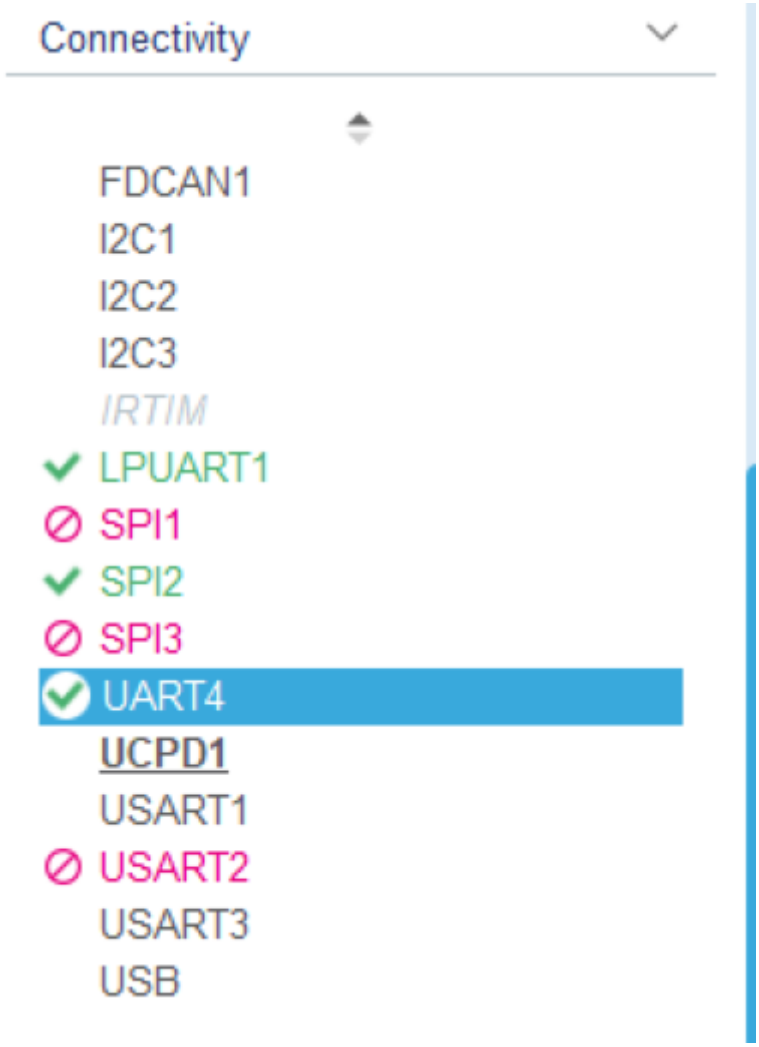
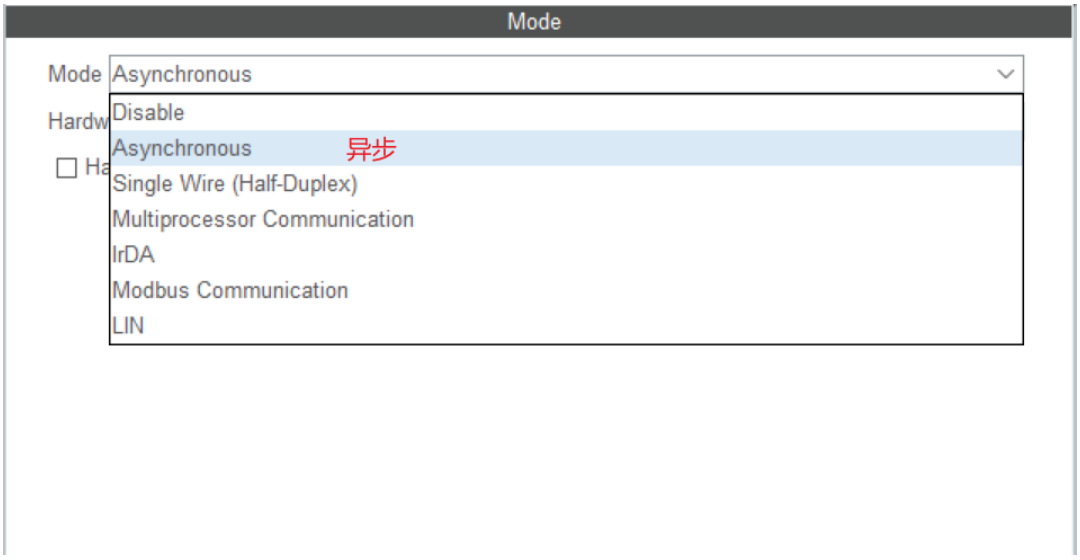
# 实验4：UART/SPI/IIC通信

## 1. UART+DMA

1. Cubemx配置
    1. 选择Connectivity其中可选的UART，以UART4为例，




    2. 选择异步模式

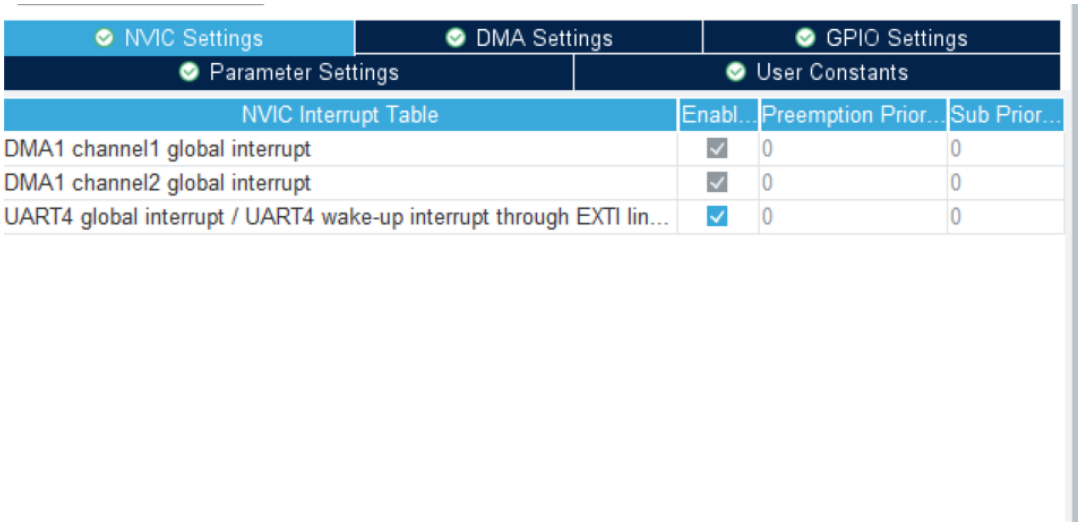



    3. 基本参数配置，例如：115200，8位，无奇偶校验位，一个停止位等

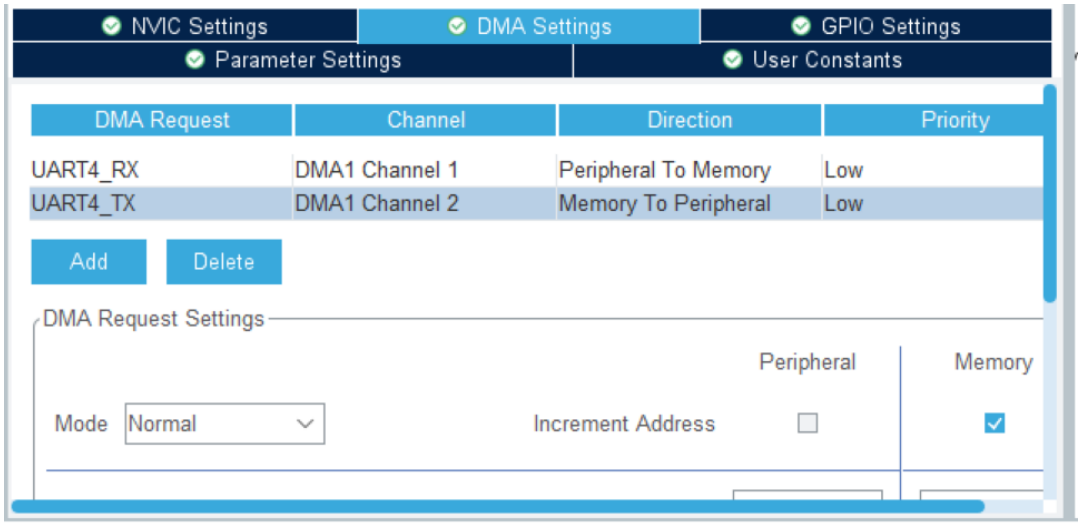
Basic Parameters

| | |
|---|---|
| Baud Rate | 115200 Bits/s |
| Word Length | 8 Bits (including Parity) |
| Parity | None |
| Stop Bits | 1 |

4. 打开串口中断，配置中断优先级组数，并设置串口中断优先级（这里的DMA中断会在下一步配置完DMA后完成）

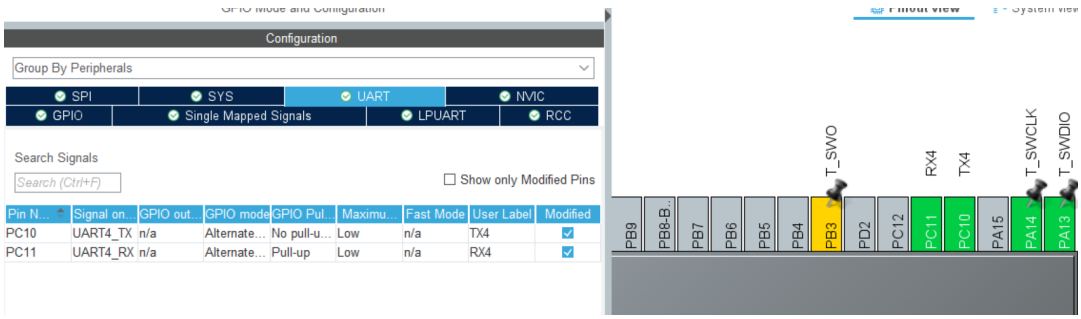



5. 配置DMA




关于下面DMA的参数配置保持默认即可

6. 记得记住引脚哪个分别是TX和RX




7. 重写printf函数代替`HAL_UART_Transmit`

```c
#include <stdio.h>
#include <string.h>
#pragma import(__use_no_semihosting)

struct __FILE
```

```
{
    int handle;
};                                                  // 标准库需要的支持函数

FILE __stdout;                                      // FILE在stdio.h文件
void _sys_exit(int x)
{
    x = x;                                          // 定义_sys_exit()以避免使用半主机模式
}
int fputc(int ch, FILE *f)                          // 重写fputc函数，使printf的输出由UART1实现，  这里使用USART1
{
    // 注意，不能使用HAL_UART_Transmit_IT()，机制上会冲突；因为调用中断发送函数后，如果上次发送还在进行，就会直接返回！它不会继续等待，也不会数据填入队列排队发送
    HAL_UART_Transmit(&huart4, (uint8_t *)&ch, 1, 0x02);    // 使用HAL_UART_Transmit，相等于USART4->DR = ch，函数内部加了简单的超时判断(ms)，防止卡死
    return ch;
}
```

8. 定义结构体方便数据收发

```
typedef struct{
        uint16_t ReceiveNum;
        uint8_t ReceiveData[512];
        uint8_t BuffTemp[512];
}xUSART_Typedef;
```

9. 重写DMA中断函数

```
void HAL_UARTEx_RxEventCallback(UART_HandleTypeDef *huart, uint16_t Size)
{
    if (huart == &huart4)// 判断串口
    {
        __HAL_UNLOCK(huart);// 解锁串口状态

        USART4.ReceiveNum  = Size;  // 把接收字节数，存入结构体USART4.ReceiveNum，以备使用
        memset(USART4.ReceiveData, 0, sizeof(USART4.ReceiveData));//清0前一帧的接收数据
        memcpy(USART4.ReceiveData, USART4.BuffTemp, Size); //存储数据到USART4.ReceiveData[]
        HAL_UARTEx_ReceiveToIdle_DMA(&huart4, USART4.BuffTemp, sizeof(USART4.BuffTemp)); // 再次开启DMA空闲中断；每当接收完指定长度，或者产生空闲中断时，就会来到这个
    }
}
```

10. 一个随意的主程序（能完成基本通信展示作用即可）

```c
  while (1)
  {

HAL_UARTEx_ReceiveToIdle_DMA(&huart4,USART4.BuffTemp,sizeof(USART4.BuffTemp));
        if (USART4.ReceiveNum > 0)
        {
            int num1, num2;
            sscanf((char*)USART4.ReceiveData, "%d %d", &num1, &num2);
            num1++;num2++;
            char sendData[512];
            snprintf(sendData, sizeof(sendData), "%d %d", num1, num2);
            printf("%s ", sendData);
            USART4.ReceiveNum = 0;
            memset(USART4.ReceiveData, 0, sizeof(USART4.ReceiveData));
        }
        HAL_Delay(1000);
  }
```

# 注：引脚可在datasheet查看

即提供的`um2505-stm32g4-nucleo64-boards-mb1367-stmicroelectronics.pdf`

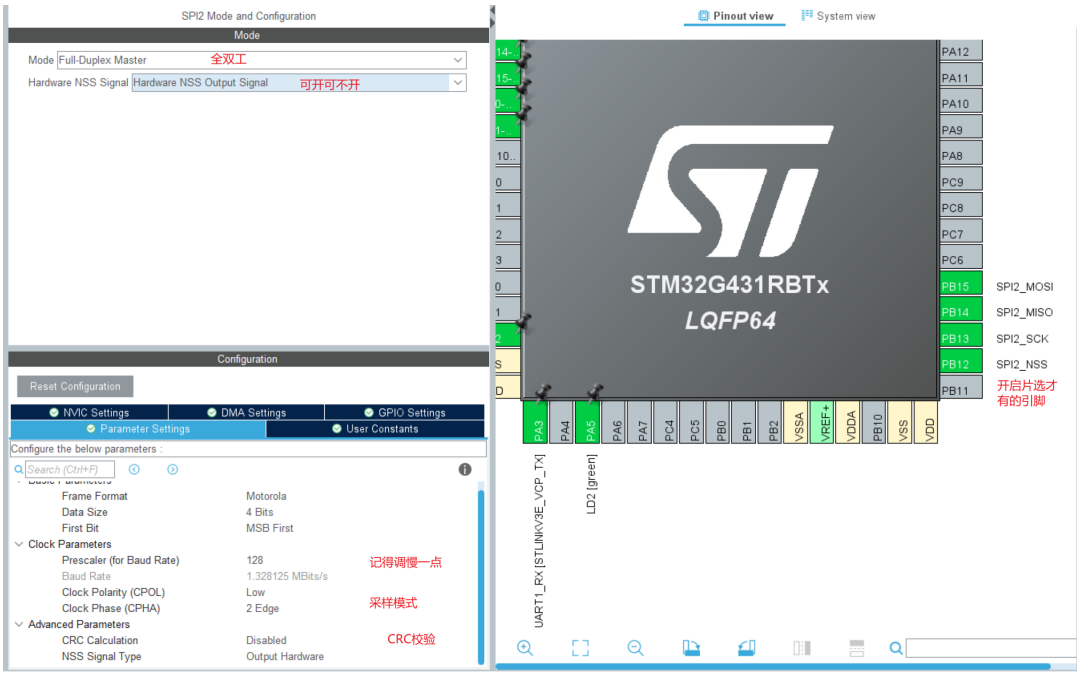
## 可能使用的库函数

1. `HAL_UART_Transmit()`：这个函数用于发送数据。接口参数主要包括UART_HandleTypeDef结构体指针、要发送的数据缓冲区指针、要发送的数据字节数、超时时间。
2. `HAL_UART_Receive()`：这个函数用于接收数据。接口参数主要包括UART_HandleTypeDef结构体指针、要接收的数据缓冲区指针、要接收的数据字节数、超时时间。
3. `HAL_UART_Transmit_IT()`：这个函数用于中断方式发送数据。参数与`HAL_UART_Transmit()`相同。
4. `HAL_UART_Receive_IT()`：这个函数用于中断方式接收数据。参数与`HAL_UART_Receive()`相同。
5. `HAL_UART_Transmit_DMA()`：这个函数用于DMA方式发送数据。参数与`HAL_UART_Transmit()`相同。
6. `HAL_UART_Receive_DMA()`：这个函数用于DMA方式接收数据。参数与`HAL_UART_Receive()`相同。

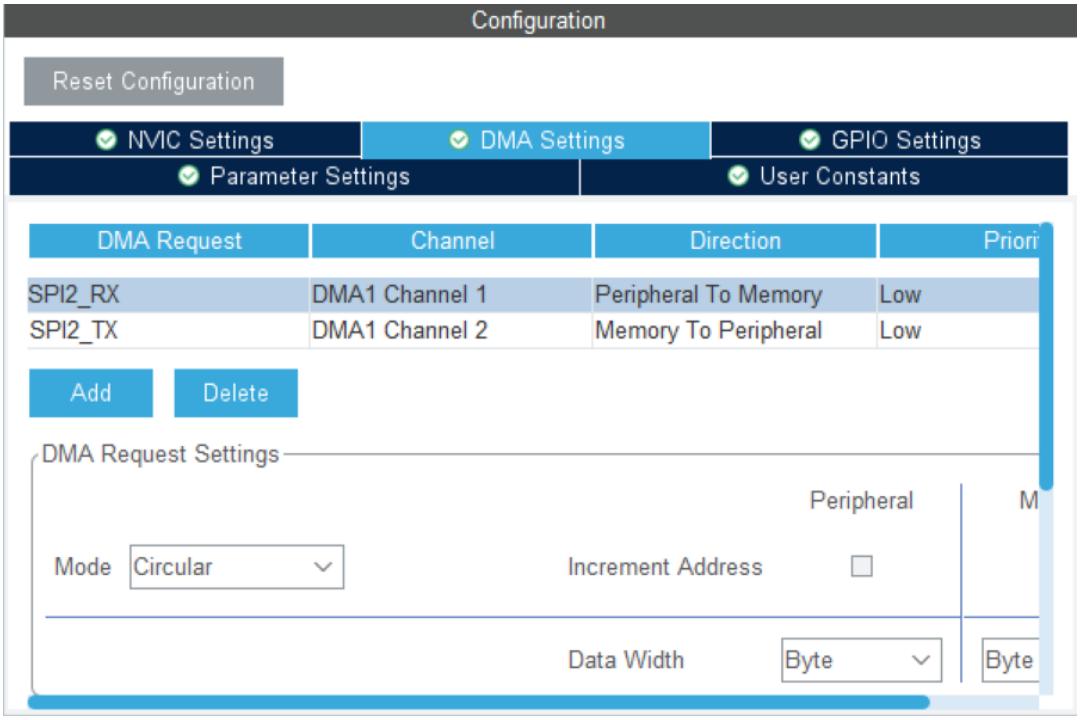
# 2. SPI

可参考CSDN：STM32 SPI 双机通信，SPI从机模式使用

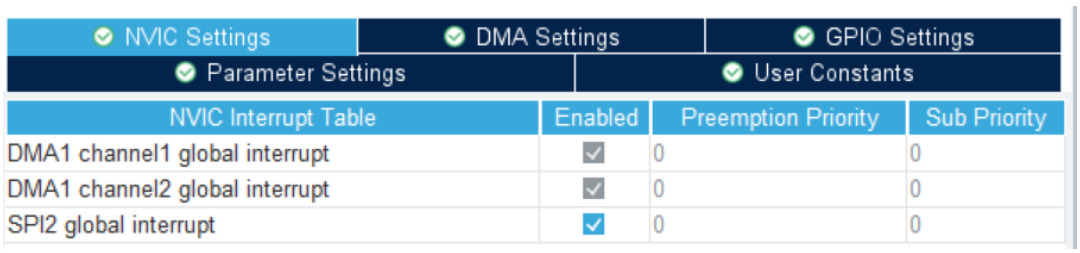
1. Cubemx配置
    1. 配置SPI



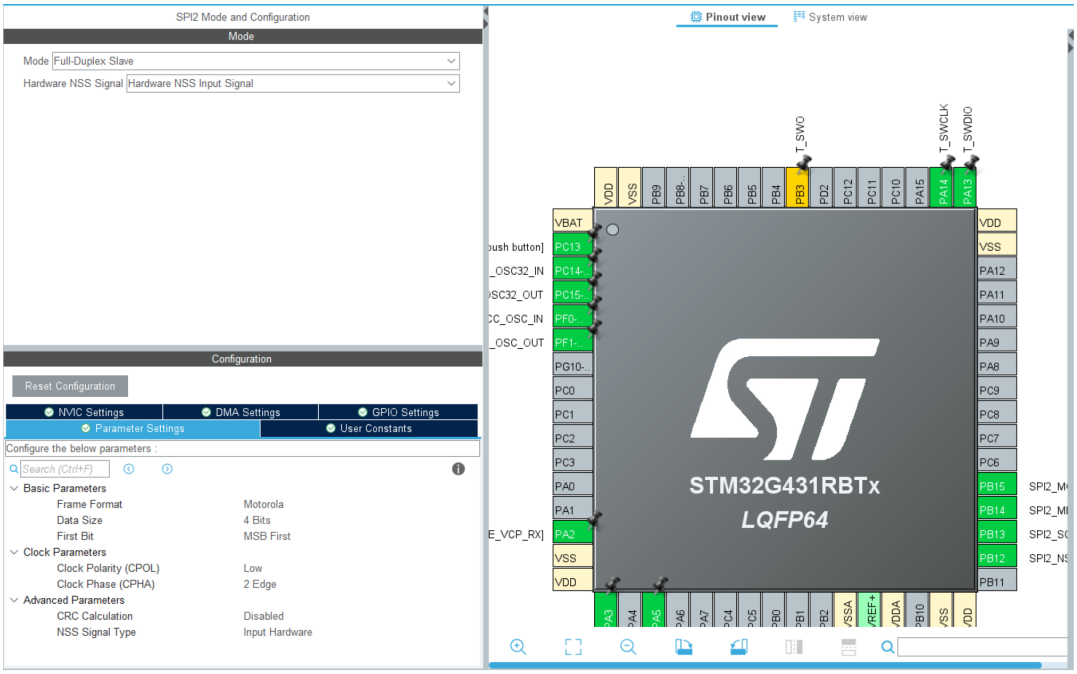

2. 配置DMA




3. 配置中断




4. 从机配置参数记得与主机一致




## 可能使用的库函数

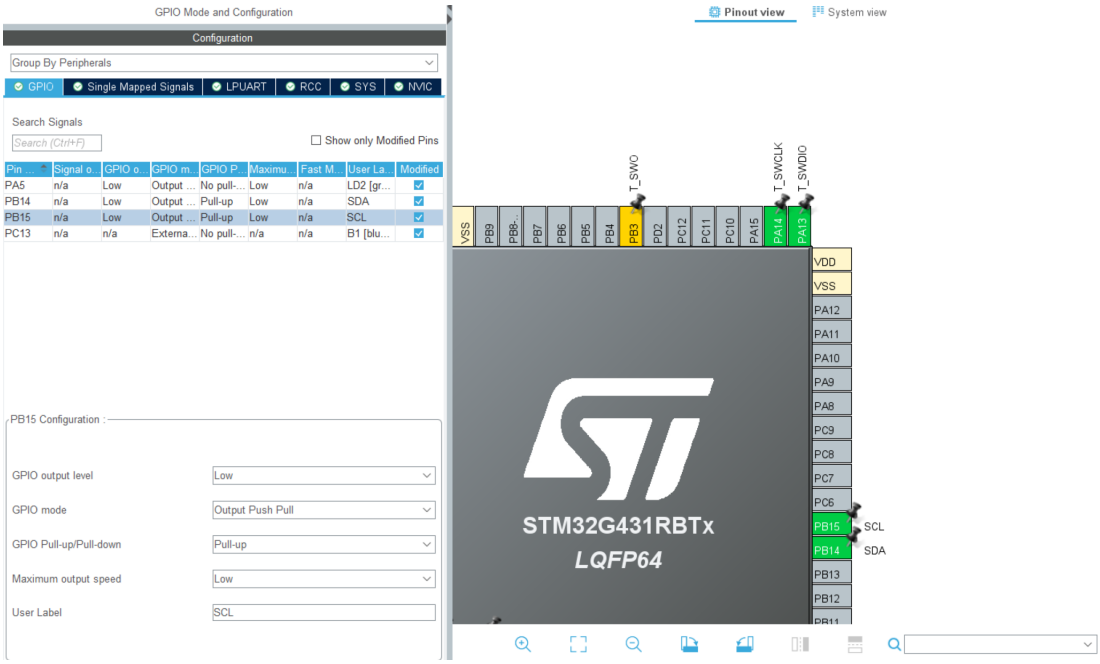
| 函数名称 | 描述 | 参数 |
| --- | --- | --- |
| `HAL_SPI_Transmit` | 同步发送数据 | - `SPI_HandleTypeDef *hspi`：SPI句柄<br>- `uint8_t *pData`：发送数据缓冲区<br>- `uint16_t Size`：发送数据大小<br>- `uint32_t Timeout`：超时时间 |
| `HAL_SPI_Receive` | 同步接收数据 | - `SPI_HandleTypeDef *hspi`：SPI句柄<br>- `uint8_t *pData`：接收数据缓冲区<br>- `uint16_t Size`：接收数据大小<br>- `uint32_t Timeout`：超时时间 |
| `HAL_SPI_TransmitReceive` | 同步发送和接收数据 | - `SPI_HandleTypeDef *hspi`：SPI句柄<br>- `uint8_t *pTxData`：发送数据缓冲区<br>- `uint8_t *pRxData`：接收数据缓冲区<br>- `uint16_t Size`：数据大小<br>- `uint32_t Timeout`：超时时间 |

| 函数名称 | 描述 | 参数 |
| --- | --- | --- |
| `HAL_SPI_Transmit_IT` | 中断模式发送数据 | - `SPI_HandleTypeDef *hspi`：SPI句柄<br>- `uint8_t *pData`：发送数据缓冲区<br>- `uint16_t Size`：发送数据大小 |
| `HAL_SPI_Receive_IT` | 中断模式接收数据 | - `SPI_HandleTypeDef *hspi`：SPI句柄<br>- `uint8_t *pData`：接收数据缓冲区<br>- `uint16_t Size`：接收数据大小 |
| `HAL_SPI_TransmitReceive_IT` | 中断模式发送和接收数据 | - `SPI_HandleTypeDef *hspi`：SPI句柄<br>- `uint8_t *pTxData`：发送数据缓冲区<br>- `uint8_t *pRxData`：接收数据缓冲区<br>- `uint16_t Size`：数据大小 |
| `HAL_SPI_Transmit_DMA` | DMA模式发送数据 | - `SPI_HandleTypeDef *hspi`：SPI句柄<br>- `uint8_t *pData`：发送数据缓冲区<br>- `uint16_t Size`：发送数据大小 |
| `HAL_SPI_Receive_DMA` | DMA模式接收数据 | - `SPI_HandleTypeDef *hspi`：SPI句柄<br>- `uint8_t *pData`：接收数据缓冲区<br>- `uint16_t Size`：接收数据大小 |
| `HAL_SPI_TransmitReceive_DMA` | DMA模式发送和接收数据 | - `SPI_HandleTypeDef *hspi`：SPI句柄<br>- `uint8_t *pTxData`：发送数据缓冲区<br>- `uint8_t *pRxData`：接收数据缓冲区<br>- `uint16_t Size`：数据大小 |

# 3. 软件IIC

1. Cubemx配置

   IIC需要上拉电阻




2. 根据IIC通信协议进行程序编写，模拟通信过程