片外存储器 FLASH

1. 实验目的

- 掌握外部 FLASH 存储器 W25Q128 引脚定义和硬件电路设计。
- 掌握通过 STC15W4K32S4 系列单片机 SPI 外设对 W25Q128 的程序设计。

2. 实验内容

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

3. 硬件设计

3.1. 外部 FLASH 存储器介绍

FLASH 存储器属内存器件的一种，是一种非易失性（ Non-Volatile ）内存。该存储器结合了 ROM 和 RAM 的长处，不仅具备电可擦除可编程（EEPROM）的性能，还不会断电丢失数据，同时可以快速读取数据，所以现如今大多数单片机片内都集成了这种存储器。

接下来，我们主要介绍的是外部的 FLASH 存储器芯片，外部的 FLASH 存储器芯片种类和生产厂家都比较多。其中 WINBOND（华邦）公司生产的 W25Q 系列存储器应用很广泛，是我们介绍的重点。W25Q 系列存储器相比于普通的串行 FLASH 器件提供了更好的灵活性和性能，该系列存储器可在 2.7V 到 3.6V 的供电电压下工作（注意芯片后缀 JV），工作时电流消耗最低 4mA，在睡眠模式下消耗仅 1uA。另外，所有的 W25Q 系列存储器芯片都提供节省空间的封装。

WINBOND（华邦）公司生产的 W25Q 系列存储器支持标准 SPI 接口，操作寄存器指令兼容，但不同存储空间的芯片有不同的 ID，具体如下表。

表 1：W25Q 系列存储器介绍

序号	芯片型号	存储空间	供电电压	芯片 ID	工作温度	备注
1	W25Q16JV	16M bit	2.7V~3.6V	0xEF14	-40℃~85℃	
2	W25Q32JV	32M bit	2.7V~3.6V	0xEF15	-40℃~85℃	
3	W25Q64JV	64M bit	2.7V~3.6V	0xEF16	-40℃~85℃	
4	W25Q128JV	128M bit	2.7V~3.6V	0xEF17	-40℃~85℃	
5	W25Q256JV	256M bit	2.7V~3.6V	0xEF18	-40℃~85℃	
6	W25Q512JV	512M bit	2.7V~3.6V	0xEF19	-40℃~85℃	

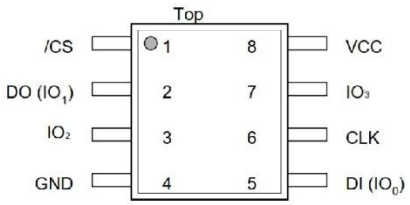
✧ 注：芯片 ID 中的高 8 位 0xEF 代表的 Winbond Serial Flash 系列。厂家还有 1G bit 和 2G bit 的存储器芯片 W25Q01JV 和 W25Q02JV，有需要更大存储空间可以考虑。

3.2. W25Q128JV 存储芯片介绍

3.2.1. 芯片引脚定义

W25Q128JV 支持标准的 SPI 接口，双线/四线 IO 模式 SPI：串行时钟，数据选择，串行数据 IO0（DI），IO1（DO），IO2（/WP），IO3（/HOLD）。

SPI 时钟频率最高可达 133MHz，因此在双 IO SPI 模式下等效于 266MHz，在四 IO SPI/QPI 模式下等效于 532MHz。这些传输速率比标准的异步 8 位或者 16 位并行 FLASH 存储器表现得更好。



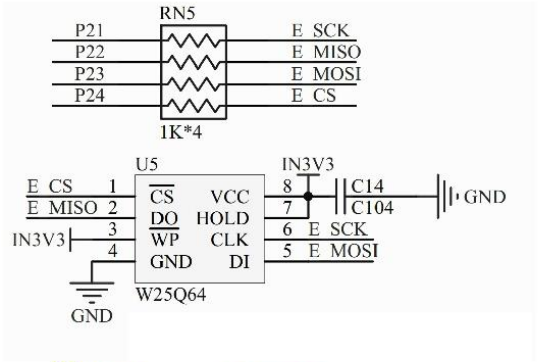
1 W25Q128JV芯片引脚

PAD NO.	PAD NAME	I/O	FUNCTION
1	/CS	I	Chip Select Input
2	DO (IO1)	I/O	Data Output (Data Input Output 1)(1)
3	IO2	I/O	Data Input Output 2(2)
4	GND		Ground
5	DI (IO0)	I/O	Data Input (Data Input Output 0)(1)
6	CLK	I	Serial Clock Input
7	IO3	I/O	Data Input Output 3(2)
8	VCC		Power Supply

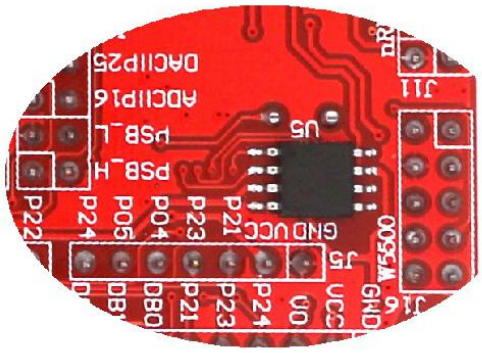
2 W25Q128JV芯片引脚定义

图 1：W25Q128JV 芯片引脚

进取者 STC15 开发板上设计了可供用户焊接使用的 W25Q 系列存储器芯片的接口，其中原理图部分及硬件实物部分如下。



1 外部FLASH电路原理图



2 外部FLASH芯片焊接实物照

图 2：开发板外部 FLASH 电路

3.2.2. 芯片介绍及使用注意事项

W25Q128JV 存储芯片是由 65536 可编程的页组成的，每页有 256 个字节。一次最多可以写 256 个字节。可以一次擦除 16 页（4K 字节）、128 页（32K 字节）、256 页(64K 字节)或者一整片。W25Q128JV 有 4096 个可擦除的扇区，256 个可擦除的块。4K 字节的扇区对于数据和参数存储有更高的灵活性。

■ **W25Q128JV 内部框图**

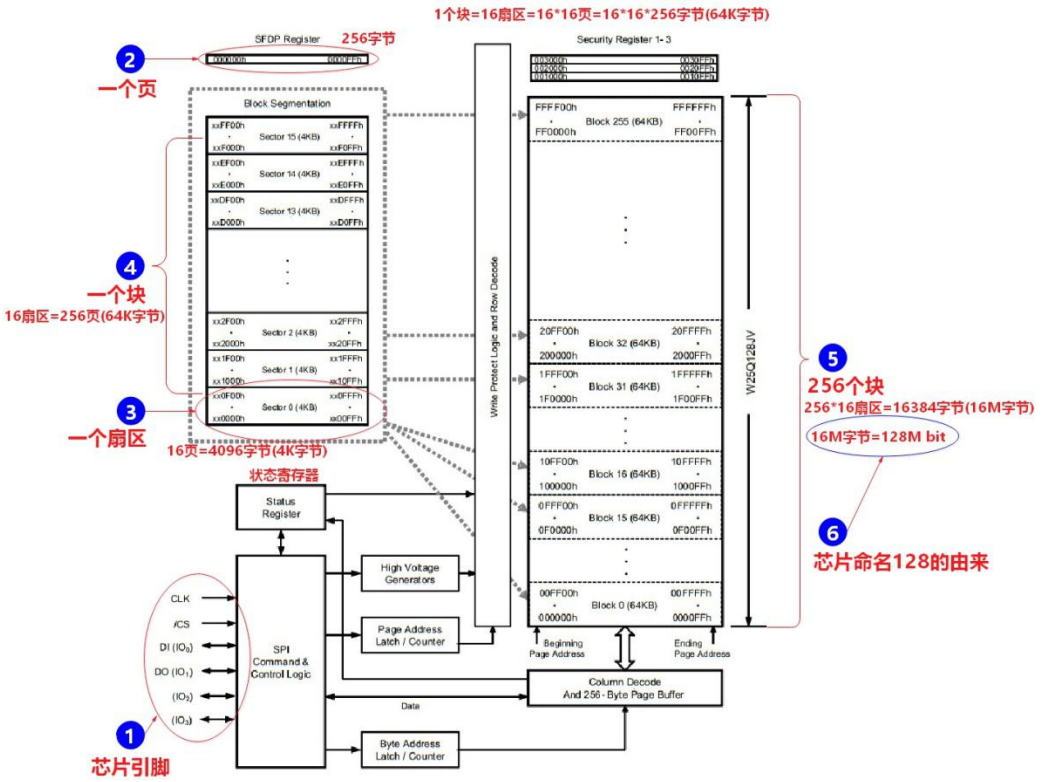


图 3：W25Q128JV 芯片内部框图

✧ 注：W25Q128JV 存储器实际的存储空间会略大于 128M bit 的，因为比如 4096 字节都按照 4K 来计算的。其他 W25Q 系列存储器的页、扇区、块计算都和上面 W25Q128JV 内部框图一样，不一样的是不同的存储器块数不一样。比如，W25Q64JV 存储器只有 128 个块，也就是 8M 字节存储空间（即 64M bit）。

■ **使用注意事项**

1） 由于 FLASH 的物理特性，决定了 FLASH 每一位的操作只能从 1 变为 0（写操作）。

2） 大多数 FLASH 芯片或单片机内未使用 FLASH 存储空间每一位出厂默认都是 1。

3） 对 FLASH 写操作之前必须将待操作 FLASH 空间数据都置为 1。

4） 对 FLASH 的擦除操作即是把待操作的空间的每一位都置为 1。

5） 所以 FLASH 写操作前需有擦除操作。

6） W25Q128JV 存储芯片的擦除比较灵活，可以按扇区、块甚至是整片擦除。（擦除是需要时间的，比如整片擦除约用时几十秒）

✧ 注：有些用户说我没有擦除直接写的，第一次没有问题，之后怎么出错了？想一想什么原因。（往往第一次是出厂的默认的每个空间位都是 1，所以写操作没有问题）

4. 软件设计

关于 STC15W4K32S4 系列 SPI 外设原理及相关寄存器的介绍请参考外接 FRAM 存储器实验部分，对外接 FLASH 存储器的 SPI 配置是完全一样的。

4.1.外接 FLASH 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 2-14-1：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 2-15-1：外接 FLASH 存储器读写单字节实验（模拟 SPI）”。

4.1.1. 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 2：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	W25q128	.c	SPI 通信及操作 FLASH 有关的用户自定义函数。
3	delay	.c	包含用户自定义延时函数。

4.1.2. 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 " w25q128.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 3：头文件包含路径

序号	路径	描述
1	..\ Source	uart.h、w25q128.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。

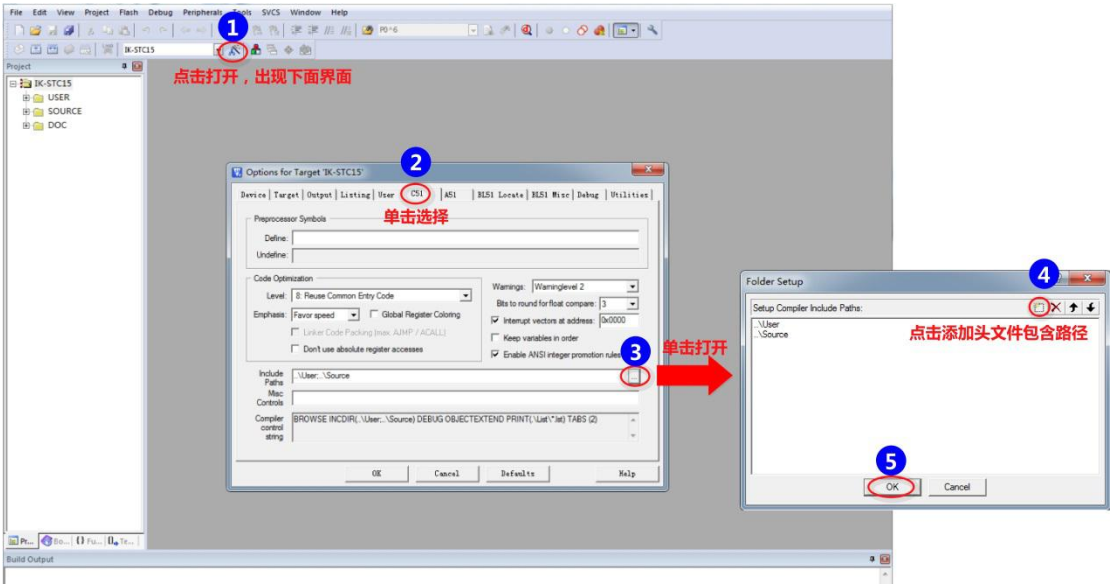


图 4：添加头文件包含路径

4.1.3. 编写代码

首先，在 w25q128.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

```
/*****************************************************************************
 * 描  述 : 模拟SPI写入一个字节
 * 入  参 : uint8 date
 * 返回值 : 无
 *****************************************************************************/
void SPI_WriteByte(uint8 date)
{
  uint8 temp,i;
  temp = date;

  for (i = 0; i < 8; i++)
    {
     SPI_SCK_0;
         delay_us(10);
     if((temp&0x80)==0x80)
     { SPI_MOSI_1; }
     else
     { SPI_MOSI_0; }
     SPI_SCK_1 ;
         delay_us(10);
     temp <<= 1;
   }
```

```
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 FLASH 存储器的基本操作函数，如下表所示。

表 4：FLASH 相关用户函数汇集

序号	函数名	功能描述
1	WriteEnable	FLASH 芯片写使能。
2	WriteDisable	FLASH 芯片写禁止。
3	W25Q_ReadID	读 FLASH 芯片 ID。
4	W25Q_ReadStatus	读 FLASH 状态寄存器。
5	W25Q_WriteStatus	写 FLASH 状态寄存器。
6	W25Q_Write_Page	在一页内写入多字节数据。
7	W25Q_Write_N	向 FLASH 指定地址存入多字节数据。

8	W25Q_Read_N	从 FLASH 指定地址读取多字节数据。
9	W25Q_SectorErase	按扇区擦除数据。
10	W25Q_BlockErase	按块擦除数据。
11	W25Q_ChipErase	擦除整片芯片。

关于每个操作外部 FLASH 相关用户函数，下面给出详细代码。

程序清单：FLASH 芯片写使能函数

```
/*****************************************************************************
 * 描  述 : 写使能（将 WEL 置位）
 * 入  参 : 无
 * 返回值 : 无
 *****************************************************************************/
void WriteEnable  (void)
{
    SPI_CS_0;
    SPI_WriteByte(W25X_WriteEnable);
    SPI_CS_1;
}
```

程序清单：FLASH 芯片写禁止函数

```
/*****************************************************************************
 * 描  述 : 写禁止（将 WEL 清 0）
 * 入  参 : 无
 * 返回值 : 无
 *****************************************************************************/
void WriteDisable (void)
{
    SPI_CS_0;
    SPI_WriteByte(W25X_WriteDisable);
    SPI_CS_1;
}
```

程序清单：读 FLASH 芯片 ID 函数

```
/*****************************************************************************
 * 描  述 : 读取存储器芯片 ID
 * 入  参 : 无
 * 返回值 : uint16 ID
备注：W25Q16 的 ID:0XEF14  W25Q32 的 ID:0XEF15  W25Q64 的 ID:0XEF16  W25Q128 的
ID:0XEF17
*****************************************************************************/
uint16 W25Q_ReadID(void)
```

```
{
    uint16 Temp = 0;
  uint8 Temp1 = 0;
    uint8   Temp2 = 0;
    SPI_CS_0;
    SPI_WriteByte(0x90);        //发送读取 ID 命令
    SPI_WriteByte(0x00);
    SPI_WriteByte(0x00);
    SPI_WriteByte(0x00);
    Temp1|=SPI_ReadByte();
    Temp2|=SPI_ReadByte();
    Temp = Temp1*256+Temp2;
    SPI_CS_1;
    return Temp;
}
```

程序清单：读 FLASH 状态寄存器函数

```
/*********************************************************************
 * 描  述 : 读取存储器芯片的状态
 * 入  参 : 无
 * 返回值 : uint8 状态寄存器数据字节
备注：芯片内部状态寄存器第 0 位=0 表示空闲，0 位=1 表示忙
*********************************************************************/
uint8 W25Q_ReadStatus(void)
{
    uint8 status=0;
    SPI_CS_0;
    SPI_WriteByte(W25X_ReadStatus);    // 0x05 读取状态的命令字
    status=SPI_ReadByte();             // 读取状态字节
    SPI_CS_1;
    return status;
}
```

程序清单：写 FLASH 状态寄存器函数

```
/*********************************************************************
 * 描  述 : 写芯片的状态寄存器
 * 入  参 : uint8 Status
 * 返回值 : 无
备注：只有 SPR,TB,BP2,BP1,BP0(bit 7,5,4,3,2)可以写!!!
*********************************************************************/
void W25Q_WriteStatus(uint8 Status)
{
```

```
    SPI_CS_0;
    SPI_WriteByte(W25X_WriteStatus);  // 0x01 读取状态的命令字
    SPI_WriteByte(Status);            // 写入一个字节
    SPI_CS_1;
}
```

程序清单：在一页内写入多字节数据函数

```
/*****************************************************************************
 * 描  述 : 在一页(0~65535)内写入少于 256 个字节的数据(在指定地址开始写入最大 256 字节的数据)
 * 入  参 : pbuf:数据存储区  WriteAddr:开始写入的地址(24bit)  Len:要写入的字节数(最大 256)
 * 返回值 : 无
备注：Len:要写入的字节数,该数不应该超过该页的剩余字节数!!!
*****************************************************************************/
void W25Q_Write_Page(uint8* pbuf,uint32 WriteAddr,uint16 Len)
{
   uint16 i;
   while(W25Q_ReadStatus()&0x01);         //判断是否忙
   WriteEnable();                           //写使能
   SPI_CS_0;                                //使能器件
   SPI_WriteByte(W25X_Writepage);           //发送写页命令
   SPI_WriteByte((uint8)((WriteAddr)>>16));   //发送 24bit 地址
   SPI_WriteByte((uint8)((WriteAddr)>>8));
   SPI_WriteByte((uint8)WriteAddr);
   for(i=0;i<Len;i++)                       //循环写数
   {
        SPI_WriteByte(*pbuf++);
   }
   SPI_CS_1;                                //取消片选
   while(W25Q_ReadStatus()&0x01);         //等待写入结束
}
```

程序清单：向 FLASH 指定地址存入多字节数据函数

```
/*****************************************************************************
 * 描  述 : 在指定地址开始写入指定长度的数据
 * 入  参 : pbuf:数据存储区  WriteAddr:开始写入的地址(24bit)  Len:要写入的字节数(最大 65535)
 * 返回值 : 无
备注：必须确保所写的地址范围内的数据全部为 0XFF,否则在非 0XFF 处写入的数据将失败!
*****************************************************************************/
```

```
void W25Q_Write_N(uint8 * pbuf,uint32 WriteAddr,uint16 Len)
{
    uint16 PageLen;                 // 页内写入字节长度
    PageLen=256-WriteAddr%256;      // 单页剩余的字节数 （单页剩余空间）
    if(Len<=PageLen) PageLen=Len;   // 不大于 256 个字节
    while(1)
    {
        W25Q_Write_Page(pbuf,WriteAddr,PageLen);
        if(PageLen==Len)break;          // 写入结束了
        else
        {
            pbuf+=PageLen;
            WriteAddr+=PageLen;
            Len-=PageLen;               //  减去已经写入了的字节数
            if(Len>256)PageLen=256;     // 一次可以写入 256 个字节
            else PageLen=Len;           // 不够 256 个字节了
        }
    }
}
```

程序清单：从 FLASH 指定地址读取多字节数据函数

```
/*****************************************************************************
 * 描  述 : 在指定地址开始写入指定长度的数据
 * 入  参 : pbuf:数据存储区  WriteAddr:开始写入的地址(24bit)  Len:要写入的字节数(最大 65535)
 * 返回值 : 无
备注：必须确保所写的地址范围内的数据全部为 0XFF,否则在非 0XFF 处写入的数据将失败!
*****************************************************************************/
void W25Q_Write_N(uint8 * pbuf,uint32 WriteAddr,uint16 Len)
{
    uint16 PageLen;                 // 页内写入字节长度
    PageLen=256-WriteAddr%256;      // 单页剩余的字节数 （单页剩余空间）
    if(Len<=PageLen) PageLen=Len;   // 不大于 256 个字节
    while(1)
    {
        W25Q_Write_Page(pbuf,WriteAddr,PageLen);
        if(PageLen==Len)break;          // 写入结束了
        else
        {
            pbuf+=PageLen;
            WriteAddr+=PageLen;
            Len-=PageLen;               //  减去已经写入了的字节数
```

```
            if(Len>256)PageLen=256;     // 一次可以写入 256 个字节
            else PageLen=Len;           // 不够 256 个字节了
        }
    }
}
```

程序清单：按扇区擦除数据函数

```
/*****************************************************************
 * 描  述 : 擦除一个扇区( 4K 扇擦除)
 * 入  参 : uint32 Addr24  扇区地址
 * 返回值 : 无
备注：擦除一个扇区的最少时间:150ms
*****************************************************************/
void W25Q_SectorErase(uint32 Addr24) //擦除资料图示的 4KB 空间
{
    unsigned char Addr1;        // 最低地址字节
    unsigned char Addr2;        // 中间地址字节
    unsigned char Addr3;        // 最高地址字节
    Addr1=Addr24;
    Addr24=Addr24>>8;
    Addr2=Addr24;
    Addr24=Addr24>>8;
    Addr3=Addr24;                       // 把地址拆开来
    while(W25Q_ReadStatus()&0x01);      // 判断是否忙
    WriteEnable();                      // 写允许
    SPI_CS_0;
    SPI_WriteByte(W25X_S_Erase);        // 整扇擦除命令
    SPI_WriteByte(Addr3);
    SPI_WriteByte(Addr2);
    SPI_WriteByte(Addr1);
    SPI_CS_1;
    while(W25Q_ReadStatus()&0x01);    // 等待擦除完成
}
```

程序清单：按块擦除数据函数

```
/*****************************************************************
 * 描  述 : 擦除一块擦除( 64K)
 * 入  参 : uint32 Addr24  扇区地址
 * 返回值 : 无
*****************************************************************/
void W25Q_BlockErase(uint32 Addr24)  //擦除资料图示的 64KB 空间
```

```
{
    uint8 Addr1;        // 最低地址字节
    uint8 Addr2;        // 中间地址字节
    uint8 Addr3;        // 最高地址字节
    Addr1=Addr24;
    Addr24=Addr24>>8;
    Addr2=Addr24;
    Addr24=Addr24>>8;
    Addr3=Addr24;                       // 把地址拆开来
    while(W25Q_ReadStatus()&0x01);      // 判断是否忙
    WriteEnable();                      // 写允许
    SPI_CS_0;
    SPI_WriteByte(W25X_B_Erase);        // 整扇擦除命令
    SPI_WriteByte(Addr3);
    SPI_WriteByte(Addr2);
    SPI_WriteByte(Addr1);
    SPI_CS_1;
    while(W25Q_ReadStatus()&0x01);    // 等待擦除完成
}
```

程序清单：擦除整片芯片函数

```
/*****************************************************************************
 * 描  述 : 擦除整片芯片
 * 入  参 : 无
 * 返回值 : 无
备注：不同型号的芯片时间不一样
*****************************************************************************/
void W25Q_ChipErase(void)
{
    while(W25Q_ReadStatus()&0x01);      // 判断是否忙
    WriteEnable();                      // 写允许
    SPI_CS_0;
    SPI_WriteByte(W25X_C_Erase);        // 整片擦除命令
    SPI_CS_1;                           // 从 CS=1 时开始执行擦除
    while(W25Q_ReadStatus()&0x01);      // 等待擦除完成
}
```

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 FLASH 的单字节读、写及擦除等操作。

代码清单：主函数

```
int main()
{
    uint16  Temp;
    uint8   Temp1,Temp2;

////////////////////////////////////////////////
//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

    Temp=W25Q_ReadID();                             //读取外扩存储器芯片 ID
    Temp1=(uint8)(Temp/256);            //将读取的存储器芯片 ID 高字节存放到 Temp1 中
    Temp2=(uint8)Temp;                  //将读取的存储器芯片 ID 低字节存放到 Temp2 中

  while (1)
    {
        if(W_ID)                                //读存储器芯片 ID 模式
        {
            W_ID=0;                             //ID 标志变量清零,发送一次
            SendDataByUart1(Temp1);             //串口 1 发送芯片 ID 高字节
            SendDataByUart1(Temp2);             //串口 1 发送芯片 ID 低字节
         }
     if(WriteFLAG)                      //写模式
        {
            WriteFLAG=0;                        //写标志变量清零,发送一次
            W25Q_Write_N(scan,0x00000010,1);  //向 FLASH 地址 0x00000010 中写入单字
节数据 0x33
        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
            W25Q_Read_N(buffer,0x00000010,1);   //从 FLASH 地址 0x00000010 读取单字
节数据并存入到 buffer 数组中
```

```
42.          SendStringByUart1_n(buffer,1);       //串口 1 发送读取的字符
43.           }
44.           if(ClearFLAG)                       //扇区擦除模式
45.           {
46.                ClearFLAG=0;                        //清除标志变量清零,发送一次
47.                W25Q_SectorErase(0x00000010);       //擦除一个扇区（0x00000010 在的扇区）
48.          SendDataByUart1(0x00);                //串口 1 发送数据 0x00 表示擦除完成
49.           }
50.     }
51. }
```

4.1.4. 硬件连接

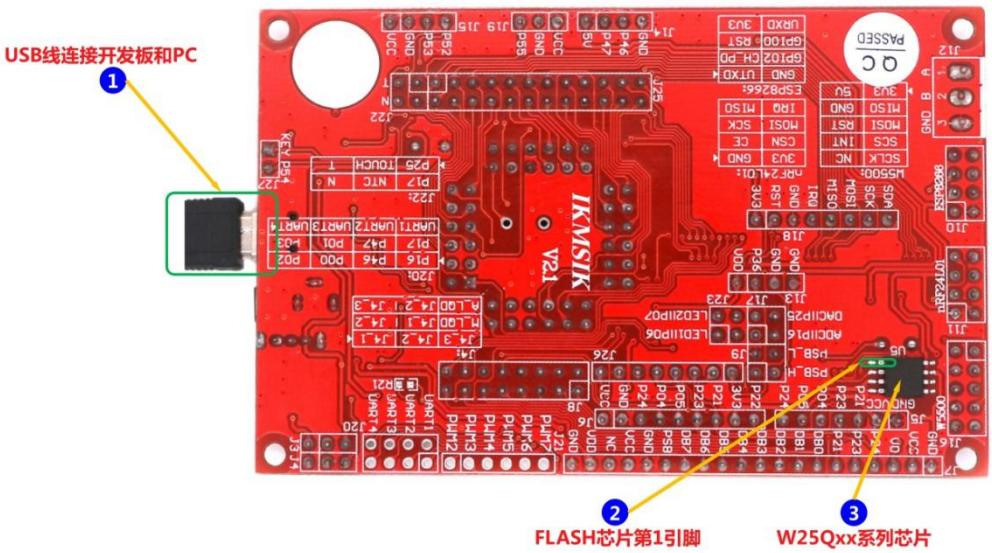


图 5：开发板连接图

4.1.5. 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 2-15-1：外接 FLASH 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“…\W25Qxx\project”目录下的工程“W25Qxx.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“W25Qxx.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. 打开串口调试助手选择正确的串口号，波特率设置为 9600，数据位为 8、停止位为 1，选择 HEX 显示。

7. 实验现象及步骤操作如下：

1）在发送区写 5，则在接收窗口可看到读取的外部 FLASH 芯片的 ID： EF 17。

2）在发送区写 3，则在接收窗口可看到读取的指定存储单元的数值： FF 。

3）在发送区写 4，则在接收窗口可看到表示擦除操作完成的命令： 00 。

4）在发送区写 3，则在接收窗口可看到读取的指定存储单元的数值： FF 。

5）在发送区写 2，则在接收窗口可看到表示写操作完成的命令： 33 。

6）在发送区写 3，则在接收窗口可看到读取的指定存储单元的数值： 33 。

7）在发送区写 4，则在接收窗口可看到表示擦除操作完成的命令： 00 。

8）在发送区写 3，则在接收窗口可看到读取的指定存储单元的数值： FF 。

4.2. 外接 FLASH 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 2-15-1：外接 FLASH 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 2-15-2：外接 FLASH 存储器读写多字节实验（模拟 SPI）”。

4.2.1. 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 2-15-1：外接 FLASH 存储器读写单字节实验（模拟 SPI）”部分。

4.2.2. 编写代码

首先，在 w25q128.c 文件中编写模拟 SPI 方式的读写字节函数和对 FLASH 存储器的基本操作函数。请参考“实验 2-15-1：外接 FLASH 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 FLASH 的多字节读、写及擦除等操作。

代码清单：主函数

```
1.  int main()
2.  {
3.      uint16  Temp;
4.      uint8   Temp1,Temp2;
5.
6.  /////////////////////////////////////////////////
7.  //注意: STC15W4K32S4 系列的芯片,上电后所有与 PWM 相关的 IO 口均为
8.  //       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
9.  //相关 IO: P0.6/P0.7/P1.6/P1.7/P2.1/P2.2
10. //         P2.3/P2.7/P3.7/P4.2/P4.4/P4.5
11. /////////////////////////////////////////////////
12.     P2M1 &= 0xE1;   P2M0 &= 0xE1;                   //设置 P2.1~P2.4 为准双向口
13.     P3M1 &= 0xFC;   P3M0 &= 0xFC;                   //设置 P3.0~P3.1 为准双向口
14.
15.     SPI_CS_1;                                       //SPI 使能引脚初始化
```

```
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.     Temp=W25Q_ReadID();                             //读取外扩存储器芯片 ID
21.     Temp1=(uint8)(Temp/256);                //将读取的存储器芯片 ID 高字节存放到 Temp1 中
22.     Temp2=(uint8)Temp;                      //将读取的存储器芯片 ID 低字节存放到 Temp2 中
23.
24.   while (1)
25.     {
26.         if(W_ID)                                //读存储器芯片 ID 模式
27.         {
28.               W_ID=0;                           //ID 标志变量清零,发送一次
29.               SendDataByUart1(Temp1);           //串口 1 发送芯片 ID 高字节
30.               SendDataByUart1(Temp2);           //串口 1 发送芯片 ID 低字节
31.          }
32.      if(WriteFLAG)                      //写模式
33.          {
34.               WriteFLAG=0;                      //写标志变量清零,发送一次
35.               W25Q_SectorErase(0x00000000);     //擦除一个扇区（0x00000000 在的扇区）
36.               W25Q_Write_N(scan,0x00000000,10); //向 FLASH 地址 0x00000000 中写入
   scan 数组中 10 个字节数据
37.              SendDataByUart1(0x33);           //串口 1 发送数据 0x33 表示写操作完成
38.          }
39.          if(ReadFLAG)                       //读模式
40.          {
41.               ReadFLAG=0;                       //读标志变量清零,发送一次
42.               W25Q_Read_N(buffer,0x00000000,10);      //从 FLASH 地址 0x00000000 开
   始读取 10 字节数据并存入到 buffer 数组中
43.         SendStringByUart1_n(buffer,10);      //串口 1 发送数组 buffer 中的值（即读取的
   多字节数据）
44.          }
45.          if(ClearFLAG)                      //扇区擦除模式
46.          {
47.               ClearFLAG=0;                      //清除标志变量清零,发送一次
48.               W25Q_SectorErase(0x00000000);     //擦除一个扇区（0x00000000 在的扇区）
49.              SendDataByUart1(0x00);            //串口 1 发送数据 0x00 表示擦除完成
50.          }
51.     }
52. }
```

4.2.3. 硬件连接

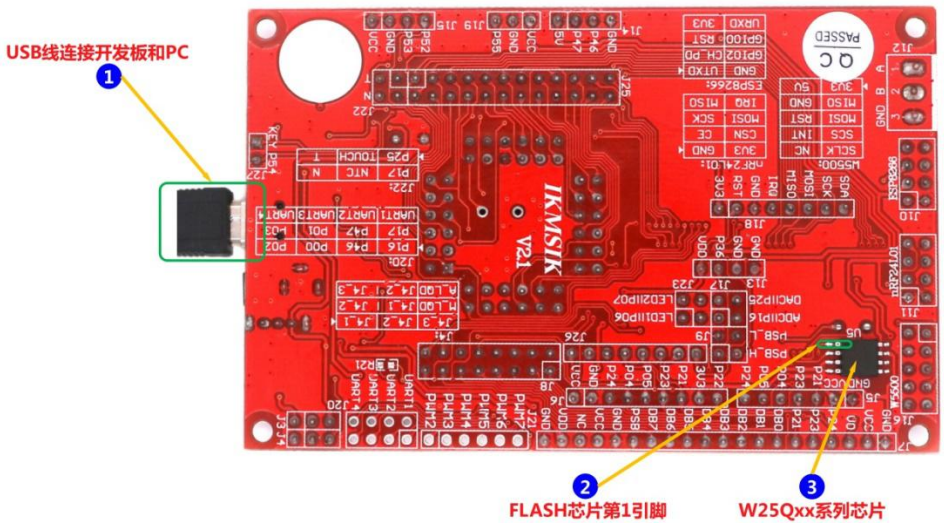


图 6：开发板连接图

4.2.4. 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 2-15-2：外接 FLASH 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“…\W25Qxx\project”目录下的工程“W25Qxx.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“W25Qxx.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. 打开串口调试助手选择正确的串口号，波特率设置为 9600，数据位为 8、停止位为 1，选择 HEX 显示。
7. 实验现象及步骤操作如下：

   1）在发送区写 5，则在接收窗口可看到读取的外部 FLASH 芯片的 ID： EF 17。

   2）在发送区写 3，则在接收窗口可看到读取的指定存储单元的数值： FF FF FF FF FF FF FF FF FF FF 。

   3）在发送区写 2，则在接收窗口可看到表示写操作完成的命令： 33 。

   4）在发送区写 3，则在接收窗口可看到读取的指定存储单元的数值： 11 22 33 44 55 66 77 88 99 AA 。

   5）在发送区写 4，则在接收窗口可看到表示擦除操作完成的命令： 00 。

   6）在发送区写 3，则在接收窗口可看到读取的指定存储单元的数值： FF FF FF FF FF FF FF FF FF FF 。

4.3.外接 FLASH 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 2-15-1：外接 FLASH 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 2-15-3：外接 FLASH 存储器读写单字节实验（硬件 SPI）”。

4.3.1. 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 5：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	w25q128	.c	SPI 通信及操作 FLASH 有关的用户自定义函数。
3	delay	.c	包含用户自定义延时函数。

4.3.2. 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 " w25q128.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 6：头文件包含路径

序号	路径	描述
1	..\ Source	uart.h、w25q128.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。

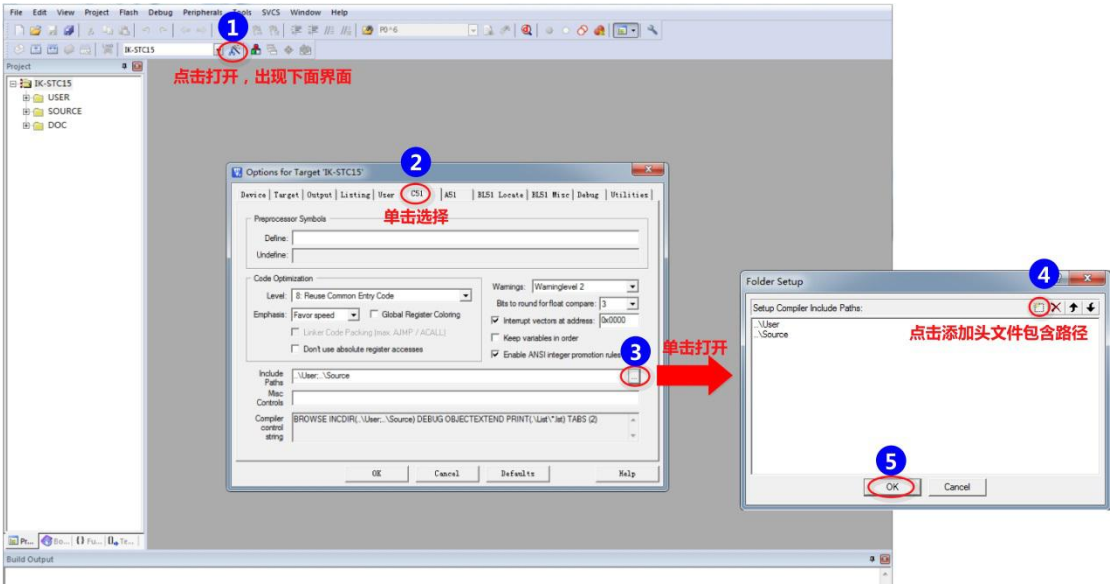


图 7：添加头文件包含路径

4.3.3. 编写代码

首先，在 w25q128.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

```
1.  /*********************************************************************************
2.   * 描   述 : 硬件 SPI 初始化
3.   * 入   参 : 无
4.   * 返回值 : 无
5.   *********************************************************************************/
6.  void Init_SPI(void)
7.  {
8.     AUXR1|=0X04;                              //将 SPI 调整到 P2.1 P2.2 P2.3
9.     AUXR1&=0XF7;
10.    SPDAT = 0;
11.    SPSTAT = SPIF | WCOL;                    //清除 SPI 状态位
12.    SPCTL = SPEN | MSTR | SSIG;              //主机模式
13. }
```

程序清单：硬件 SPI 读写字节函数

```
1.  /*********************************************************************************
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)
```

```
{
    SPDAT = SPI_SendData;          //触发 SPI 发送数据
    while (!(SPSTAT & SPIF));      //等待发送完成
    SPSTAT = SPIF | WCOL;          //清除 SPI 状态位
    return SPDAT;                  //返回 SPI 数据
}
```

然后，编写对 FLASH 存储器的基本操作函数，如下表所示。

表 7：FLASH 相关用户函数汇集

序号	函数名	功能描述
1	WriteEnable	FLASH 芯片写使能。
2	WriteDisable	FLASH 芯片写禁止。
3	W25Q_ReadID	读 FLASH 芯片 ID。
4	W25Q_ReadStatus	读 FLASH 状态寄存器。
5	W25Q_WriteStatus	写 FLASH 状态寄存器。
6	W25Q_Write_Page	在一页内写入多字节数据。
7	W25Q_Write_N	向 FLASH 指定地址存入多字节数据。
8	W25Q_Read_N	从 FLASH 指定地址读取多字节数据。
9	W25Q_SectorErase	按扇区擦除数据。
10	W25Q_BlockErase	按块擦除数据。
11	W25Q_ChipErase	擦除整片芯片。

关于每个操作外部 FLASH 相关用户函数，下面给出详细代码。

程序清单：FLASH 芯片写使能函数

```
/*****************************************************************************
 * 描   述 : 写使能（将 WEL 置位）
 * 入   参 : 无
 * 返回值 : 无
 *****************************************************************************/
void WriteEnable (void)
{
    SPI_CS_0;
    SPI_SendByte(W25X_WriteEnable);
    SPI_CS_1;
}
```

程序清单：FLASH 芯片写禁止函数

```
/*****************************************************************************
 * 描  述 : 写禁止（将 WEL 清 0）
 * 入  参 : 无
 * 返回值 : 无
 *****************************************************************************/
void WriteDisable (void)
{
    SPI_CS_0;
    SPI_SendByte(W25X_WriteDisable);
    SPI_CS_1;
}
```

程序清单：读 FLASH 芯片 ID 函数

```
/*****************************************************************************
 * 描  述 : 读取存储器芯片 ID
 * 入  参 : 无
 * 返回值 : uint16 ID
备注：W25Q16 的 ID:0XEF14  W25Q32 的 ID:0XEF15  W25Q64 的 ID:0XEF16  W25Q128 的 ID:0XEF17
*****************************************************************************/
uint16 W25Q_ReadID(void)
{
    uint16 Temp = 0;
  uint8 Temp1 = 0;
    uint8  Temp2 = 0;
    SPI_CS_0;
    SPI_SendByte(0x90);         //发送读取 ID 命令
    SPI_SendByte(0x00);
    SPI_SendByte(0x00);
    SPI_SendByte(0x00);
    Temp1|=SPI_SendByte(0xFF);
    Temp2|=SPI_SendByte(0xFF);
    Temp = Temp1*256+Temp2;
    SPI_CS_1;
    return Temp;
}
```

程序清单：读 FLASH 状态寄存器函数

```
/*****************************************************************************
 * 描  述 : 读取存储器芯片的状态
 * 入  参 : 无
 * 返回值 : uint8 状态寄存器数据字节
备注：芯片内部状态寄存器第 0 位=0 表示空闲，0 位=1 表示忙
```

```
6.  *****************************************************************/
7.  uint8 W25Q_ReadStatus(void)
8.  {
9.      uint8 status=0;
10.     SPI_CS_0;
11.     SPI_SendByte(W25X_ReadStatus);    // 0x05 读取状态的命令字
12.     status=SPI_SendByte(0xFF);              // 读取状态字节
13.     SPI_CS_1;
14.     return status;
15. }
```

程序清单：写 FLASH 状态寄存器函数

```
1.  /*****************************************************************************
2.   * 描   述 : 写芯片的状态寄存器
3.   * 入   参 : uint8 Status
4.   * 返回值 : 无
5.  备注：只有 SPR,TB,BP2,BP1,BP0(bit 7,5,4,3,2)可以写!!!
6.  *****************************************************************************/
7.  void W25Q_WriteStatus(uint8 Status)
8.  {
9.      SPI_CS_0;
10.     SPI_SendByte(W25X_WriteStatus);  // 0x01 读取状态的命令字
11.     SPI_SendByte(Status);            // 写入一个字节
12.     SPI_CS_1;
13. }
```

程序清单：在一页内写入多字节数据函数

```
1.  /*****************************************************************************
2.   * 描   述 : 在一页(0~65535)内写入少于 256 个字节的数据(在指定地址开始写入最大 256 字节的
    数据)
3.   * 入   参 : pbuf:数据存储区   WriteAddr:开始写入的地址(24bit)   Len:要写入的字节数(最
    大 256)
4.   * 返回值 : 无
5.  备注：Len:要写入的字节数,该数不应该超过该页的剩余字节数!!!
6.  *****************************************************************************/
7.  void W25Q_Write_Page(uint8* pbuf,uint32 WriteAddr,uint16 Len)
8.  {
9.     uint16 i;
10.    while(W25Q_ReadStatus()&0x01);         //判断是否忙
11.    WriteEnable();                           //写使能
12.    SPI_CS_0;                                //使能器件
```

```
    SPI_SendByte(W25X_Writepage);           //发送写页命令
    SPI_SendByte((uint8)((WriteAddr)>>16));   //发送 24bit 地址
    SPI_SendByte((uint8)((WriteAddr)>>8));
    SPI_SendByte((uint8)WriteAddr);
    for(i=0;i<Len;i++)                      //循环写数
    {
        SPI_SendByte(*pbuf++);
    }
    SPI_CS_1;                               //取消片选
    while(W25Q_ReadStatus()&0x01);          //等待写入结束
}
```

程序清单：向 FLASH 指定地址存入多字节数据函数

```
/*****************************************************************************
 * 描  述 : 在指定地址开始写入指定长度的数据
 * 入  参 : pbuf:数据存储区  WriteAddr:开始写入的地址(24bit)  Len:要写入的字节数(最大 65535)
 * 返回值 : 无
备注：必须确保所写的地址范围内的数据全部为 0XFF,否则在非 0XFF 处写入的数据将失败!
*****************************************************************************/
void W25Q_Write_N(uint8 * pbuf,uint32 WriteAddr,uint16 Len)
{
    uint16 PageLen;                 // 页内写入字节长度
    PageLen=256-WriteAddr%256;      // 单页剩余的字节数 （单页剩余空间）
    if(Len<=PageLen) PageLen=Len;   // 不大于 256 个字节
    while(1)
    {
        W25Q_Write_Page(pbuf,WriteAddr,PageLen);
        if(PageLen==Len)break;          // 写入结束了
        else
        {
            pbuf+=PageLen;
            WriteAddr+=PageLen;
            Len-=PageLen;               //  减去已经写入了的字节数
            if(Len>256)PageLen=256;     // 一次可以写入 256 个字节
            else PageLen=Len;           // 不够 256 个字节了
        }
    }
}
```

程序清单：从 FLASH 指定地址读取多字节数据函数

```
/*****************************************************************************
 * 描  述 : 在指定地址开始读取指定长度的数据
 * 入  参 : pbuf:数据存储区  ReadAddr:开始读取的地址(24bit)  Len:要读取的字节数(最大65535)
 * 返回值 : 无
*****************************************************************************/
void W25Q_Read_N(uint8 * pbuf,uint32 ReadAddr,uint16 Len)
{
    uint16 i;
    while(W25Q_ReadStatus()&0x01);          // 判断是否忙
    SPI_CS_0;                               // 使能器件
    SPI_SendByte(W25X_ReadDATA8);          // 发送读取命令
    SPI_SendByte((uint8)((ReadAddr)>>16));  // 发送 24bit 地址
    SPI_SendByte((uint8)((ReadAddr)>>8));
    SPI_SendByte((uint8)ReadAddr);
    for(i=0;i<Len;i++)
    {
       *pbuf++=SPI_SendByte(0xFF);             // 读一个字节
    }
    SPI_CS_1;                               // 取消片选
}
```

程序清单：按扇区擦除数据函数

```
/*****************************************************************************
 * 描  述 : 擦除一个扇区( 4K 扇擦除)
 * 入  参 : uint32 Addr24  扇区地址
 * 返回值 : 无
备注：擦除一个扇区的最少时间:150ms
*****************************************************************************/
void W25Q_SectorErase(uint32 Addr24) //擦除资料图示的 4KB 空间
{
    unsigned char Addr1;        // 最低地址字节
    unsigned char Addr2;        // 中间地址字节
    unsigned char Addr3;        // 最高地址字节
    Addr1=Addr24;
    Addr24=Addr24>>8;
    Addr2=Addr24;
    Addr24=Addr24>>8;
    Addr3=Addr24;                       // 把地址拆开来
    while(W25Q_ReadStatus()&0x01);      // 判断是否忙
    WriteEnable();                      // 写允许
    SPI_CS_0;
```

```
    SPI_SendByte(W25X_S_Erase);        // 整扇擦除命令
    SPI_SendByte(Addr3);
    SPI_SendByte(Addr2);
    SPI_SendByte(Addr1);
    SPI_CS_1;
    while(W25Q_ReadStatus()&0x01);    // 等待擦除完成
}
```

程序清单：按块擦除数据函数

```
/*****************************************************************************
 * 描  述 : 擦除一块擦除( 64K)
 * 入  参 : uint32 Addr24  扇区地址
 * 返回值 : 无
 *****************************************************************************/
void W25Q_BlockErase(uint32 Addr24)  //擦除资料图示的 64KB 空间
{
    uint8 Addr1;        // 最低地址字节
    uint8 Addr2;        // 中间地址字节
    uint8 Addr3;        // 最高地址字节
    Addr1=Addr24;
    Addr24=Addr24>>8;
    Addr2=Addr24;
    Addr24=Addr24>>8;
    Addr3=Addr24;                      // 把地址拆开来
    while(W25Q_ReadStatus()&0x01);     // 判断是否忙
    WriteEnable();                     // 写允许
    SPI_CS_0;
    SPI_SendByte(W25X_B_Erase);        // 整扇擦除命令
    SPI_SendByte(Addr3);
    SPI_SendByte(Addr2);
    SPI_SendByte(Addr1);
    SPI_CS_1;
    while(W25Q_ReadStatus()&0x01);    // 等待擦除完成
}
```

程序清单：擦除整片芯片函数

```
/*****************************************************************************
 * 描  述 : 擦除整片芯片
 * 入  参 : 无
 * 返回值 : 无
```

```
5.  备注：不同型号的芯片时间不一样
6.  ******************************************************************************/
7.  void W25Q_ChipErase(void)
8.  {
9.      while(W25Q_ReadStatus()&0x01);      // 判断是否忙
10.     WriteEnable();                      // 写允许
11.     SPI_CS_0;
12.     SPI_SendByte(W25X_C_Erase);        // 整片擦除命令
13.     SPI_CS_1;                           // 从 CS=1 时开始执行擦除
14.     while(W25Q_ReadStatus()&0x01);      // 等待擦除完成
15. }
```

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

代码清单：主函数

```
1.  int main()
2.  {
3.      uint16  Temp;
4.      uint8   Temp1,Temp2;
5.
6.  /////////////////////////////////////////////
7.  //注意: STC15W4K32S4 系列的芯片,上电后所有与 PWM 相关的 IO 口均为
8.  //       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
9.  //相关 IO: P0.6/P0.7/P1.6/P1.7/P2.1/P2.2
10. //         P2.3/P2.7/P3.7/P4.2/P4.4/P4.5
11. /////////////////////////////////////////////
12.     P2M1 &= 0xE1;   P2M0 &= 0xE1;                   //设置 P2.1~P2.4 为准双向口
13.     P3M1 &= 0xFC;   P3M0 &= 0xFC;                   //设置 P3.0~P3.1 为准双向口
14.
15.     SPI_CS_1;                                       //SPI 使能引脚初始化
16.     Init_SPI();                                       //初始化 SPI
17.     Uart1_Init();                                   //串口 1 初始化
18.     EA = 1;                                         //使能总中断
19.   delay_ms(10);                                    //初始化后延时
20.
21.     Temp=W25Q_ReadID();                             //读取外扩存储器芯片 ID
22.     Temp1=(uint8)(Temp/256);                  //将读取的存储器芯片 ID 高字节存放到 Temp1 中
23.     Temp2=(uint8)Temp;                        //将读取的存储器芯片 ID 低字节存放到 Temp2 中
24.
25.   while (1)
26.     {
27.         if(W_ID)                                    //读存储器芯片 ID 模式
28.         {
```

```
29.                 W_ID=0;                                 //ID 标志变量清零,发送一次
30.                 SendDataByUart1(Temp1);                 //串口 1 发送芯片 ID 高字节
31.                 SendDataByUart1(Temp2);                 //串口 1 发送芯片 ID 低字节
32.             }
33.         if(WriteFLAG)                           //写模式
34.             {
35.                 WriteFLAG=0;                            //写标志变量清零,发送一次
36.                 W25Q_Write_N(scan,0x00000010,1);    //向 FLASH 地址 0x00000010 中写入单字
    节数据 0x33
37.             SendDataByUart1(0x33);              //串口 1 发送数据 0x33 表示写操作完成
38.             }
39.             if(ReadFLAG)                            //读模式
40.             {
41.                 ReadFLAG=0;                             //读标志变量清零,发送一次
42.                 W25Q_Read_N(buffer,0x00000010,1);    //从 FLASH 地址 0x00000010 读取单字
    节数据并存入到 buffer 数组中
43.             SendStringByUart1_n(buffer,1);          //串口 1 发送读取的字符
44.             }
45.             if(ClearFLAG)                           //扇区擦除模式
46.             {
47.                 ClearFLAG=0;                            //清除标志变量清零,发送一次
48.                 W25Q_SectorErase(0x00000010);           //擦除一个扇区（0x00000010 在的扇区）
49.             SendDataByUart1(0x00);                  //串口 1 发送数据 0x00 表示擦除完成
50.             }
51.     }
52. }
```

4.3.4. 硬件连接

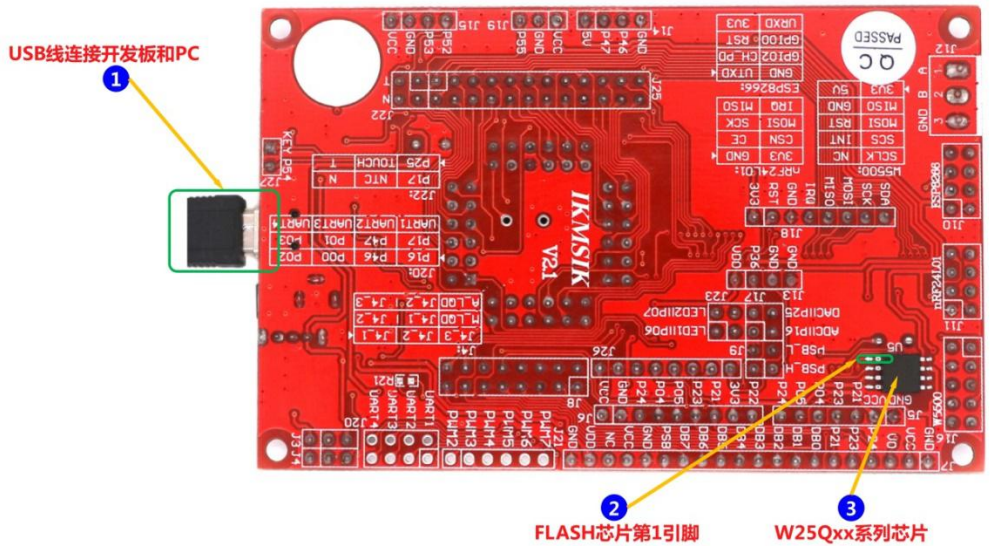


图 8：开发板连接图

4.3.5. 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 2-15-3：外接 FLASH 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“…\W25Qxx\project”目录下的工程“W25Qxx.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“W25Qxx.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. 打开串口调试助手选择正确的串口号，波特率设置为 9600，数据位为 8、停止位为 1，选择 HEX 显示。
7. 实验现象及步骤操作如下：
   1）在发送区写 5，则在接收窗口可看到读取的外部 FLASH 芯片的 ID： EF 17。
   2）在发送区写 3，则在接收窗口可看到读取的指定存储单元的数值： FF 。
   3）在发送区写 4，则在接收窗口可看到表示擦除操作完成的命令： 00 。
   4）在发送区写 3，则在接收窗口可看到读取的指定存储单元的数值： FF 。
   5）在发送区写 2，则在接收窗口可看到表示写操作完成的命令： 33 。
   6）在发送区写 3，则在接收窗口可看到读取的指定存储单元的数值： 33 。
   7）在发送区写 4，则在接收窗口可看到表示擦除操作完成的命令： 00 。
   8）在发送区写 3，则在接收窗口可看到读取的指定存储单元的数值： FF 。

4.4. 外接 FLASH 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 2-15-3：外接 FLASH 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 2-15-4：外接 FLASH 存储器读写多字节实验（硬件 SPI）”。

4.4.1. 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 2-15-3：外接 FLASH 存储器读写单字节实验（硬件 SPI）”部分。

4.4.2. 编写代码

首先，在 w25q128.c 文件中编写硬件 SPI 方式的读写字节函数和对 FLASH 存储器的基本操作函数。请参考“实验 2-15-3：外接 FLASH 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 FLASH 的多字节读、写及擦除等操作。

代码清单：主函数

```
1.  int main()
2.  {
3.      uint16  Temp;
4.      uint8   Temp1,Temp2;
5.
6.  ////////////////////////////////////////////////
7.  //注意: STC15W4K32S4 系列的芯片,上电后所有与 PWM 相关的 IO 口均为
8.  //       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
9.  //相关 IO: P0.6/P0.7/P1.6/P1.7/P2.1/P2.2
10. //         P2.3/P2.7/P3.7/P4.2/P4.4/P4.5
11. ////////////////////////////////////////////////
12.     P2M1 &= 0xE1;   P2M0 &= 0xE1;                   //设置 P2.1~P2.4 为准双向口
13.     P3M1 &= 0xFC;   P3M0 &= 0xFC;                   //设置 P3.0~P3.1 为准双向口
14.
15.     SPI_CS_1;                                       //SPI 使能引脚初始化
16.     Init_SPI();                                      //初始化 SPI
17.     Uart1_Init();                                   //串口 1 初始化
18.     EA = 1;                                         //使能总中断
19.   delay_ms(10);                                    //初始化后延时
20.
21.     Temp=W25Q_ReadID();                             //读取外扩存储器芯片 ID
22.     Temp1=(uint8)(Temp/256);            //将读取的存储器芯片 ID 高字节存放到 Temp1 中
23.     Temp2=(uint8)Temp;                  //将读取的存储器芯片 ID 低字节存放到 Temp2 中
24.
25.   while (1)
26.     {
27.         if(W_ID)                                //读存储器芯片 ID 模式
28.         {
29.               W_ID=0;                           //ID 标志变量清零,发送一次
30.               SendDataByUart1(Temp1);           //串口 1 发送芯片 ID 高字节
31.               SendDataByUart1(Temp2);           //串口 1 发送芯片 ID 低字节
32.          }
33.      if(WriteFLAG)                      //写模式
34.          {
35.               WriteFLAG=0;                      //写标志变量清零,发送一次
36.               W25Q_SectorErase(0x00000000);     //擦除一个扇区(0x00000000 在的扇区)
37.               W25Q_Write_N(scan,0x00000000,10); //向 FLASH 地址 0x00000000 中写入
    scan 数组中 10 个字节数据
38.         SendDataByUart1(0x33);            //串口 1 发送数据 0x33 表示写操作完成
39.          }
40.          if(ReadFLAG)                       //读模式
41.          {
42.               ReadFLAG=0;                       //读标志变量清零,发送一次
```

```
43.                 W25Q_Read_N(buffer,0x00000000,10);       //从 FLASH 地址 0x00000000 开始读取 10 字节数据并存入到 buffer 数组中
44.           SendStringByUart1_n(buffer,10);       //串口 1 发送数组 buffer 中的值（即读取的多字节数据）
45.           }
46.           if(ClearFLAG)                        //扇区擦除模式
47.           {
48.                ClearFLAG=0;                          //清除标志变量清零,发送一次
49.                W25Q_SectorErase(0x00000000);         //擦除一个扇区（0x00000000 在的扇区）
50.               SendDataByUart1(0x00);                 //串口 1 发送数据 0x00 表示擦除完成
51.           }
52.      }
53. }
```

4.4.3. 硬件连接

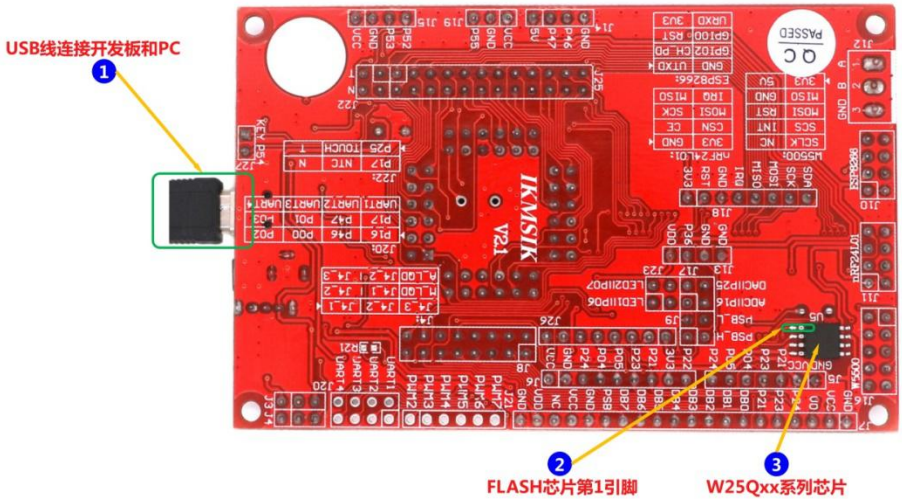


图 9：开发板连接图

4.4.4. 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 2-15-4：外接 FLASH 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“…\W25Qxx\project”目录下的工程“W25Qxx.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“W25Qxx.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. 打开串口调试助手选择正确的串口号，波特率设置为 9600，数据位为 8、停止位为 1，选择 HEX 显示。
7. 实验现象及步骤操作如下：

   1）在发送区写 5，则在接收窗口可看到读取的外部 FLASH 芯片的 ID： EF 17。

   2）在发送区写 3，则在接收窗口可看到读取的指定存储单元的数值： FF FF FF FF FF FF FF FF FF FF 。

   3）在发送区写 2，则在接收窗口可看到表示写操作完成的命令： 33 。

   4）在发送区写 3，则在接收窗口可看到读取的指定存储单元的数值： 11 22 33 44 55 66 77 88 99 AA 。

   5）在发送区写 4，则在接收窗口可看到表示擦除操作完成的命令： 00 。

   6）在发送区写 3，则在接收窗口可看到读取的指定存储单元的数值： FF FF FF FF FF FF FF FF FF FF 。