5. Serial communication

5.1. Experimental purpose

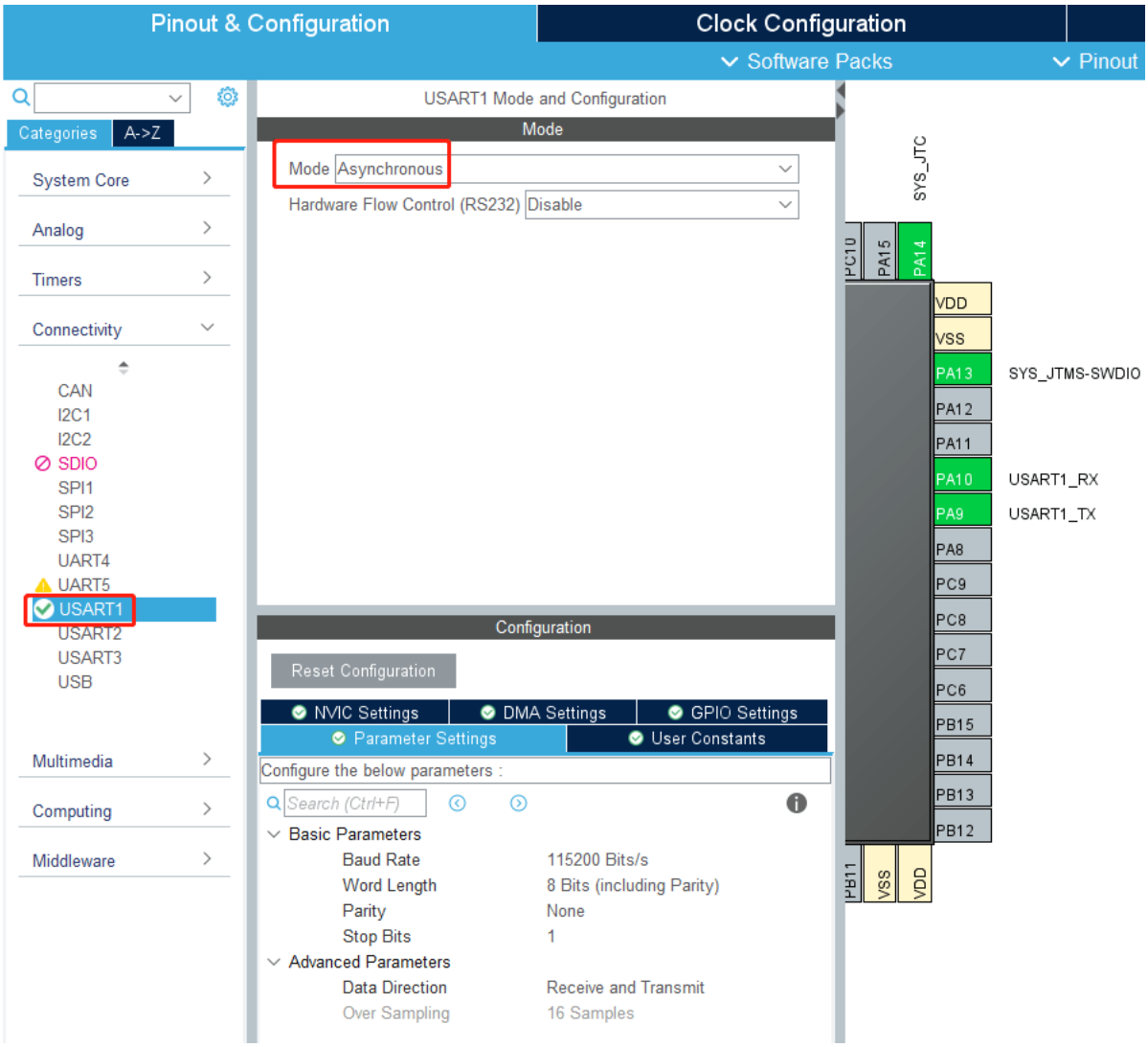
Using STM32 serial communication, receive the data of the serial assistant, and return the received data to the serial port, redefine the printf function.

5.2. Configure pin information

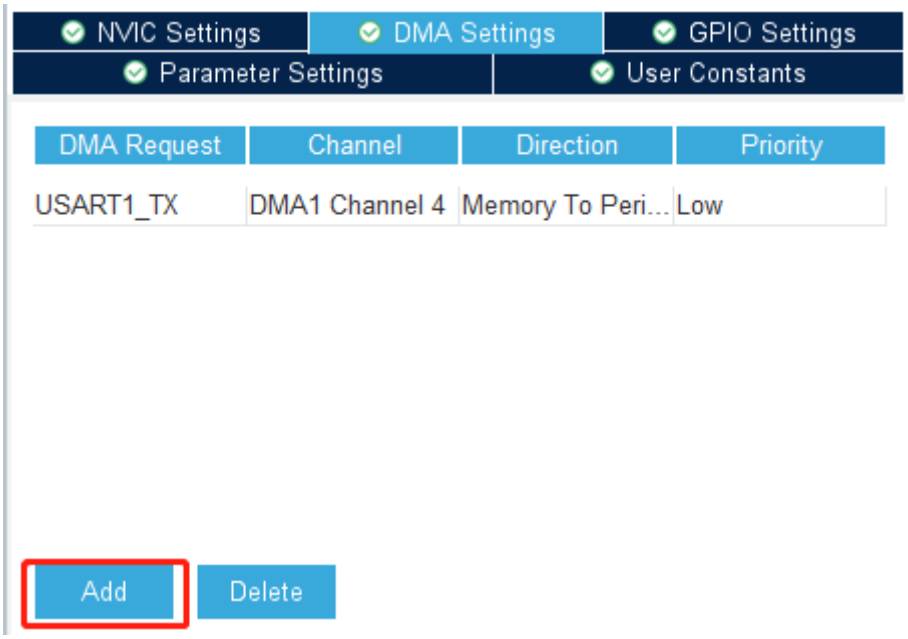
Since each new project needs configuration information, it is troublesome. Fortunately, STM32CubeIDE provides the function of importing.IOC file, which can help us save time.

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

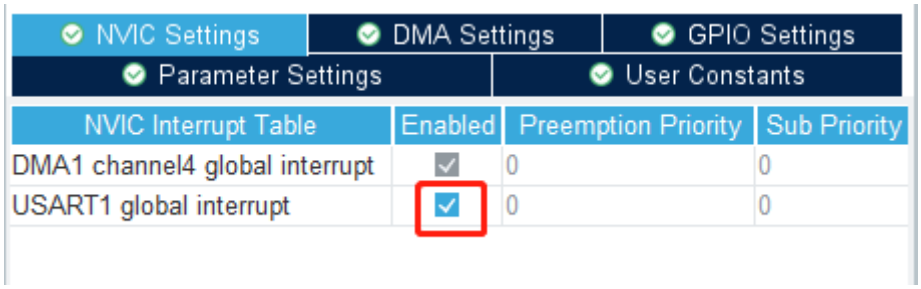
The mode of serial port 1 is modified to Asynchronous synchronous communication, baud rate is 115200, data width: 8 bits, check: None, stop bit: 1 bit.



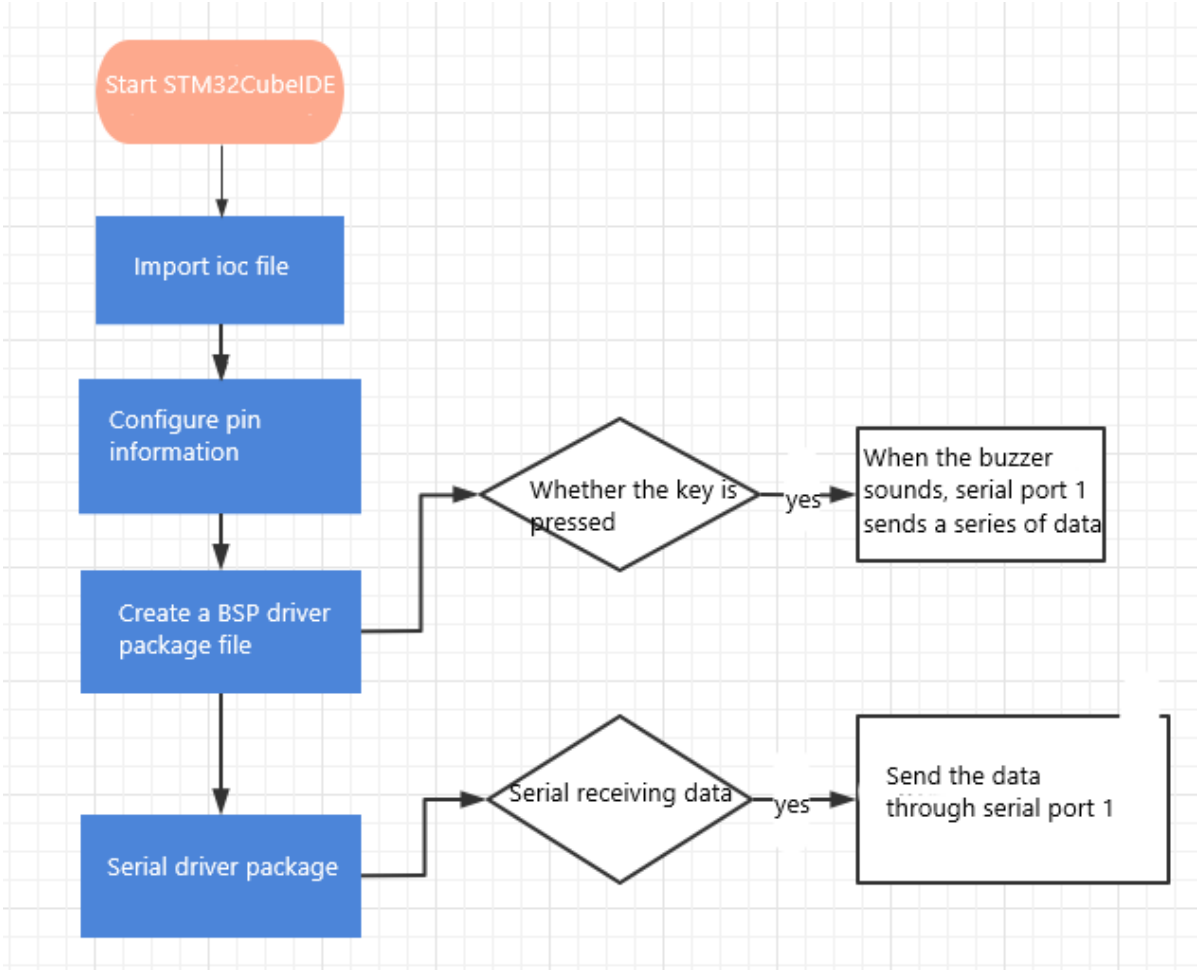
2. Add a DMA send channel



3. Open the serial port 1 Interrupt Settings.



5.3. Analysis of experimental flow chart



5.4. Core code interpretation

1. Create the buzzer driver library bsp_uart.h and bsp_uart.c files in the 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 content to bsp_uart.c:

USART1_Init(): initializes content related to the serial port and enables the serial port to receive one 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 one 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 a switch for serial 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 performed only once, it needs to call receive data again after receiving data. In this case, to facilitate the test, the serial port is interrupted to send data. In actual applications, data should not be sent in the interrupt. Sending data through the serial port takes time, which may cause packet loss or even an error in the serial port.

```
// 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 send data using serial port 1.

```
#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() and 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();
}
```

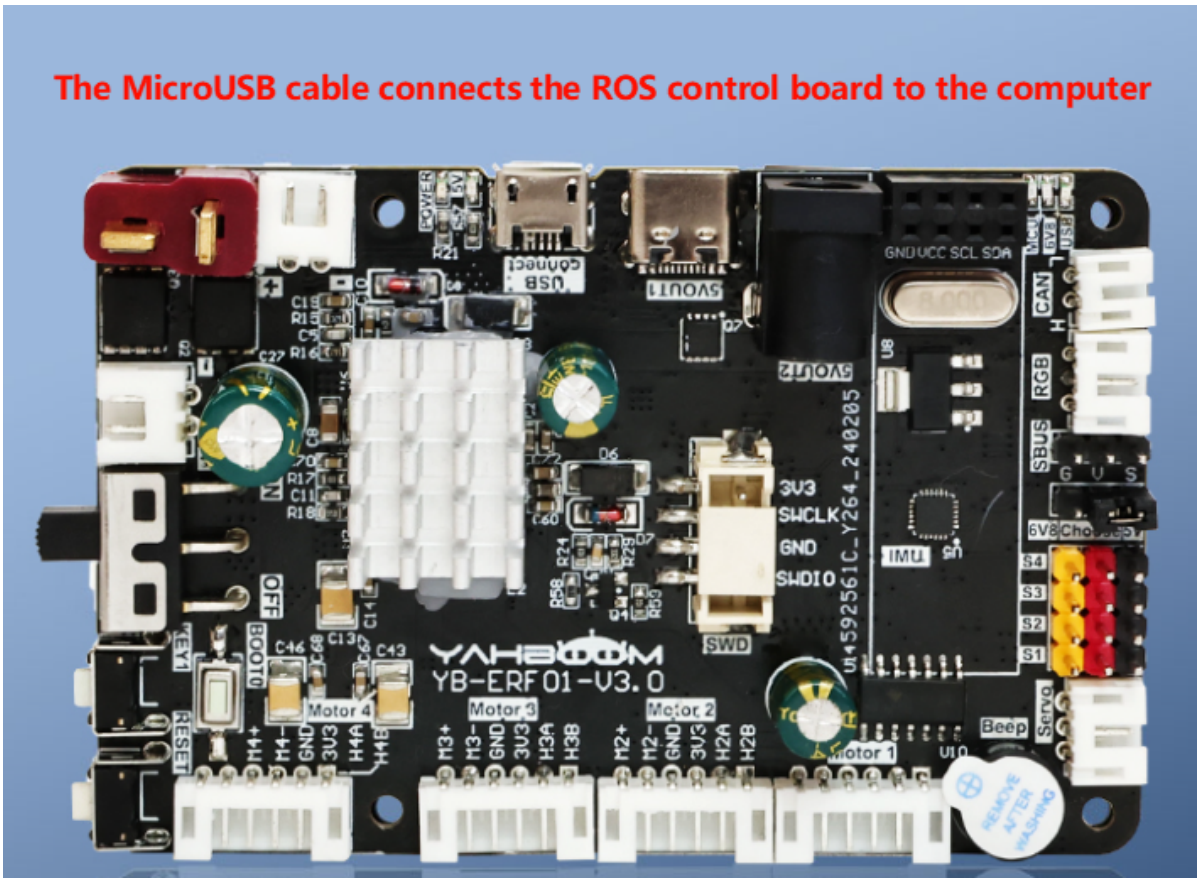
6. After the key is pressed, add the printf() function to print information through serial port 1.

```
// 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 cable. The connection diagram is as follows:



5.6. Experimental effect

After the program is burned, the LED light flashes every 200 milliseconds, connect the expansion board to the computer through the micro-USB data cable, and open the serial assistant (specific parameters are shown in the figure below). Every time you press the button, the buzzer will ring for 50 milliseconds, you can see that the serial assistant will display press:xx, and every time you press the button xx will automatically add 1. The serial assistant sends character a, and the expansion board automatically returns character a. Since the above is used to send data in an interrupt, you cannot send too many characters at once, otherwise you will lose characters or even serial port errors.