5. Serial communication

5.1. Purpose of the experiment

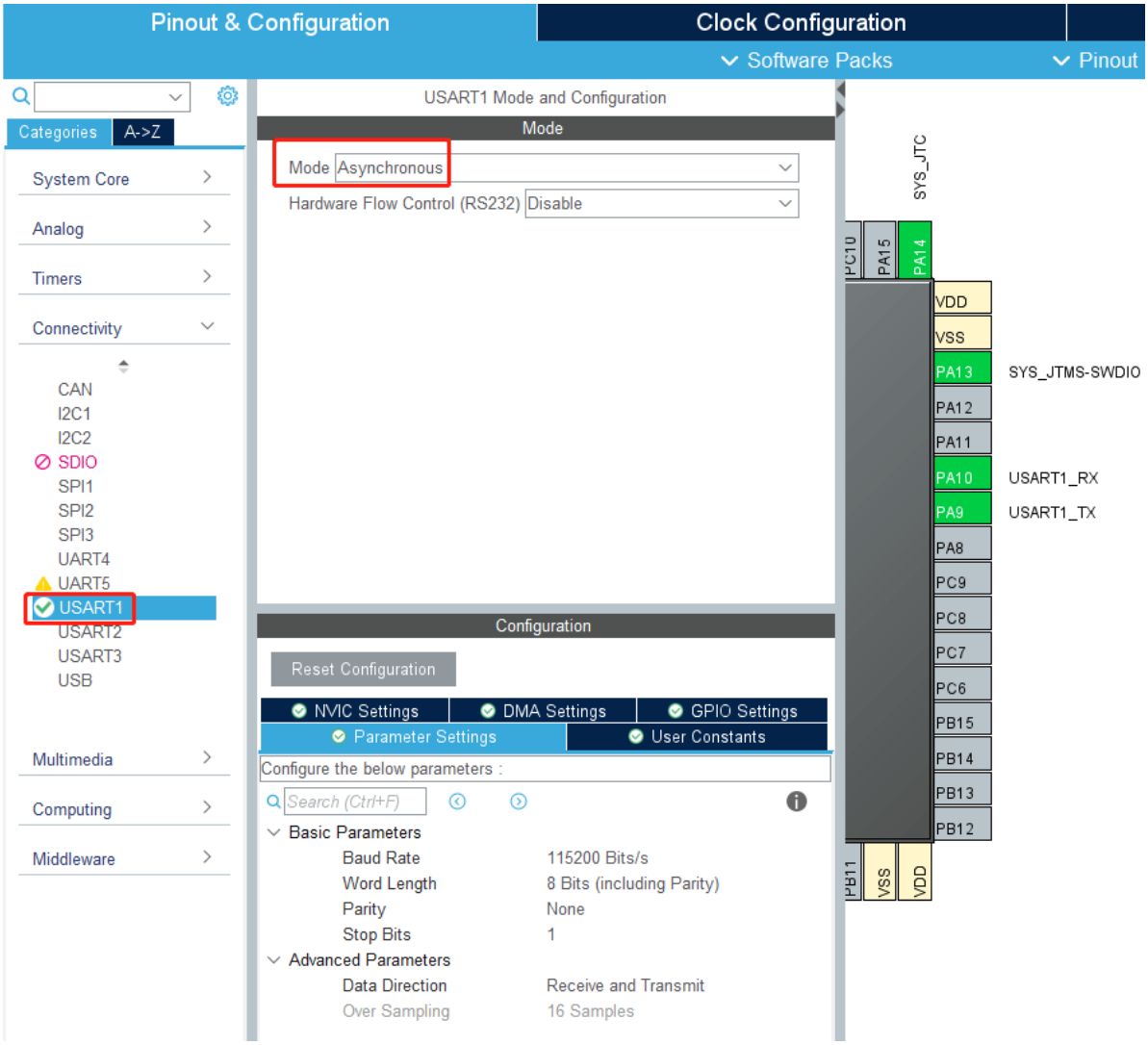
Use STM32's serial communication to receive data from the serial assistant and return the received data to the serial port and redefine the printf function.

5.2 Configuring pin information

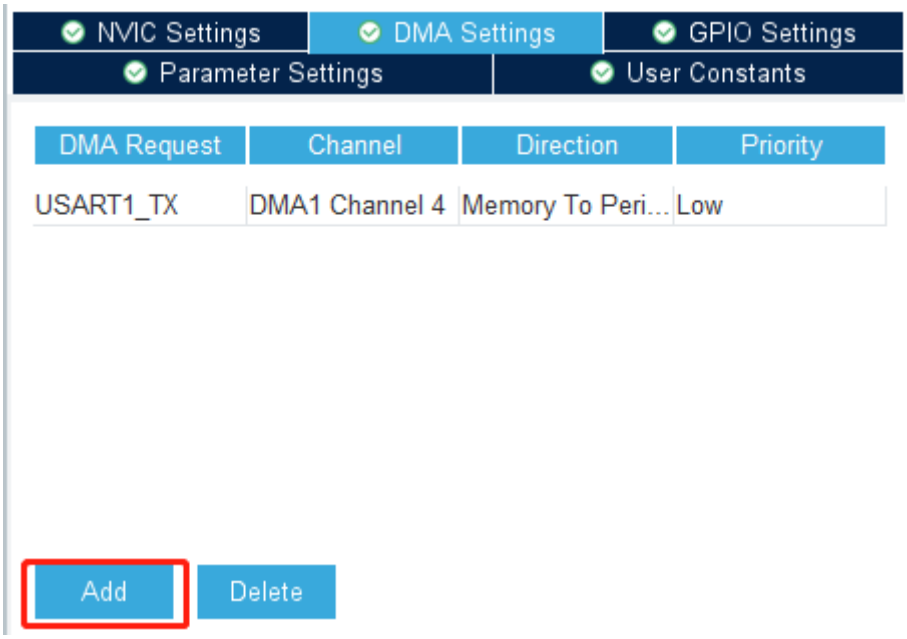
Because every time we create a new project, we need to configure the information, which is quite troublesome. It is good that STM32CubeIDE provides the function of importing .ioc files, which can help us save time.

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

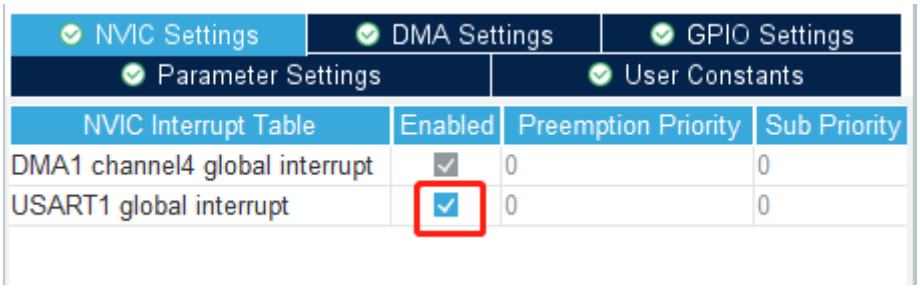
Modify the mode of serial port 1 to Asynchronous synchronous communication, baud rate is 115200, data width: 8 bits, check: None, stop bit: 1 bit.



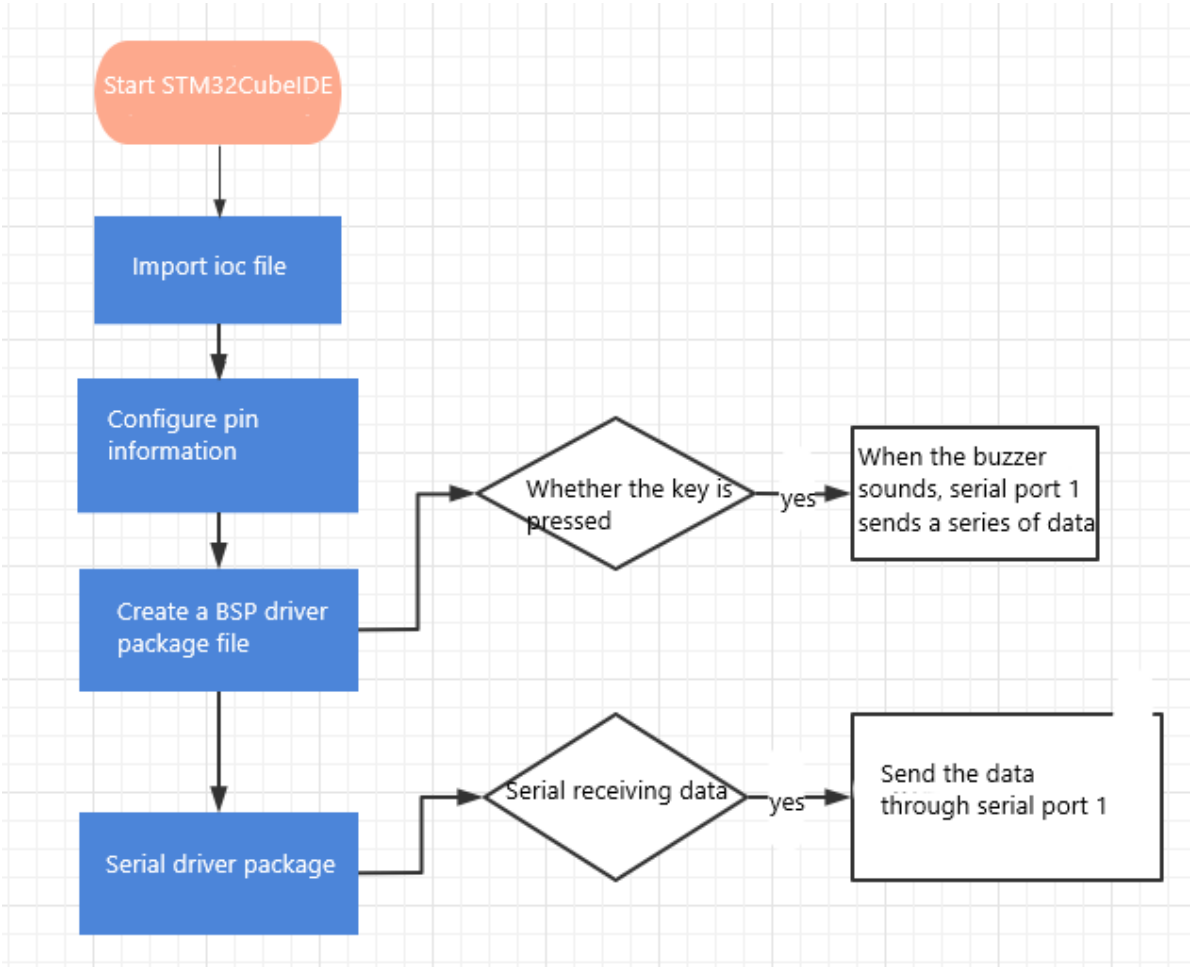
2. Add DMA send channel



3. Open the serial port 1 interrupt setting.



5.3. Experimental flow chart analysis



5.4. Core Code Explanation

1. Create a new buzzer driver library bsp_uart.h and bsp_uart.c file 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 contents in bsp_uart.c:

USART1_Init():Initialize the serial port related contents and turn on 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, 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. As the serial port 1 receive interrupt only once, so after receiving data need to be called again to receive data. Here in order to facilitate the test, so in the interrupt call the serial port to send data, the actual application should not be in the interrupt to send data, the serial port to send data is more time-consuming, which may lead to packet loss or even serial port errors and other issues.

```
// 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 the BSP initialization, call the USART1_Init() function to request receiving data.

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

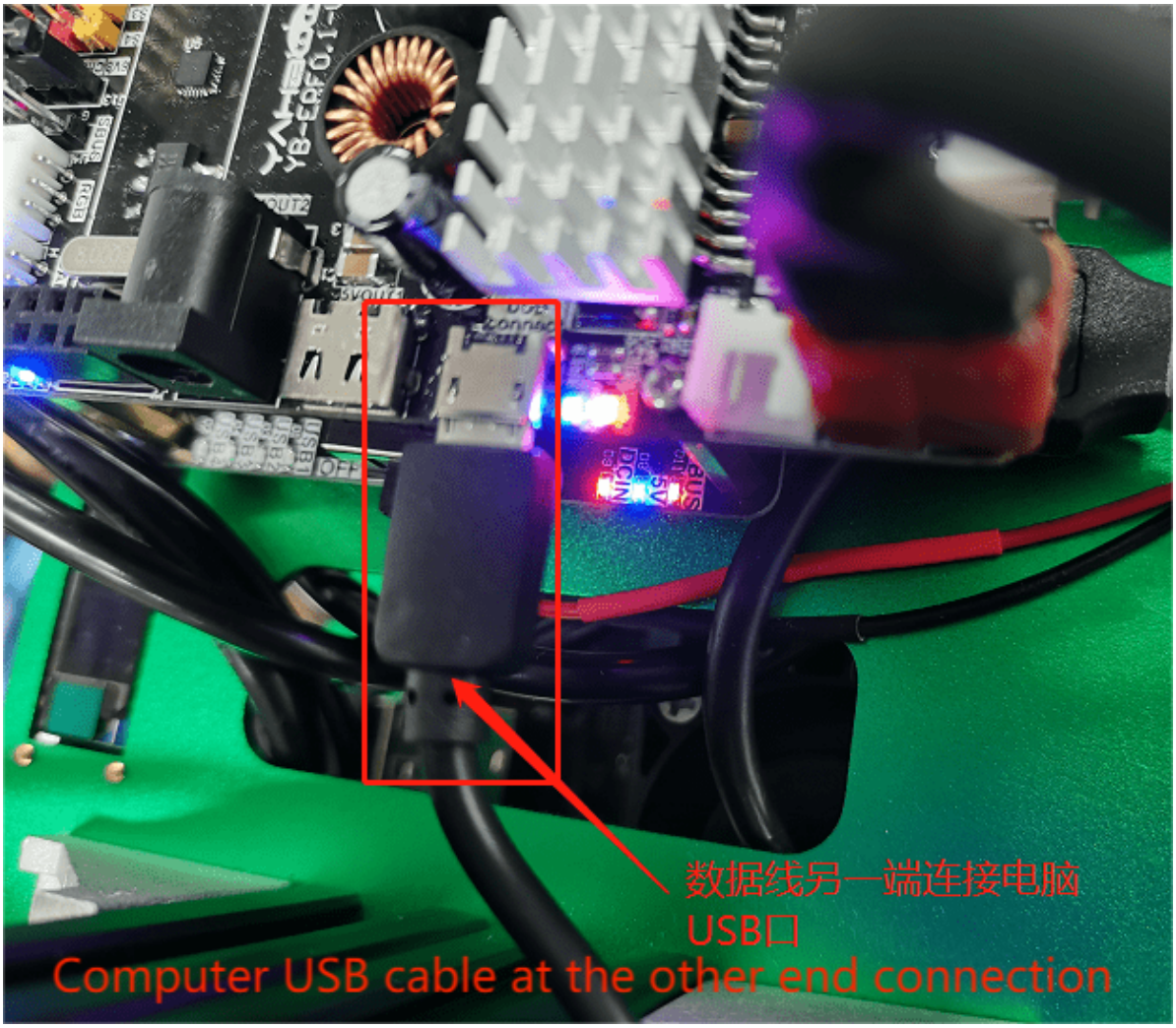
6. Add the printf() function to print information through serial port 1 after a key is pressed.

```
// 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 burning the program, the LED flashes every 200 milliseconds, connect the expansion board to the computer via micro-USB cable and open the serial assistant (the specific parameters are shown in the following figure), every time you press the key, the buzzer will sound for 50 milliseconds, and you can see that the serial assistant will display press:xx, and every time you press the key, the xx is automatically increased by 1. The serial assistant sends the character a and the expansion board will The serial assistant sends the character a and the expansion board will automatically return the character a. Since the above use to send data in interrupt, so can't send too many characters at a time, otherwise there will be character loss or even serial port error.


串口调试助手
串口设置
串口号
COM33 #US
波特率
115200
校验位
NONE
数据位
8
停止位
1
流控制
NONE
关闭
接收设置
ASCII
HEX
按日志模式显示
接收区自动换行
接收数据不显示
接收保存到文件...
自动滚屏
清除接收
发送设置
ASCII
HEX
转义符指令解析
自动发送附加位
打开文件数据源...
循环周期
1000
ms
快捷指令
历史发送
数据日志
用户支持
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
数据发送
1.DCD
2.RXD
3.TXD
4.DTR
5.GND
6.DSR
清除
清除
hi
发送
就绪!
3/1
RX:18
TX:2
复位计数