROTOTYPE int __io_putchar(int ch)
#else
#define PUTCHAR_PROTOTYPE int fputc(int ch, FILE *f)
#endif /* __GNUC__ */
PUTCHAR_PROTOTYPE
{
    /* Place your implementation of fputc here */
    /* e.g. write a character to the EVAL_COM1 and Loop until the end of transmission */
    HAL_UART_Transmit(&huart1, (uint8_t *)&ch, 1, 0xFFFF);
    return ch;
}
```

5. In BSP initialization, call USART1_Init() function to request to receive data.

```
// The peripheral device is initialized  外设设备初始化
void Bsp_Init(void)
{
    Beep_On_Time(50);
    USART1_Init();
}
```

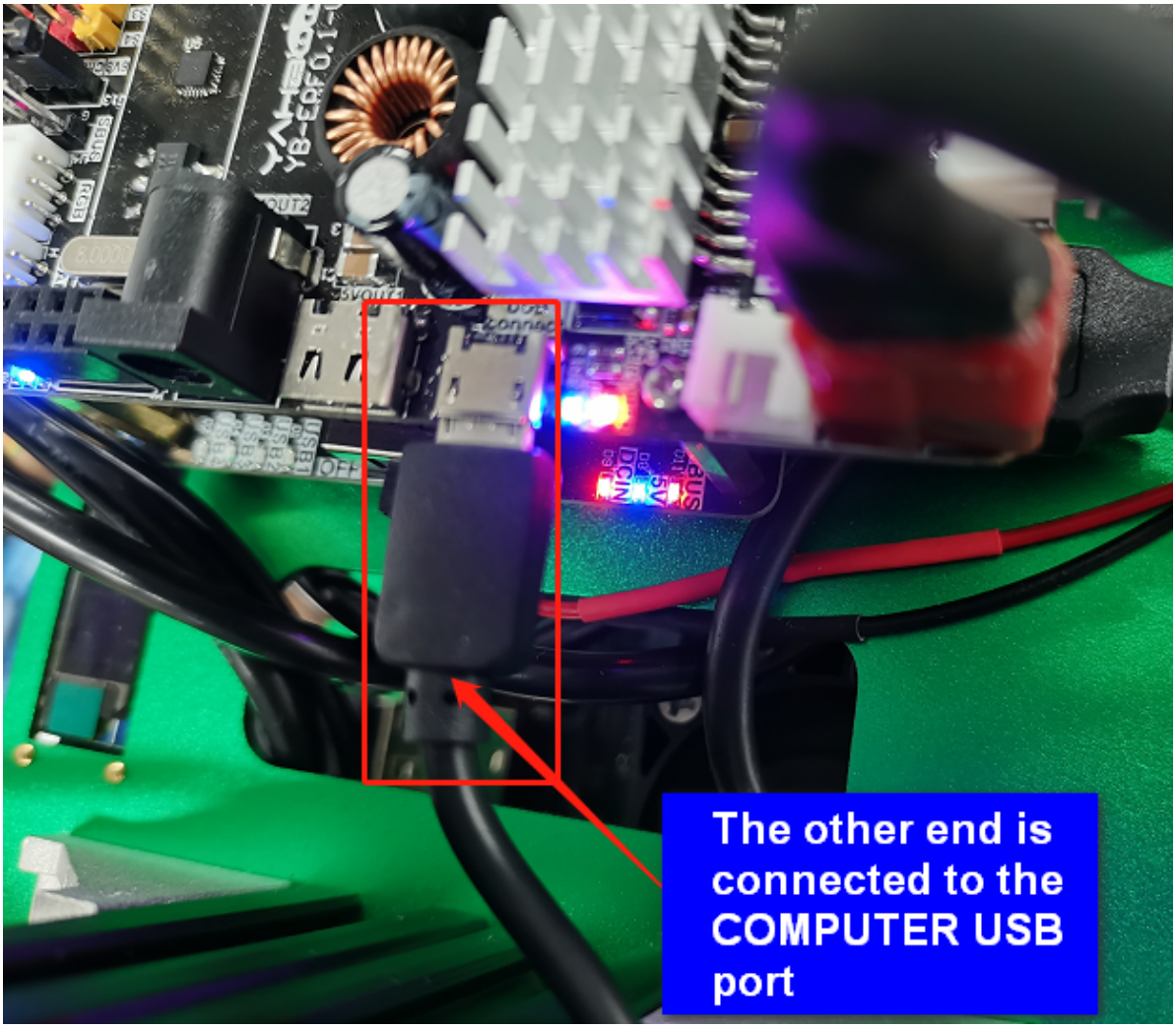
6. After the button is pressed, add the printf() function to print information through serial port 1.

```
// main.c中循环调用此函数，避免多次修改main.c文件。
// This function is called in a loop in main.c to avoid multiple modifications to the main.c file
void Bsp_Loop(void)
{
    // Detect button down events   检测按键按下事件
    if (Key1_State(KEY_MODE_ONE_TIME))
    {
        Beep_On_Time(50);
        static int press = 0;
        press++;
        printf("press:%d\n", press);
    }

    Bsp_Led_Show_State_Handle();
    // The buzzer automatically shuts down when times out   蜂鸣器超时自动关闭
    Beep_Timeout_Close_Handle();
    HAL_Delay(10);
}
```

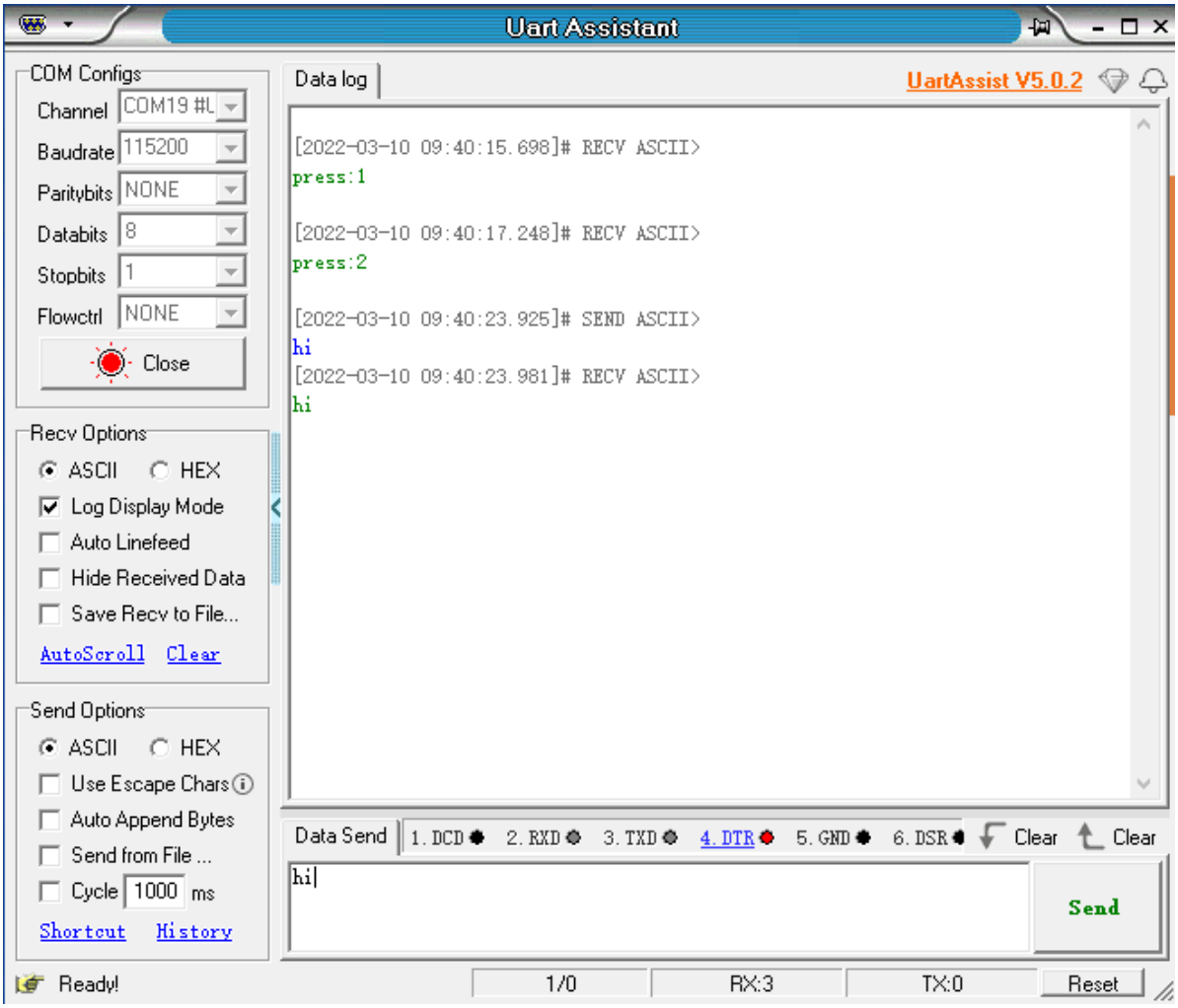
5.5. Hardware connection

The expansion board needs to be connected to the USB port of the computer using a micro-USB data cable. The connection diagram is as follows:



5.6. Experimental effect

After programming the program, the LED light flashes once every 200 milliseconds. Connect the expansion board to the computer through the micro-USB data cable and open the serial port assistant (the specific parameters are shown in the figure below). The buzzer will sound every time the button is pressed. After 50 milliseconds, you can see that the serial port assistant will display press:xx, and each time you press the button xx will automatically increase by 1. The serial port assistant sends the character a, and the expansion board will automatically return the character a. Since the above uses to send data in an interrupt, you cannot send too many characters at a time, otherwise characters will be lost or even serial port errors will occur.


Uart Assistant
COM Configs
Channel COM19 #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 V5.0.2
[2022-03-10 09:40:15.698]# RECV ASCII>
press:1
[2022-03-10 09:40:17.248]# RECV ASCII>
press:2
[2022-03-10 09:40:23.925]# SEND ASCII>
hi
[2022-03-10 09:40:23.981]# RECV ASCII>
hi
Data Send
1. DCD
2. RXD
3. TXD
4. DTR
5. GND
6. DSR
Clear
Clear
hi
Send
Ready!
1/0
RX:3
TX:0
Reset