片外存储器 FRAM

1. 实验目的

- 掌握 FRAM 铁电存储器原理和 SPI 总线的原理及工作模式。
- 掌握 STC15W4K32S4 系列单片机 SPI 外设相关的寄存器配置及程序设计。

2. 实验内容

- 编写程序实现模拟 SPI 对片外 FRAM 读写的程序设计。
- 编写程序实现硬件 SPI 对片外 FRAM 读写的程序设计。

3. 硬件设计

3.1. FRAM 铁电存储器介绍

FRAM（全称是 Ferroelectric Random Access Memory）铁电存储器，该存储器能兼容 RAM 的一切功能，并且和 ROM 技术一样，是一种非易失性的存储器。可以说铁电存储器在这两类存储类型间搭起了一座跨越沟壑的桥梁——一种非易失性的 RAM。

铁电存储器的工作原理是：当在铁电晶体材料上加入电场，晶体中的中心原子会沿着电场方向运动，达到稳定状态。晶体中的每个自由浮动的中心原子只有 2 个稳定状态，一个记为逻辑中的 0，另一个记为 1。中心原子能在常温、没有电场的情况下，停留在此状态达 100 年以上。铁电存储器不需要定时刷新，能在断电情况下保存数据。由于整个物理过程中没有任何原子碰撞，铁电存储器有高速读写、超低功耗和无限次写入等优点。

说到 FRAM 铁电存储器，就必须要介绍下美国 Ramtron 公司，该公司成立于 1984 年，是一家研究和开发铁电技术用于半导体存储器的公司。2012 年 9 月，Ramtron 公司被美国著名半导体公司赛普拉斯（Cypress）并购。2020 年 4 月，infineon 英飞凌完成了总价值 90 亿欧元（合人民币 693 亿元）对 Cypress 赛普拉斯半导体公司的收购。

Ramtron 公司的 FRAM 主要包括两大类：串行 FRAM 和并行 FRAM。其中串行 FRAM 又分 I2C 总线方式的 FM24xx 系列和 SPI 总线方式的 FM25xx 系列。艾克姆科技 FRAM 选择的存储器芯片是 FM25CL64B 芯片（SPI 总线方式）。

图 1：外部 FRAM 存储器实物图

✧ 注：Ramtron 公司的商业 FRAM 产品全部由美国和日本的战略代工厂所制造。

3.2. SPI 总线原理

SPI 是串行外设接口(Serial Peripheral Interface)的缩写，是一种高速、全双工、同步的通信总线。SPI 是 Motorola 公司推出的一种同步串行接口技术，SPI 由一个主设备和一个或多个从设备组成，在一次数据传输过程中，接口上只能有一个主机和一个从机通信。

SPI 总线的优点是操作简单、数据传输速率较高、全双工，缺点是只支持单个主机、没有指定的流控制、没有应答机制确认是否接收到数据。

3.2.1. 接口信号定义

SPI 总线接口包括以下四种信号：

1) MOSI（Master Output，Slave Input）：主器件数据输出，从器件数据输入。
2) MISO（Master Input， Slave Output）：主器件数据输入，从器件数据输出。
3) SCK（Serial Clock）：有时也称为 SCLK，时钟信号，由主器件产生。
4) CS（Chip select）：有时也称为 SS，从器件使能信号，由主器件控制，实际使用时，经常用 GPIO 来代替。

SPI 总线支持连接多个从机，如下图所示，SPI 主机通过连接到从机的片选信号使能/禁止从机，并且同时只能使能一个从机，因此总线里面有多少个从机，就需要多少个片选信号。当 SPI 主机需要和总线中某个从机进行通信时，主机会拉低对应的 CS 信号使能该从机，之后发起通信，通信完成后，拉高 CS 信号，解除总线的占用。

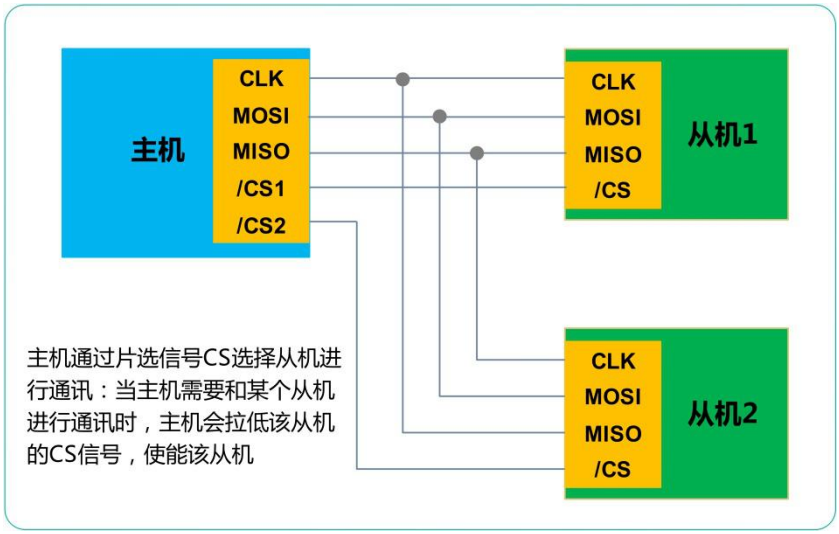

图 2：SPI 总线结构

对于 SPI 总线，我们还需要深刻理解下面几个知识点。

■ **硬件片选和软件片选的区别**

所谓硬件片选指的是 SPI 本身具有片选信号，当我们通过 SPI 发送数据时，SPI 外设自动拉低 CS 信号使能从机，发送完成后自动拉高 CS 信号释放从机，这个过程是不需要软件操作的。而软件片选则是需要使用 GPIO 作为片选信号，SPI 在发送数据之前，需要先通过软件设置作为片选信号的 GPIO 输出低电平，发送完成之后再设置该 GPIO 输出高电平。

■ **SPI 总线是回环结构**

SPI 是一个环形总线结构，如下图所示，主设备和从设备构成一个环形。在时钟 SCK 的作用下，主设备发送一个位到从设备，因为是环形结构，所以从设备必定会同时传送一个位到主设备。同样，主设备向从设备发送一个字节，从设备也必定会同时传送一个字节到主设备。理解了环形结构，就很容易理解下面几点：

- SPI 是全双工，同步的通信总线。
- 主设备向从设备发送数据时，无论我们需不需要从设备返回数据，从设备都会返回数据。
- 主设备从从设备读取一个字节数据时，为什么需要写一个字节数据到从设备：因为是环形结构，不写数据过去，对方的数据就不会被移位过来。

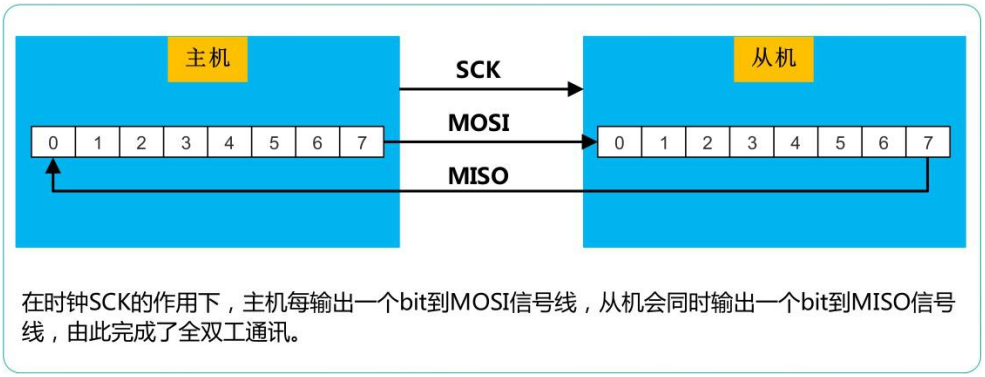

图 3：SPI 数据传输示意图

■ **SPI 主机和从机之间连接时信号不需要交叉**

SPI 主机和从机连接时，MOSI 和 MISO 信号是不需要交叉连接的，因为 MOSI 本身就表示了主机输出、从机输入，MISO 表示主机输入、从机输出，因此不能交叉连接。

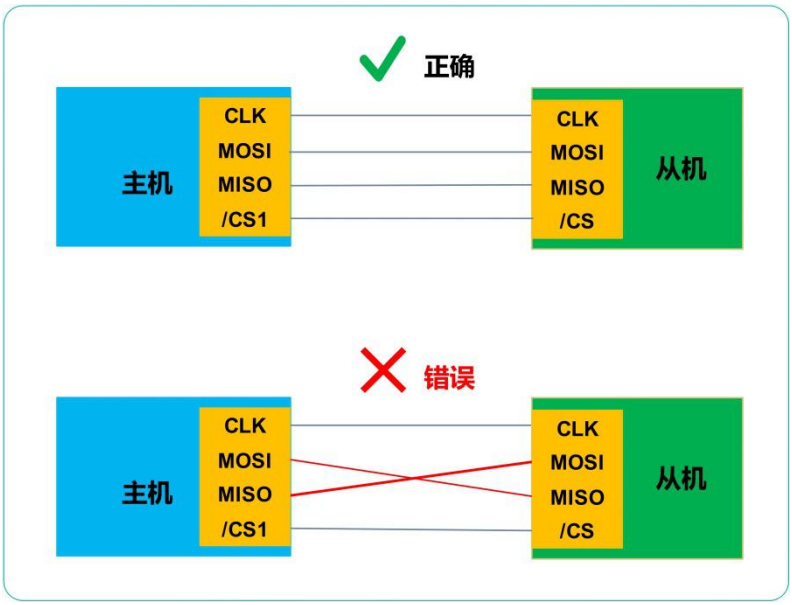

图 4：MOSI 和 MISO 不能交叉连接

3.2.2. SPI 的 4 种工作模式

SPI 总线共有 4 种工作模式：模式 0~模式 3，这 4 种工作模式是由时钟相位和时钟极性确定的。

1. 时钟相位 CPOL（Clock polarity）：SPI 总线空闲时，时钟信号 SCLK 的电平称为时钟极性，有以下两种模式：

- CPOL=0：SPI 总线空闲时，时钟信号为低电平。
- CPOL=1：SPI 总线空闲时，时钟信号为高电平。

2. 时钟极性 CPHA（Clock phase）：SPI 在时钟信号 SCLK 第几个边沿开始采样数据，有以下两种模式：

- CPHA=0：在第 1 个时钟边沿进行数据采样。
- CPHA=1：在第 2 个时钟边沿进行数据采样。

时钟极性 CPOL 时钟相位 CPHA 各有 2 种模式，他们两两组合就形成了 SPI 的 4 种工作模式，如下表所示。

表 1：SPI 总线的 4 种工作模式

序号	SPI 工作模式	描述	备注
1	模式 0	CPOL=0，CPHA=0	**重要**
2	模式 1	CPOL=0，CPHA=1	
3	模式 2	CPOL=1，CPHA=0	
4	模式 3	CPOL=1，CPHA=1	**重要**

SPI 的 4 种模式中，最常用的是模式 0 和模式 3。正是由于 SPI 有 4 种工作模式，因此当我们使用 SPI 总线时，需要去查询 SPI 总线中主机设备（如单片机）和从机设备（如 SPI 存储器）的数据手册，确定他们支持什么模式，从而选择适合的工作模式。

SPI 的 4 种模式的时序图如下。

■ **时钟相位 CPHA=0 时的时序**

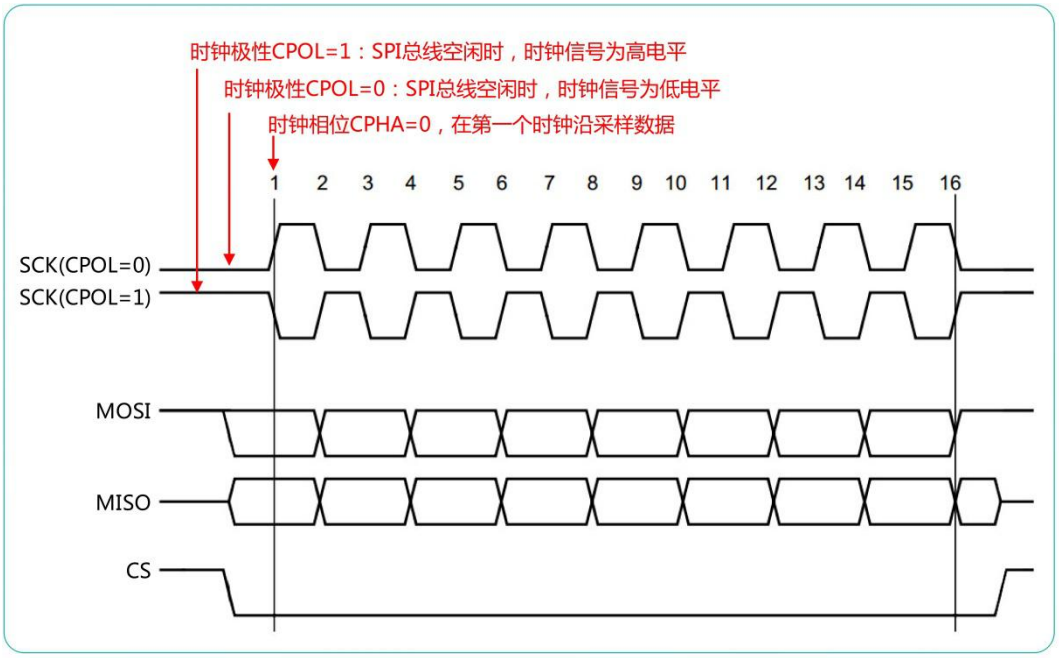

图 5：CPHA=0 时 SPI 时序

■ **时钟相位 CPHA=1 时的时序**

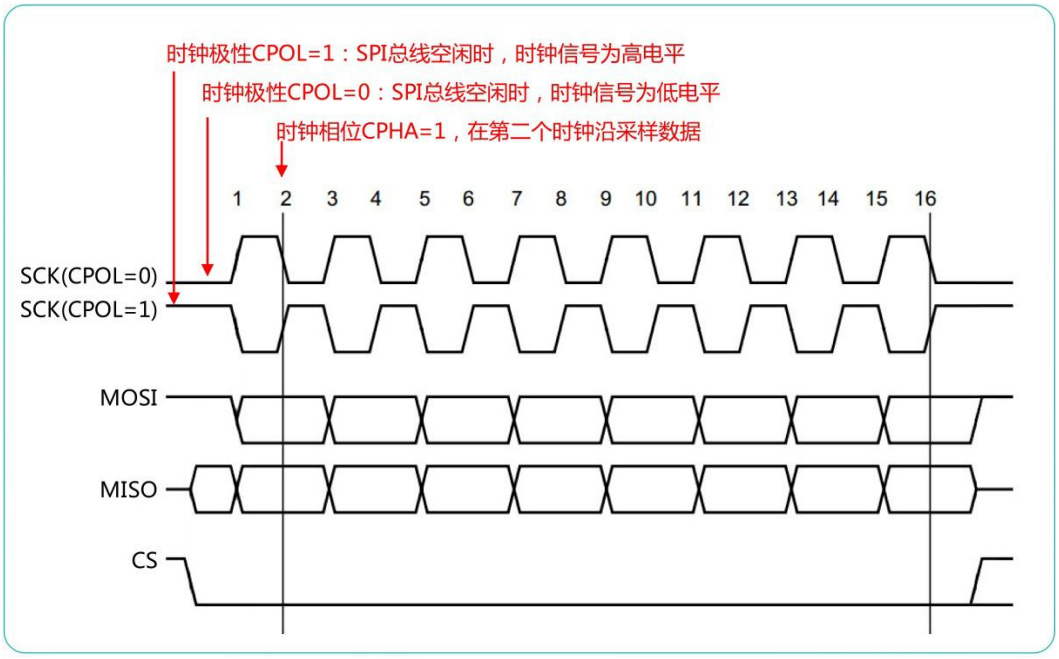

图 6：CPHA=0 时 SPI 时序

3.3. STC15W4K32S4 系列单片机 SPI 介绍

STC15W4K32S4 系列单片机片内有 1 个高速串行通信接口 SPI 接口（以下简称 SPI），该接口支持两种操作模式：主模式和从模式。STC15W4K32S4 系列单片机 SPI 工作在主模式中可支持高达 3Mbps 的速率（工作频率为 12MHz 时，如果 CPU 主频采用 20MHz 到 36MHz，则可更高，从模式时速度无法太快，以 SYSclk/4 以内为宜），还具有传输完成标志和写冲突

标志保护。

STC15W4K32S4 系列单片机每一组 SPI 都有 4 个 IO 引脚供选择使用，如下表所示。

表 2：单片机 SPI 外设引脚分配

分组	SPIx	对应 IO 口	功能描述	说明	备注
第一组	SS	P1.2	使能信号引脚	非独立 GPIO	nRF24L01+接口
	MOSI	P1.3	主出从入引脚	非独立 GPIO	nRF24L01+接口
	MISO	P1.4	主入从出引脚	非独立 GPIO	nRF24L01+接口
	SCLK	P1.5	时钟信号引脚	非独立 GPIO	nRF24L01+接口
第二组	SS_2	P2.4	使能信号引脚 2	非独立 GPIO	LCD 屏接口
	MOSI_2	P2.3	主出从入引脚 2	非独立 GPIO	LCD 屏接口
	MISO_2	P2.2	主入从出引脚 2	非独立 GPIO	LCD 屏接口
	SCLK_2	P2.1	时钟信号引脚 2	非独立 GPIO	LCD 屏接口
第三组	SS_3	P5.4	使能信号引脚 3	非独立 GPIO	W5500 接口
	MOSI_3	P4.0	主出从入引脚 3	非独立 GPIO	W5500 接口
	MISO_3	P4.1	主入从出引脚 3	非独立 GPIO	W5500 接口
	SCLK_3	P4.3	时钟信号引脚 3	非独立 GPIO	W5500 接口

✧ 注：独立 GPIO 表示开发板没有其他的电路使用这个 GPIO，非独立 GPIO 说明开发板有其他电路用到了该 GPIO。须知同一时刻只能使能一组 IO 口作为 SPI 使用。

STC15W4K32S4 系列单片机 SPI 使用哪一组 IO 口由 P_SW1 外围设备切换控制寄存器的 B2、B3 位决定，如下图所示。

图 7：外围设备切换控制寄存器 P_SW1

✧ 注：一般 P_SW1 寄存器 B2、B3 位默认是 0，即如果没有对 P_SW1 寄存器进行操作，则默认选择的 SPI 是 P1.2、P1.3、P1.4、P1.5 这一组。

4. 软件设计

4.1. SPI 相关寄存器汇集

STC15W4K32S4 系列单片机使用 SPI 外设时会用到 6 个寄存器，如下表所示：

表 3：STC15W4K32S4 系列 SPI 使用寄存器汇总

序号	寄存器名	读/写	功能描述
1	P_SW1/AUXR1	读/写	外围设备切换控制寄存器。
2	SPCTL	读/写	SPI 控制寄存器。
3	SPSTAT	读/写	SPI 状态寄存器。
4	SPDAT	读/写	SPI 数据寄存器。
5	IE2	读/写	中断允许寄存器 2。
6	IP2	读/写	中断优先级控制寄存器 2。

4.2. 寄存器解析

4.2.1. SPI 控制寄存器 SPCTL

SPI 控制寄存器控制 SPI 工作模式、时钟频率、使能选择等，详见下图。

图 8：SPI 控制寄存器 SPCTL

4.2.2. SPI 状态寄存器 SPSTAT

SPI 状态寄存器 SPSTAT 的 B7 位用来标志 SPI 传输完成，B6 位则在 SPI 对数据寄存器 SODAT 写操作时置 1，详见下图。

SFR name	Address	bit	B7	B6	B5	B4	B3	B2	B1	B0
SPSTAT	CDH	name	SPIF	WCOL	-	-	-	-	-	-

1 寄存器名

SPIF：SPI传输完成标志。
当一次串行传输完成时，SPIF置位。此时，如果SPI中断被打开(即ESPI (IE2.1) 和EA(IE.7) 都置位)，则产生中断。当SPI处于主模式且SSIG=0时，如果$\overline{SS}$为输入并被驱动为低电平，SPIF也将置位，表示“模式改变”。SPIF标志通过软件向其写入"1"清零。

2 SPI结束标志位

WCOL：SPI写冲突标志。
在数据传输的过程中如果对SPI 数据寄存器SPDAT执行写操作，WCOL将置位。WCOL标志通过软件向其写入"1"清零。

3 SPI写冲突标志位

图 9：SPI 状态寄存器 SPSTAT

4.2.3. 中断允许寄存器 IE2

中断允许寄存器 IE2 中 B1 位用于对 SPI 中断的使能控制，参考下图。

IE2：中断允许寄存器2

SFR name	Address	bit	B7	B6	B5	B4	B3	B2	B1	B0
IE2	AFH	name	-	ET4	ET3	ES4	ES3	ET2	ESPI	ES2

1 寄存器名

ESPI：SPI中断允许位
ESPI=1，允许SPI中断，
ESPI=0，禁止SPI中断。

2 SPI中断使能位

图 10：ISP/IAP 命令触发寄存器

✧ 注：中断允许寄存器 IE2 还有对其他外设的中断使能控制位，所以操作该寄存器时要按位操作。

4.2.4. 中断优先级控制寄存器 IP2

中断优先级控制寄存器 IP2 中 B1 位用于对 SPI 中断的优先级进行配置，详见下图。。

IP2：中断优先级控制寄存器2

SFR name	Address	bit	B7	B6	B5	B4	B3	B2	B1	B0
IP2	B5H	name	-	-	-	-	-	-	PSPI	PS2

1 寄存器名

PSPI：SPI中断优先级控制位。
当PSPI=0时，SPI中断为最低优先级中断(优先级0)
当PSPI=1时，SPI中断为最高优先级中断(优先级1)

2 SPI中断优先级控制位

图 11：ISP/IAP 控制寄存器

✧ 注：中断优先级控制寄存器 IP2 还有对其他外设的中断优先级控制位，所以操作该寄存器时要按位操作。

4.3. SPI 配置步骤

针对 STC15W4K32S4 系列单片机 SPI，软件的配置过程如下：

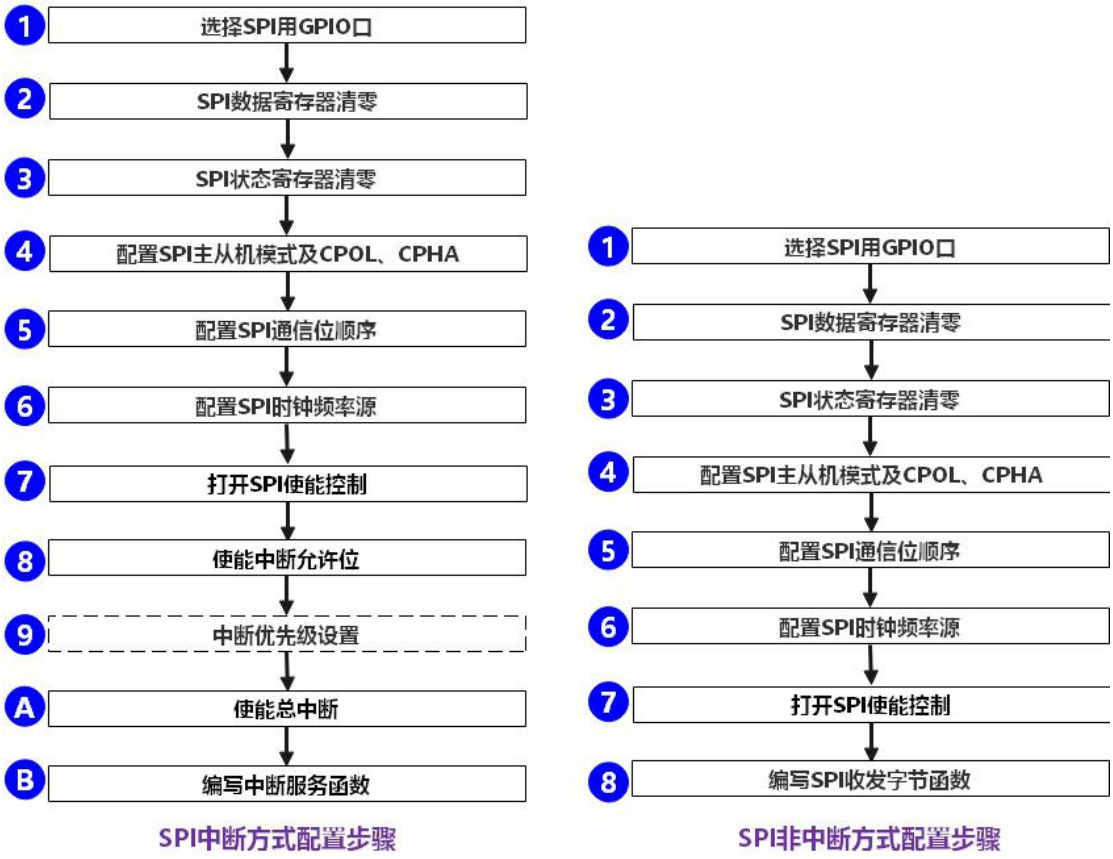

图 12：SPI 中断方式和非中断方式软件配置步骤

4.4.外接 FRAM 存储器读写单字节实验（模拟 SPI）

✧ 注：本节的实验源码是在“实验 2-8-1：串口 1 收发实验（P3.0 和 P3.1）”的基础上修改。本节对应的实验源码是:“实验 2-14-1:外接 FRAM 存储器读写单字节实验(模拟 SPI)”。

4.4.1. 工程需要用到的 c 文件

本例需要用到的 c 文件如下表所示，工程需要添加下表中的 c 文件。

表 4：实验需要用到的 c 文件

序号	文件名	后缀	功能描述
1	uart	.c	包含与用户 uart 有关的用户自定义函数。
2	fm25cl64b	.c	SPI 通信及操作 FRAM 有关的用户自定义函数。
3	delay	.c	包含用户自定义延时函数。

4.4.2. 头文件引用和路径设置

■ 需要引用的头文件

```
1. #include "delay.h"
2. #include "uart.h"
3. #include " fm25cl64b.h"
```

■ 需要包含的头文件路径

本例需要包含的头文件路径如下表：

表 5：头文件包含路径

序号	路径	描述
1	..\ Source	uart.h、fm25cl64b.h 和 delay.h 头文件在该路径，所以要包含。
2	..\User	15W4KxxS4.h 头文件在该路径，所以要包含。

MDK 中点击魔术棒，打开工程配置窗口，按照下图所示添加头文件包含路径。

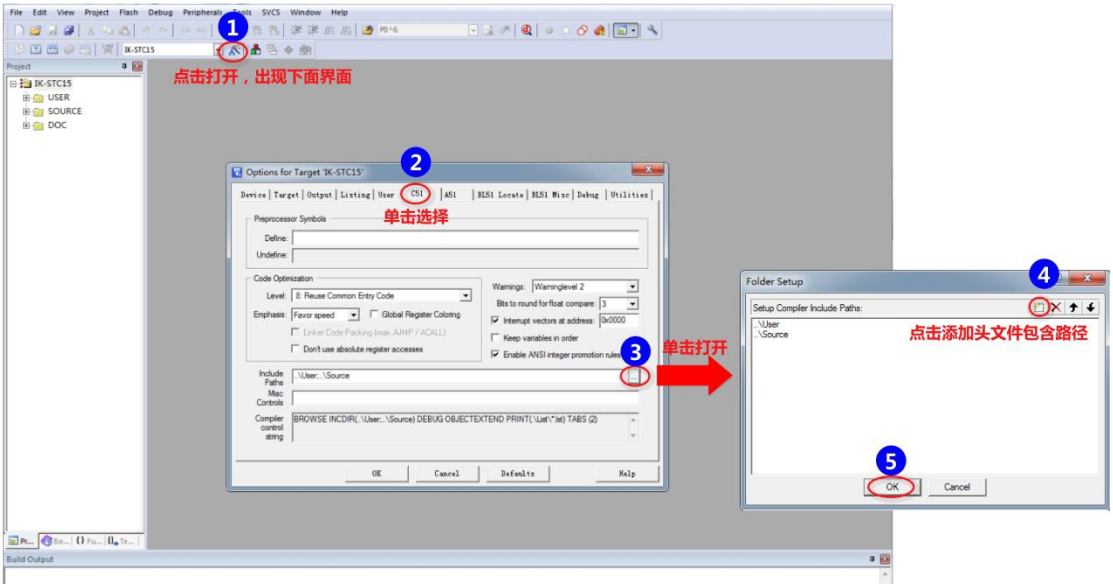

图 13：添加头文件包含路径

4.4.3. 编写代码

首先，在 fm25cl64b.c 文件中编写模拟 SPI 方式的读写字节函数，这里的模拟 SPI 读字节函数与写字节函数是分开的，代码如下：

程序清单：模拟 SPI 写字节函数

```
1.  /*********************************************************************************
2.   * 描   述 : 模拟 SPI 写入一个字节
3.   * 入   参 : uint8 date
4.   * 返回值 : 无
5.   *********************************************************************************/
6.  void SPI_WriteByte(uint8 date)
7.  {
8.    uint8 temp,i;
9.    temp = date;
10.
11.   for (i = 0; i < 8; i++)
12.     {
```

```
13.      SPI_SCK_0;
14.          delay_us(10);
15.      if((temp&0x80)==0x80)
16.      { SPI_MOSI_1; }
17.      else
18.      { SPI_MOSI_0; }
19.      SPI_SCK_1 ;
20.          delay_us(10);
21.      temp <<= 1;
22.    }
23.    SPI_MOSI_0;
24. }
```

程序清单：模拟 SPI 读字节函数

```
1.  /*********************************************************************
2.   * 描   述 : 模拟 SPI 读取一个字节
3.   * 入   参 : 无
4.   * 返回值 : 读取 uint8 数据
5.   *********************************************************************/
6.  uint8 SPI_ReadByte(void)
7.  {
8.    uint8 temp=0;
9.    uint8 i;
10.
11.   for(i = 0; i < 8; i++)
12.     {
13.     temp <<= 1;
14.     SPI_SCK_0 ;
15.     delay_us(10);
16.         if(E_SPI_MISO)
17.     {temp++; }
18.     SPI_SCK_1 ;
19.         delay_us(10);
20.    }
21.    return(temp);
22. }
```

然后，编写对 FRAM 存储器的基本操作函数，如下表所示。

表 6：FRAM 相关用户函数汇集

序号	函数名	功能描述

1	FramWriteByte	向 FRAM 指定地址存入单字节数据。
2	FramReadByte	从 FRAM 指定地址读取单字节数据。
3	FRAM_Read_Nbyte	从 FRAM 指定地址读取多字节数据。
4	FRAM_Write_Nbyte	向 FRAM 指定地址存入多字节数据。
5	FRAM_Erase_Nbyte	在 FRAM 指定地址开始擦除指定长度的数据。

关于每个操作外部 FRAM 相关用户函数，下面给出详细代码。

程序清单：向指定地址存入单字节数据函数

```
/*****************************************************************************
 * 描  述 : 向指定地址存入数据
 * 入  参 : uint16 address 写入的地址,uint8  da 写入的数据
 * 返回值 : 无
 *****************************************************************************/
void FramWriteByte(uint16 address,uint8  da)
{
  uint8 temM,temL;

  temM=(uint8)((address&0xff00)>>8);
  temL=(uint8)(address&0x00ff);

  SPI_CS_0;                          //使能器件
  SPI_WriteByte(FM25CL64_WREN);      //写使能 FRAM
  SPI_CS_1;                          //取消片选
  delay_80us();

  SPI_CS_0;                          //使能器件
  SPI_WriteByte(FM25CL64_WRITE);     //写数据到 FRAM 存储器命令
  SPI_WriteByte(temM);               //写存储器地址高 8 位
  SPI_WriteByte(temL);               //写存储器地址低 8 位
  SPI_WriteByte(da);                 //写入要存入的数据
  SPI_CS_1;                          //取消片选
}
```

程序清单：从指定地址读取单字节数据函数

```
/*****************************************************************************
 * 描  述 : 向指定地址读取数据
 * 入  参 : uint16 address 要读取的地址
 * 返回值 : 返回读取到的数据 uint8
 *****************************************************************************/
uint8 FramReadByte(uint16 address)
{
```

```
    uint8  temp,temM,temL;

    temM=(uint8)((address&0xff00)>>8);
    temL=(uint8)(address&0x00ff);

    SPI_CS_0;                          //使能器件
    SPI_WriteByte(FM25CL64_READ);      //读存储器命令下发
    SPI_WriteByte(temM);               //待读地址高位下发
    SPI_WriteByte(temL);               //待读地址低位下发
    temp = SPI_ReadByte();             //读存储器数据
    SPI_CS_1;                          //取消片选

    return temp;
}
```

程序清单：从指定地址读取多字节数据函数

```
/*****************************************************************************
功能描述：在指定地址开始读取多字节数据
入口参数：pbuf:数据存储区  ReadAddr:读数据首地址  Len:要读取的字节数
返回值：无
*****************************************************************************/
void FRAM_Read_Nbyte(uint8 * pbuf,uint16 ReadAddr,uint16 Len)
{
    uint16 i;

    SPI_CS_0;                                  //使能器件
    SPI_WriteByte(FM25CL64_READ);              //发送读数据命令
    SPI_WriteByte((uint8)((ReadAddr&0xE0)>>8)); //写入高 8 位址，最大地址 13 位
    SPI_WriteByte((uint8)ReadAddr);            //写入低 8 位址
    for(i=0;i<Len;i++)                         //连续读取数据
    {
       *pbuf++=SPI_ReadByte();                 //读一个字节
    }
    SPI_CS_1;                                  //取消片选
}
```

程序清单：向指定地址存入多字节数据函数

```
/*****************************************************************************
功能描述：在指定地址开始写入多字节数据
入口参数：pbuf:数据存储区  WriteAddr:写入数据首地址  Len:要写入的字节数
返回值：无
*****************************************************************************/
void FRAM_Write_Nbyte(uint8 * pbuf,uint16 WriteAddr,uint16 Len)
```

```
{
    uint16 i;

    SPI_CS_0;                                   //使能器件
    SPI_WriteByte(FM25CL64_WREN);               //写使能 FRAM
    SPI_CS_1;                                   //取消片选
      delay_80us();

    SPI_CS_0;                                   //使能器件
      SPI_WriteByte(FM25CL64_WRITE);              //发送写数据命令
    SPI_WriteByte((uint8)((WriteAddr&0xE0)>>8));  //写入高 8 位址,最大地址 13 位
    SPI_WriteByte((uint8)WriteAddr);            //写入低 8 位址

    for(i=0;i<Len;i++)                          //连续写入数据
    {
            SPI_WriteByte(pbuf[i]);             //写入要存入的数据
    }
        SPI_CS_1;                               //取消片选
}
```

程序清单：擦除指定长度的数据函数

```
/*****************************************************************************
功能描述：在指定地址开始擦除指定长度的数据
入口参数：EraseAddr:擦除首地址   Len:要擦除的字节数
返回值：无
*****************************************************************************/
void FRAM_Erase_Nbyte(uint16 EraseAddr,uint16 Len)
{
    uint16 i;

    SPI_CS_0;                                   //使能器件
    SPI_WriteByte(FM25CL64_WREN);               //写使能 FRAM
    SPI_CS_1;                                   //取消片选
    delay_80us();

    SPI_CS_0;                                   //使能器件
    SPI_WriteByte(FM25CL64_WRITE);              //发送写数据命令
    SPI_WriteByte((uint8)((EraseAddr&0xE0)>>8));  //写入高 8 位址,最大地址 13 位
    SPI_WriteByte((uint8)EraseAddr);            //写入低 8 位址

    for(i=0;i<Len;i++)                          //连续擦除数据
    {
```

```
22.             SPI_WriteByte(0x00);                    //擦除数据即写入 0x00
23.     }
24.         SPI_CS_1;                                   //取消片选
25. }
```

最后，在主函数中对串口 1 进行初始化，并通过串口 1 发送不同的命令实现对单片机片外 FRAM 的单字节读、写及擦除等操作。

代码清单：主函数

```
1.  int main()
2.  {
3.      uint8   Temp;
4.
5.  ////////////////////////////////////////////////
6.  //注意: STC15W4K32S4 系列的芯片,上电后所有与 PWM 相关的 IO 口均为
7.  //       高阻态,需将这些口设置为准双向口或强推挽模式方可正常使用
8.  //相关 IO: P0.6/P0.7/P1.6/P1.7/P2.1/P2.2
9.  //         P2.3/P2.7/P3.7/P4.2/P4.4/P4.5
10. ////////////////////////////////////////////////
11.     P2M1 &= 0xE1;   P2M0 &= 0xE1;                   //设置 P2.1~P2.4 为准双向口
12.     P3M1 &= 0xFC;   P3M0 &= 0xFC;                   //设置 P3.0~P3.1 为准双向口
13.
14.     SPI_CS_1;                                       //SPI 使能引脚初始化
15.     Uart1_Init();                                   //串口 1 初始化
16.     EA = 1;                                         //使能总中断
17.     delay_ms(10);                                   //初始化后延时
18.
19.   while (1)
20.     {
21.      if(WriteFLAG)                          //写模式
22.         {
23.             WriteFLAG=0;                        //写标志变量清零,发送一次
24.             FramWriteByte(0x0050,0x33);  //向 FRAM 地址 0x0050 中写入单字节数据 0x33
25.         SendDataByUart1(0x33);              //串口 1 发送数据 0x33 表示写操作完成
26.          }
27.          if(ReadFLAG)                           //读模式
28.          {
29.             ReadFLAG=0;                         //读标志变量清零,发送一次
30.             Temp=FramReadByte(0x0050);          //从 FRAM 地址 0x0050 读取单字节数据
    并赋值给变量 Temp
31.         SendDataByUart1(Temp);          //串口 1 发送 Temp 变量值（即读取的单字节数据）
32.          }
33.          if(ClearFLAG)                          //擦除模式
34.          {
```

```
35.                 ClearFLAG=0;                    //清除标志变量清零,发送一次
36.                 FRAM_Erase_Nbyte(0x0050,1);     //擦除 FRAM 地址 0x0050 的数据
37.                 SendDataByUart1(0x00);          //串口 1 发送数据 0x00 表示擦除完成
38.             }
39.         }
40. }
```

4.4.4. 硬件连接

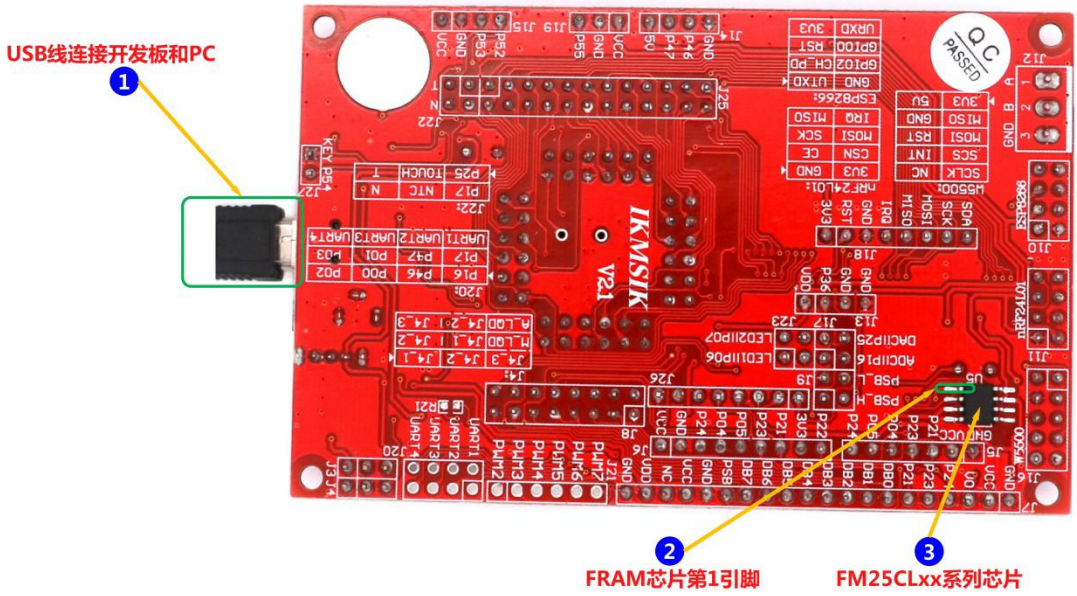

图 14：开发板连接图

4.4.5. 实验步骤

1. 解压“…\第 3 部分：配套例程源码\ 1 - 基础实验程序\”目录下的压缩文件“实验 2-14-1：外接 FRAM 存储器读写单字节实验（模拟 SPI）”，将解压后得到的文件夹拷贝到合适的目录，如“D\STC15”。
2. 启动 Keil C51。
3. 在 Keil C51 中执行“Project→Open Project”打开“…\FM25CLxx\project”目录下的工程“FM25CLxx.uvproj”。
4. 点击编译按钮编译工程。注意查看编译输出栏，观察编译的结果，如果有错误，修改程序，直到编译成功为止。编译后生成的 HEX 文件“FM25CLxx.hex”位于工程目录下的“Output”文件夹中。
5. 打开 STC-ISP 软件下载程序 。下载使用内部 IRC 时钟，IRC 频率选择为 11.0592MHZ。
6. 程序运行后，实验现象及步骤操作如下：

   1）在串口调试助手发送端发"3"，会在接收端显示 00。

   2）在串口调试助手发送端发"2"，会在接收端显示 33，在串口调试助手发送端发"3"，会在接收端显示 33。

   3）在串口调试助手发送端发"4"，会在接收端显示 00，在串口调试助手发送端发"3"，

会在接收端显示 00。

4.5. 外接 FRAM 存储器读写多字节实验（模拟 SPI）

✧ 注：本节的实验源码是在“实验 2-14-1：外接 FRAM 存储器读写单字节实验（模拟 SPI）”的基础上修改。本节对应的实验源码是：“实验 2-14-2：外接 FRAM 存储器读写多字节实验（模拟 SPI）”。

4.5.1. 工程需要用到的 c 文件

本实验需要用到的头文件以及添加头文件包含路径的方法请参考“实验 2-14-1：外接 FRAM 存储器读写单字节实验（模拟 SPI）”部分。

4.5.2. 编写代码

首先，在 fm25cl64b.c 文件中编写模拟 SPI 方式的读写字节函数和对 FRAM 存储器的基本操作函数。请参考“实验 2-14-1：外接 FRAM 存储器读写单字节实验（模拟 SPI）”部分。

然后，在主函数中对串口 1 进行初始化，并通过串口 1 发送不同的命令实现对单片机片外 FRAM 的多字节读、写及擦除等操作。

代码清单：主函数

```
int main()
{
//////////////////////////////////////////////
//注意: STC15W4K32S4 系列的芯片,上电后所有与 PWM 相关的 IO 口均为
//       高阻态,需将这些口设置为准双向口或强推挽模式方可正常使用
//相关 IO: P0.6/P0.7/P1.6/P1.7/P2.1/P2.2
//         P2.3/P2.7/P3.7/P4.2/P4.4/P4.5
//////////////////////////////////////////////
    P2M1 &= 0xE1;   P2M0 &= 0xE1;                    //设置 P2.1~P2.4 为准双向口
    P3M1 &= 0xFC;   P3M0 &= 0xFC;                    //设置 P3.0~P3.1 为准双向口

    SPI_CS_1;                                       //SPI 使能引脚初始化
    Uart1_Init();                                   //串口 1 初始化
    EA = 1;                                         //使能总中断
  delay_ms(10);                                    //初始化后延时

  while(1)
  {
     if(WriteFLAG)                                      //写模式
        {
            WriteFLAG=0;                               //写标志变量清零,发送一次
            FRAM_Write_Nbyte(FRAM_Test_Write,0x0000,10);    //向 FRAM 地址 0x0000
中写入 FRAM_Test_Write 数组中 10 个字节数据
        SendDataByUart1(0x33);           //串口 1 发送数据 0x33 表示写操作完成
        }
```

```
25.          if(ReadFLAG)                                    //读模式
26.          {
27.               ReadFLAG=0;                          //读标志变量清零,发送一次
28.               FRAM_Read_Nbyte(FRAM_Test_Read,0x0000,10) ;     //从 FRAM 地址 0x0000
     读取 10 个字节数据存放数组 FRAM_Test_Read 中
29.               SendStringByUart1_n(FRAM_Test_Read,10);         //串口 1 发送数组
     FRAM_Test_Read 中的值（即读取的多字节数据）
30.          }
31.          if(ClearFLAG)                                   //清除模式
32.          {
33.               ClearFLAG=0;                         //清除标志变量清零,发送一次
34.               FRAM_Erase_Nbyte(0x0000,10);                    //擦除 FRAM 地址
     0x0000 开始的 10 个字节长度的数据
35.          SendDataByUart1(0x00);                    //串口 1 发送数据 0x00 表示擦除完成
36.          }
37.     }
38. }
```

4.5.3. 硬件连接

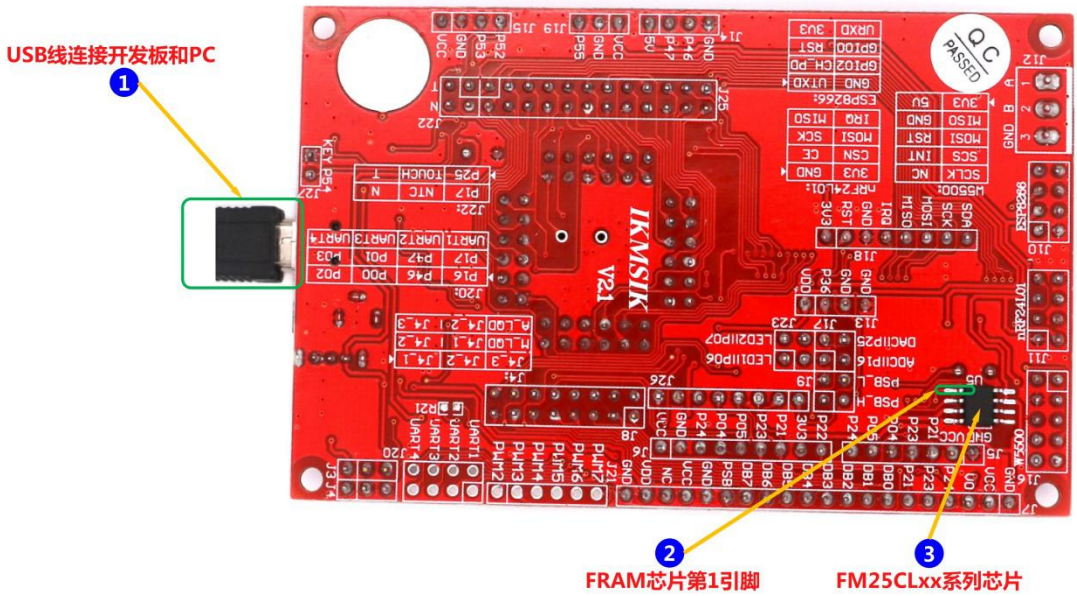

图 15：开发板连接图

4.5.4. 实验步骤

1. 解压“…\第 3 部分：配套例程源码\ 1 - 基础实验程序\”目录下的压缩文件“实验 2-14-2：外接 FRAM 存储器读写多字节实验（模拟 SPI）”，将解压后得到的文件夹拷贝到合适的目录，如“D\STC15”。
2. 启动 Keil C51。
3. 在 Keil C51 中执行“Project→Open Project”打开“…\FM25CLxx\project”目录下的工

程“FM25CLxx.uvproj”。

4. 点击编译按钮编译工程。注意查看编译输出栏，观察编译的结果，如果有错误，修改程序，直到编译成功为止。编译后生成的 HEX 文件“FM25CLxx.hex”位于工程目录下的“Output”文件夹中。
5. 打开 STC-ISP 软件下载程序 。下载使用内部 IRC 时钟，IRC 频率选择为 11.0592MHZ。
6. 程序运行后，实验现象及步骤操作如下：

   1）在串口调试助手发送端发"3"，会在接收端显示 00 00 00 00 00 00 00 00 00 00。

   2）在串口调试助手发送端发"2"，会在接收端显示 33，在串口调试助手发送端发"3"，会在接收端显示 11 22 33 44 55 66 77 88 99 AA。

   3）在串口调试助手发送端发"4"，会在接收端显示 00，在串口调试助手发送端发"3"，会在接收端显示 00 00 00 00 00 00 00 00 00 00。

4.6. 外接 FRAM 存储器读写单字节实验（硬件 SPI）

✧ 注：本节的实验源码是在“实验 2-14-1：外接 FRAM 存储器读写单字节实验（模拟 SPI）”的基础上修改。本节对应的实验源码是：“实验 2-14-3：外接 FRAM 存储器读写单字节实验（硬件 SPI）”。

4.6.1. 工程需要用到的 c 文件

本例需要用到的 c 文件如下表所示，工程需要添加下表中的 c 文件。

表 7：实验需要用到的 c 文件

序号	文件名	后缀	功能描述
1	uart	.c	包含与用户 uart 有关的用户自定义函数。
2	fm25cl64b	.c	SPI 通信及操作 FRAM 有关的用户自定义函数。
3	delay	.c	包含用户自定义延时函数。

4.6.2. 头文件引用和路径设置

■ 需要引用的头文件

```
1. #include "delay.h"
2. #include "uart.h"
3. #include " fm25cl64b.h"
```

■ 需要包含的头文件路径

本例需要包含的头文件路径如下表：

表 8：头文件包含路径

序号	路径	描述

1	..\ Source	uart.h、fm25cl64b.h 和 delay.h 头文件在该路径，所以要包含。
2	..\User	15W4KxxS4.h 头文件在该路径，所以要包含。

MDK 中点击魔术棒，打开工程配置窗口，按照下图所示添加头文件包含路径。

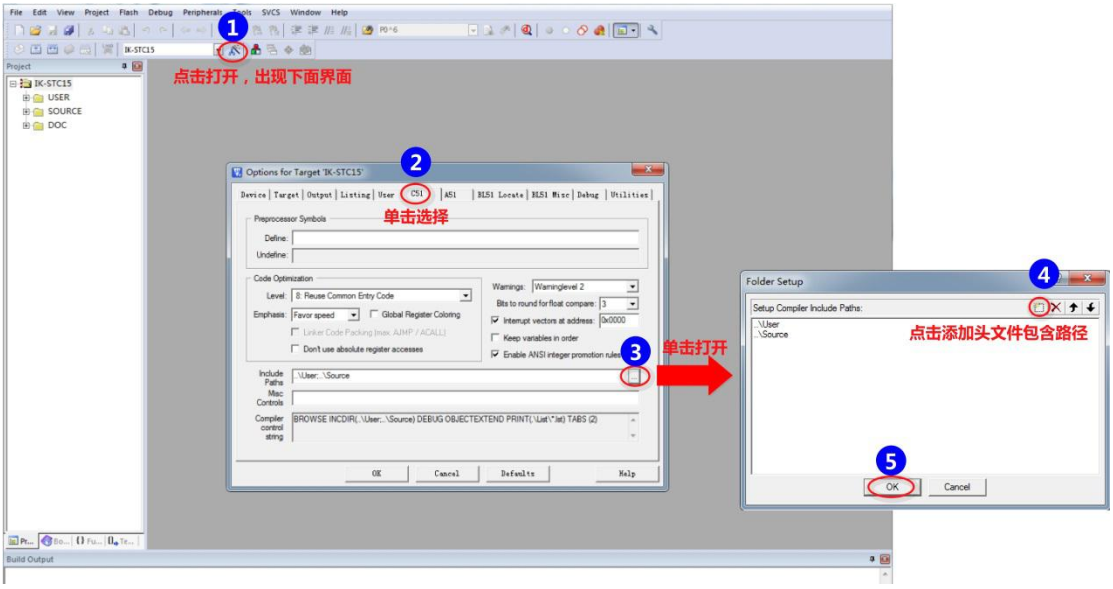

图 16：添加头文件包含路径

4.6.3. 编写代码

首先，在 fm25cl64b.c 文件中对硬件 SPI 进行初始化，并编写硬件 SPI 的读写字节函数，代码如下：

程序清单：硬件 SPI 初始化函数

```
/*********************************************************************************
 * 描  述 : 硬件 SPI 初始化
 * 入  参 : 无
 * 返回值 : 无
 *********************************************************************************/
void Init_SPI(void)
{
    AUXR1|=0X04;                                //将 SPI 调整到 P2.1 P2.2 P2.3
    AUXR1&=0XF7;
    SPDAT = 0;
    SPSTAT = SPIF | WCOL;                       //清除 SPI 状态位
    SPCTL = SPEN | MSTR | SSIG;                 //主机模式
}
```

程序清单：硬件 SPI 读写字节函数

```
/*********************************************************************************
```

```
2.   * 描   述 : 硬件 SPI 写入一个字节，并返回一个值
3.   * 入   参 : uint8 date
4.   * 返回值 : 无
5.   ******************************************************************************/
6.  uint8 SPI_SendByte(uint8 SPI_SendData)
7.  {
8.    SPDAT = SPI_SendData;          //触发 SPI 发送数据
9.    while (!(SPSTAT & SPIF));      //等待发送完成
10.   SPSTAT = SPIF | WCOL;          //清除 SPI 状态位
11.   return SPDAT;                  //返回 SPI 数据
12. }
```

然后，编写对 FRAM 存储器的基本操作函数，如下表所示。

表 9：FRAM 相关用户函数汇集

序号	函数名	功能描述
1	FramWriteByte	向 FRAM 指定地址存入单字节数据。
2	FramReadByte	从 FRAM 指定地址读取单字节数据。
3	FRAM_Read_Nbyte	从 FRAM 指定地址读取多字节数据。
4	FRAM_Write_Nbyte	向 FRAM 指定地址存入多字节数据。
5	FRAM_Erase_Nbyte	在 FRAM 指定地址开始擦除指定长度的数据。

关于每个操作外部 FRAM 相关用户函数，下面给出详细代码。

程序清单：向指定地址存入单字节数据函数

```
1.  /*********************************************************************************
2.   * 描   述 : 向指定地址存入数据
3.   * 入   参 : uint16 address 写入的地址,uint8  da 写入的数据
4.   * 返回值 : 无
5.   *********************************************************************************/
6.  void FramWriteByte(uint16 address,uint8  da)
7.  {
8.    uint8 temM,temL;
9.
10.   temM=(uint8)((address&0xff00)>>8);
11.   temL=(uint8)(address&0x00ff);
12.
13.   SPI_CS_0;                      //使能器件
14.   SPI_SendByte(WREN);            //写使能 FRAM
15.   SPI_CS_1;                      //取消片选
16.   delay_80us();
17.
```

```
18.   SPI_CS_0;                    //使能器件
19.   SPI_SendByte(WRITEE);        //写数据到 FRAM 存储器命令
20.   SPI_SendByte(temM);          //写存储器地址高 8 位
21.   SPI_SendByte(temL);          //写存储器地址低 8 位
22.   SPI_SendByte(da);            //写入要存入的数据
23.   SPI_CS_1;                    //取消片选
24. }
```

程序清单：从指定地址读取单字节数据函数

```
1.  /*****************************************************************************
2.   * 描   述 : 向指定地址读取数据
3.   * 入   参 : uint16 address 要读取的地址
4.   * 返回值 : 返回读取到的数据 uint8
5.   *****************************************************************************/
6.  uint8 FramReadByte(uint16 address)
7.  {
8.    uint8  temp,temM,temL;
9.
10.   temM=(uint8)((address&0xff00)>>8);
11.   temL=(uint8)(address&0x00ff);
12.
13.   SPI_CS_0;                     //使能器件
14.   SPI_SendByte(READD);          //读存储器命令下发
15.   SPI_SendByte(temM);           //待读地址高位下发
16.   SPI_SendByte(temL);           //待读地址低位下发
17.   temp = SPI_SendByte(0xFF);    //读存储器数据
18.   SPI_CS_1;                     //取消片选
19.
20.   return temp;
21. }
```

程序清单：从指定地址读取多字节数据函数

```
1.  /*****************************************************************************
2.  功能描述：在指定地址开始读取多字节数据
3.  入口参数：pbuf:数据存储区   ReadAddr:读数据首地址   Len:要读取的字节数
4.  返回值：无
5.  *****************************************************************************/
6.  void FRAM_Read_Nbyte(uint8 * pbuf,uint16 ReadAddr,uint16 Len)
7.  {
8.      uint16 i;
9.
10.     SPI_CS_0;                                   //使能器件
11.     SPI_SendByte(FM25CL64_READ);                //发送读数据命令
```

```
    SPI_SendByte((uint8)((ReadAddr&0xE0)>>8));      //写入高 8 位址，最大地址 13 位
    SPI_SendByte((uint8)ReadAddr);                  //写入低 8 位址
    for(i=0;i<Len;i++)                              //连续读取数据
    {
       *pbuf++=SPI_SendByte(0xFF);                  //读一个字节
    }
    SPI_CS_1;                                       //取消片选
}
```

程序清单：向指定地址存入多字节数据函数

```
/*****************************************************************************
功能描述：在指定地址开始写入多字节数据
入口参数：pbuf:数据存储区  WriteAddr:写入数据首地址  Len:要写入的字节数
返回值：无
*****************************************************************************/
void FRAM_Write_Nbyte(uint8 * pbuf,uint16 WriteAddr,uint16 Len)
{
    uint16 i;

    SPI_CS_0;                                       //使能器件
    SPI_SendByte(FM25CL64_WREN);                    //写使能 FRAM
    SPI_CS_1;                                       //取消片选
      delay_80us();

    SPI_CS_0;                                       //使能器件
    SPI_SendByte(FM25CL64_WRITE);                   //发送写数据命令
    SPI_SendByte((uint8)((WriteAddr&0xE0)>>8));     //写入高 8 位址，最大地址 13 位
    SPI_SendByte((uint8)WriteAddr);                 //写入低 8 位址

    for(i=0;i<Len;i++)                              //连续写入数据
    {
            SPI_SendByte(pbuf[i]);                  //写入要存入的数据
    }
        SPI_CS_1;                                       //取消片选
}
```

程序清单：擦除指定长度的数据函数

```
/*****************************************************************************
功能描述：在指定地址开始擦除指定长度的数据
入口参数：EraseAddr:擦除首地址  Len:要擦除的字节数
返回值：无
*****************************************************************************/
```

```
6.  void FRAM_Erase_Nbyte(uint16 EraseAddr,uint16 Len)
7.  {
8.      uint16 i;
9.
10.     SPI_CS_0;                                          //使能器件
11.     SPI_SendByte(FM25CL64_WREN);                       //写使能 FRAM
12.     SPI_CS_1;                                          //取消片选
13.       delay_80us();
14.
15.     SPI_CS_0;                                          //使能器件
16.       SPI_SendByte(FM25CL64_WRITE);                      //发送写数据命令
17.     SPI_SendByte((uint8)((EraseAddr&0xE0)>>8));        //写入高 8 位址，最大地址 13 位
18.     SPI_SendByte((uint8)EraseAddr);                    //写入低 8 位址
19.
20.     for(i=0;i<Len;i++)                                 //连续擦除数据
21.     {
22.             SPI_SendByte(0x00);                        //擦除数据即写入 0x00
23.     }
24.         SPI_CS_1;                                      //取消片选
25. }
```

最后，在主函数中对串口 1 和 SPI 进行初始化，并通过串口 1 发送不同的命令实现对单片机片外 FRAM 的单字节读、写及擦除等操作。

代码清单：主函数

```
1.  int main()
2.  {
3.      uint8   Temp;
4.
5.  ////////////////////////////////////////////////
6.  //注意: STC15W4K32S4 系列的芯片,上电后所有与 PWM 相关的 IO 口均为
7.  //       高阻态,需将这些口设置为准双向口或强推挽模式方可正常使用
8.  //相关 IO: P0.6/P0.7/P1.6/P1.7/P2.1/P2.2
9.  //         P2.3/P2.7/P3.7/P4.2/P4.4/P4.5
10. ////////////////////////////////////////////////
11.     P2M1 &= 0xE1;   P2M0 &= 0xE1;                      //设置 P2.1~P2.4 为准双向口
12.     P3M1 &= 0xFC;   P3M0 &= 0xFC;                      //设置 P3.0~P3.1 为准双向口
13.
14.     SPI_CS_1;                                          //SPI 使能引脚初始化
15.     Init_SPI();                                        //初始化 SPI
16.     Uart1_Init();                                      //串口 1 初始化
17.     EA = 1;                                            //使能总中断
18.   delay_ms(10);                                        //初始化后延时
19.
```

```
20.   while (1)
21.     {
22.      if(WriteFLAG)                          //写模式
23.          {
24.               WriteFLAG=0;                       //写标志变量清零,发送一次
25.               FramWriteByte(0x0050,0x33);     //向 FRAM 地址 0x0050 中写入单字节数据
    0x33
26.          SendDataByUart1(0x33);              //串口 1 发送数据 0x33 表示写操作完成
27.          }
28.          if(ReadFLAG)                           //读模式
29.          {
30.               ReadFLAG=0;                        //读标志变量清零,发送一次
31.               Temp=FramReadByte(0x0050);         //从 FRAM 地址 0x0050 读取单字节数据
    并赋值给变量 Temp
32.          SendDataByUart1(Temp);          //串口 1 发送 Temp 变量值（即读取的单字节数据）
33.          }
34.          if(ClearFLAG)                          //擦除模式
35.          {
36.               ClearFLAG=0;                       //清除标志变量清零,发送一次
37.               FRAM_Erase_Nbyte(0x0050,1);        //擦除 FRAM 地址 0x0050 的数据
38.          SendDataByUart1(0x00);              //串口 1 发送数据 0x00 表示擦除完成
39.          }
40.     }
41. }
```

4.6.4. 硬件连接

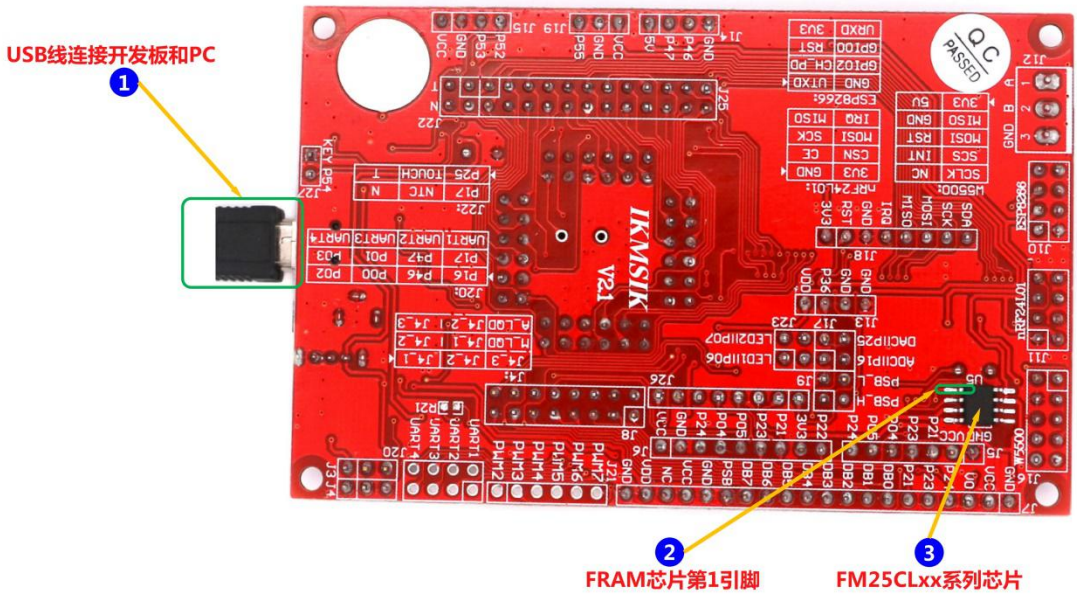

图 17：开发板连接图

4.6.5. 实验步骤

1. 解压“…\第 3 部分：配套例程源码\ 1 - 基础实验程序\”目录下的压缩文件“实验 2-14-3：外接 FRAM 存储器读写单字节实验（硬件 SPI）”，将解压后得到的文件夹拷贝到合适的目录，如“D\STC15”。
2. 启动 Keil C51。
3. 在 Keil C51 中执行“Project→Open Project”打开“…\FM25CLxx\project”目录下的工程“FM25CLxx.uvproj”。
4. 点击编译按钮编译工程。注意查看编译输出栏，观察编译的结果，如果有错误，修改程序，直到编译成功为止。编译后生成的 HEX 文件“FM25CLxx.hex”位于工程目录下的“Output”文件夹中。
5. 打开 STC-ISP 软件下载程序 。下载使用内部 IRC 时钟，IRC 频率选择为 11.0592MHZ。
6. 程序运行后，实验现象及步骤操作如下：

   1）在串口调试助手发送端发"3"，会在接收端显示 00。

   2）在串口调试助手发送端发"2"，会在接收端显示 33，在串口调试助手发送端发"3"，会在接收端显示 33。

   3）在串口调试助手发送端发"4"，会在接收端显示 00，在串口调试助手发送端发"3"，会在接收端显示 00。

4.7. 外接 FRAM 存储器读写多字节实验（硬件 SPI）

✧ 注：本节的实验源码是在“实验 2-14-3：外接 FRAM 存储器读写单字节实验（硬件 SPI）”的基础上修改。本节对应的实验源码是：“实验 2-14-4：外接 FRAM 存储器读写多字节实验（硬件 SPI）”。

4.7.1. 工程需要用到的 c 文件

本实验需要用到的头文件以及添加头文件包含路径的方法请参考“实验 2-14-3：外接 FRAM 存储器读写单字节实验（硬件 SPI）”部分。

4.7.2. 编写代码

首先，在 fm25cl64b.c 文件中编写硬件 SPI 方式的读写字节函数和对 FRAM 存储器的基本操作函数。请参考“实验 2-14-3：外接 FRAM 存储器读写单字节实验（硬件 SPI）”部分。

然后，在主函数中对串口 1 和 SPI 进行初始化，并通过串口 1 发送不同的命令实现对单片机片外 FRAM 的多字节读、写及擦除等操作。

代码清单：主函数

```
1.  int main()
2.  {
3.  ////////////////////////////////////////////////
4.  //注意: STC15W4K32S4 系列的芯片,上电后所有与 PWM 相关的 IO 口均为
5.  //       高阻态,需将这些口设置为准双向口或强推挽模式方可正常使用
```

```
//相关IO: P0.6/P0.7/P1.6/P1.7/P2.1/P2.2
//        P2.3/P2.7/P3.7/P4.2/P4.4/P4.5
////////////////////////////////////////////////
    P2M1 &= 0xE1;   P2M0 &= 0xE1;                    //设置P2.1~P2.4为准双向口
    P3M1 &= 0xFC;   P3M0 &= 0xFC;                    //设置P3.0~P3.1为准双向口

    SPI_CS_1;                                       //SPI使能引脚初始化
    Init_SPI();                                     //初始化SPI
    Uart1_Init();                                   //串口1初始化
    EA = 1;                                         //使能总中断
  delay_ms(10);                                     //初始化后延时

  while(1)
  {
     if(WriteFLAG)                                  //写模式
        {
            WriteFLAG=0;                            //写标志变量清零,发送一次
            FRAM_Write_Nbyte(FRAM_Test_Write,0x0000,10);   //向FRAM地址0x0000中写入FRAM_Test_Write数组中10个字节数据
        SendDataByUart1(0x33);                      //串口1发送数据0x33表示写操作完成
         }
         if(ReadFLAG)                               //读模式
         {
            ReadFLAG=0;                             //读标志变量清零,发送一次
            FRAM_Read_Nbyte(FRAM_Test_Read,0x0000,10) ;    //从FRAM地址0x0000读取10个字节数据存放数组FRAM_Test_Read中
            SendStringByUart1_n(FRAM_Test_Read,10);        //串口1发送数组FRAM_Test_Read中的值（即读取的多字节数据）
         }
         if(ClearFLAG)                              //清除模式
         {
            ClearFLAG=0;                            //清除标志变量清零,发送一次
            FRAM_Erase_Nbyte(0x0000,10);                   //擦除FRAM地址0x0000开始的10个字节长度的数据
        SendDataByUart1(0x00);                      //串口1发送数据0x00表示擦除完成
         }
    }
}
```

4.7.3. 硬件连接

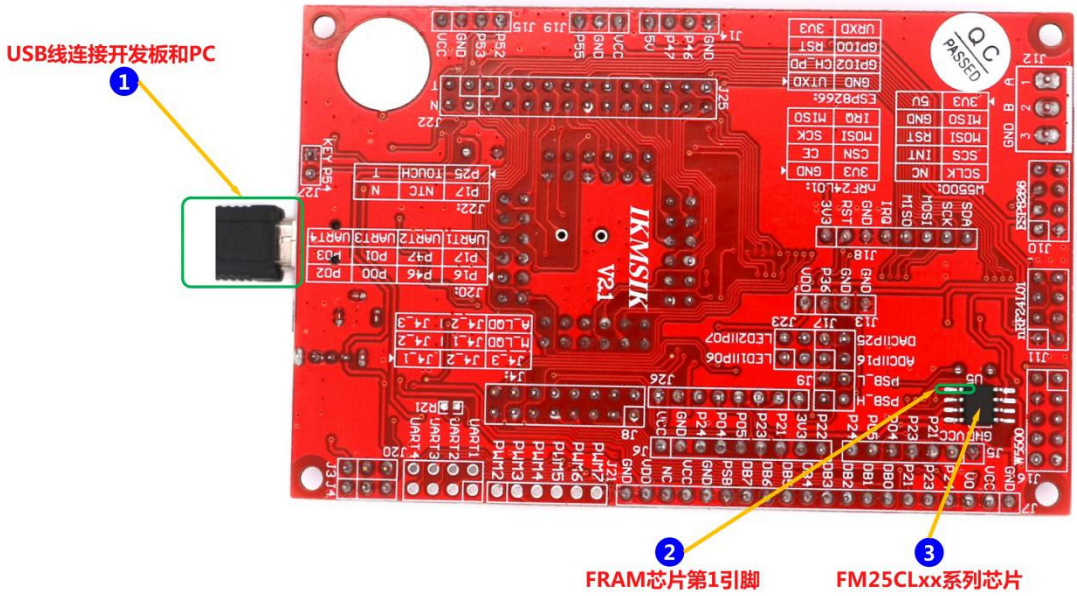

图 18：开发板连接图

4.7.4. 实验步骤

1. 解压“…\第 3 部分：配套例程源码\ 1 - 基础实验程序\”目录下的压缩文件“实验 2-14-4：外接 FRAM 存储器读写多字节实验（硬件 SPI）”，将解压后得到的文件夹拷贝到合适的目录，如“D\STC15”。
2. 启动 Keil C51。
3. 在 Keil C51 中执行“Project→Open Project”打开“…\FM25CLxx\project”目录下的工程“FM25CLxx.uvproj”。
4. 点击编译按钮编译工程。注意查看编译输出栏，观察编译的结果，如果有错误，修改程序，直到编译成功为止。编译后生成的 HEX 文件“FM25CLxx.hex”位于工程目录下的“Output”文件夹中。
5. 打开 STC-ISP 软件下载程序 。下载使用内部 IRC 时钟，IRC 频率选择为 11.0592MHZ。
6. 程序运行后，实验现象及步骤操作如下：

   1）在串口调试助手发送端发"3"，会在接收端显示 00 00 00 00 00 00 00 00 00 00。

   2）在串口调试助手发送端发"2"，会在接收端显示 33，在串口调试助手发送端发"3"，会在接收端显示 11 22 33 44 55 66 77 88 99 AA。

   3）在串口调试助手发送端发"4"，会在接收端显示 00，在串口调试助手发送端发"3"，会在接收端显示 00 00 00 00 00 00 00 00 00 00。