25.9. 大型综合实战--大学课程设计：硬件 SPI 读写串行 Flash

25.9.1. 实验介绍

1、本实验项目主要目的：
- 掌握串口通讯
- 硬件 SPI 读写串行 Flash

2、掌握 Ai8051U 实验箱原理图中的串口通信电路、硬件 SPI 电路

3、继续熟悉 AIapp-ISP 系统软件中串口助手工具

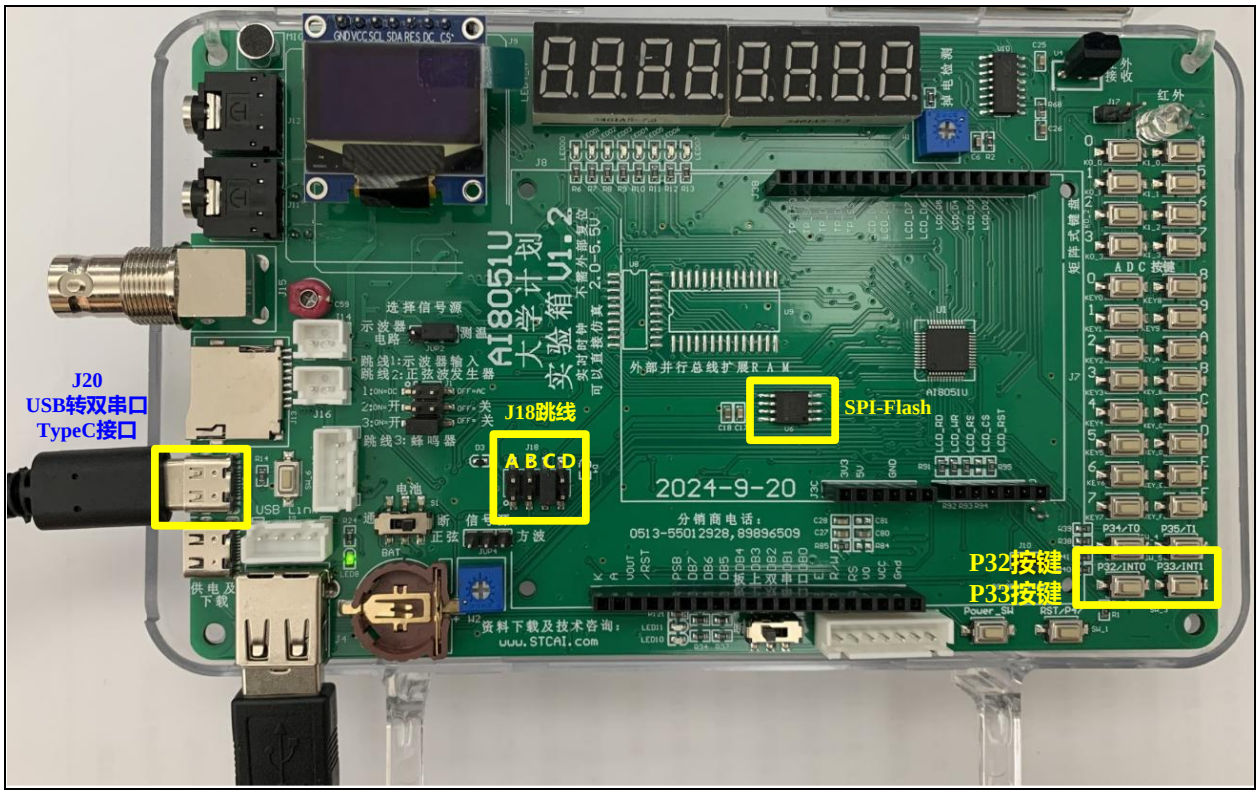
4、继续熟悉如何管理多文件项目

a) 按键去抖动说明：

如按键要判断为正常按下，而不是抖动，需要该按键保持按下持续保持为低状态时间是 20ms ~ 50ms；主循环中有个 1ms 的时间基准，将按键扫描程序作为 1ms 事件，按键状态维持 50ms 不变，即可对按键去抖动，得到稳定的键码。

b) 实验现象和操作说明：

Ai8051U 的实验箱正面图：



1、J20：　USB 转双串口 U2（Ai8H2K12U）TypeC 接口。本实验需要用 TypeA（连接电脑）-TypeC（连接实验箱 J20）线相连。（详见 Ai8051U 实验箱中 J20-U2 原理图）

2、J18 跳线：本实验需要将实验箱上“J18 跳线 C”的跳线连上。
- J18.C：使主控芯片 UART1 的 P3.7/TxD_2 与双串口 U2（Ai8H2K12U）联通。

详见 Ai8051U 实验箱中 J18 跳线原理图

3、SPI-Flash：详见 Ai8051U 实验箱中 U6 原理图

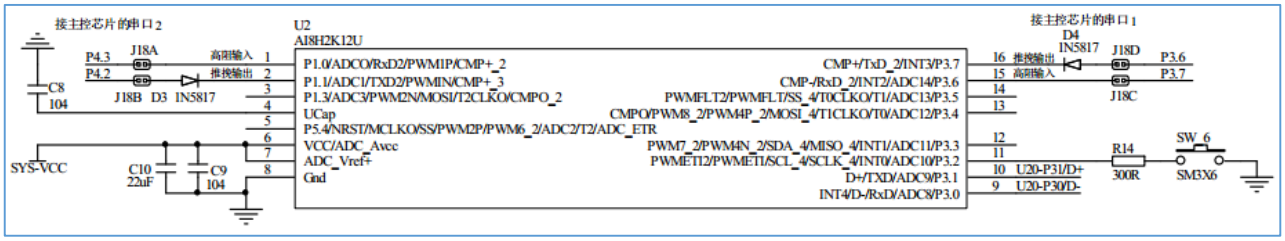
c) Keil 环境下多文件项目管理说明

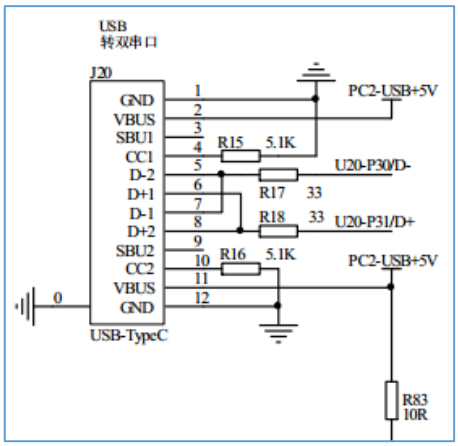
当项目的功能比较复杂时，就需要在 Keil 中建立多文件项目，以方便分工合作、代码复用、模块化管理、增强可读性和可维护性。

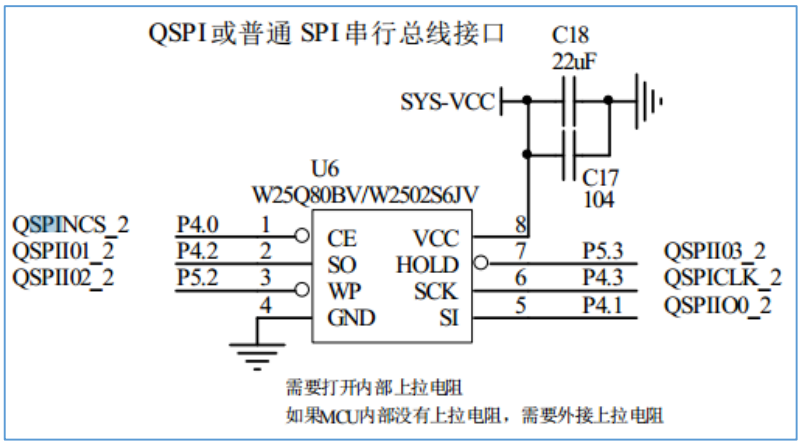
- 比较好的建议是将项目功能模块化，不同模块的实现代码放在不同的.c 文件中。
- 一般建议是一个模块对应一个.c 程序文件和一个.h 头文件
- 模块的初始化函数以及相关的数据处理函数都在.c 文件中实现
- 与模块相关的全局变量也必须在.c 文件中进行定义，一定不能在.h 文件中定义变量
- 如果有其他模块需要使用本模块定义的变量或函数，则这些函数和变量都需要在.h 文件中声明。
- 特别提醒：在.h 文件中声明外部变量必须使用 **extern** 关键字，否则就变成变量定义了，这样会出现变量重复定义的错误
- 为防止头文件被多次包含而产生错误或者警告，在头文件中使用类似如下的条件编译组合语句，可避免在同一个.c 文件中对同一个.h 头文件进行多次包含：

```
#ifndef XXXX
#define XXXX
...
#endif
```

25.9.2.　原理图







25.9.3. 实验程序代码

a) main.c -- 程序主函数

```
//main.c
//程序主函数
//烧录程序到实验箱后，短接 J18C 跳线
//USB 线连接电脑与实验箱 J20 接口，串口助手打开 CDC1 对应的 COM 口。
//通过按键触发硬件 SPI 对串行 Flash 做扇区擦除、写入、读出的操作。
//P32 按键读取串行 Flash ID，以及 Flash 指定地址数据。
//P33 按键短按擦除指定地址 Flash 扇区，并写入计数值，然后读出，通过串口打印操作结果。
//P33 按键长按擦除指定地址 Flash 扇区，重置计数值。

#include "config.h"       //头文件中已包含 ai8051u.h 以及其他头文件

void main()
{
    SYS_Init();          //系统初始化

    while (1)
    {
        if (f1ms)              //判断 1ms 标志位
        {
            f1ms = 0;      //清除 1ms 标志，并处理 1ms 事件
            ScanKey();     //按键扫描
            KeyEvents();   //按键处理事件
        }

    }
}

void SYS_Init(void)      //系统初始化函数
{
    EAXFR = 1;          //使能访问扩展 XFR
    WTST = 0x00;        //设置最快速度访问程序代码
    CKCON = 0x00;       //设置最快速度访问内部 XDATA

    P0M0 = 0x00; P0M1 = 0x00;       //初始化 P0 口为准双向口模式
    P1M0 = 0x00; P1M1 = 0x00;       //初始化 P1 口为准双向口模式
    P2M0 = 0x00; P2M1 = 0x00;       //初始化 P2 口为准双向口模式
    P3M0 = 0x00; P3M1 = 0x00;       //初始化 P3 口为准双向口模式
    P4M0 = 0x00; P4M1 = 0x00;       //初始化 P4 口为准双向口模式
    P5M0 = 0x00; P5M1 = 0x00;       //初始化 P5 口为准双向口模式
    P6M0 = 0x00; P6M1 = 0x00;       //初始化 P6 口为准双向口模式
    P7M0 = 0x00; P7M1 = 0x00;       //初始化 P7 口为准双向口模式
```

```
    P3PU |= 0x0c;       //使能 P3.2,P3.3 内部上拉

    SPI_Init();         //SPI 接口初始化
    Uart1_Init();       //UART 接口初始化
    Timer0_Init();      //定时器 0 初始化,用于产生 1ms 的时间基准

    EA = 1;             //总中断允许位打开
}
```

b) spi.c -- 初始化 SPI 接口、串行 Flash 操作函数

```
//spi.c
//初始化 SPI 接口、串行 Flash 操作函数

#include "config.h"       //头文件中已包含 ai8051u.h 以及其他头文件

u8   B_FlashOK;           //Flash 状态
u8   PM25LV040_ID, PM25LV040_ID1, PM25LV040_ID2;

void SPI_Init()           //SPI 初始化函数
{
    SPCTL = (SPCTL & ~3) | 1;       //SPI 时钟频率选择, 0: 4T, 1: 8T, 2: 16T, 3: 2T
    SSIG = 1;       //1: 忽略 SS 脚，由 MSTR 位决定主机还是从机   0: SS 脚用于决定主机还是从机。
    SPEN = 1;       //1: 允许 SPI，0: 禁止 SPI，所有 SPI 管脚均为普通 IO
    DORD = 0;       //1: LSB 先发，0: MSB 先发
    MSTR = 1;       //1: 设为主机，0: 设为从机
    CPOL = 0;       //1: 空闲时 SCLK 为高电平，0: 空闲时 SCLK 为低电平
    CPHA = 0;       //1: 数据在 SCLK 前沿驱动,后沿采样.   0: 数据在 SCLK 前沿采样,后沿驱动.
    SPI_S1 = 1;     //选择 SPI 脚位通道
    SPI_S0 = 0;     //00: P1.4 P1.5 P1.6 P1.7, 01: P2.4 P2.5 P2.6 P2.7,
                    //10: P4.0 P4.1 P4.2 P4.3, 11: P3.5 P3.4 P3.3 P3.2
}

void SPI_WriteByte(u8 out)
{
    SPDAT = out;            //写入需要发送的数据
    while(SPIF == 0);       //等待传输完成
    SPIF = 1;               //清 SPIF 标志
    WCOL = 1;               //清 WCOL 标志
}

/*********************************************************************/
u8 SPI_ReadByte(void)
{
```

```
    SPDAT = 0xff;       //MOSI 发送 0xFF(高电平)产生 CLK，同时读取 MISO 数据
    while(SPIF == 0);   //等待传输完成
    SPIF = 1;           //清 SPIF 标志
    WCOL = 1;           //清 WCOL 标志
    return (SPDAT);     //返回 MISO 读取数据
}

/*************************************************
检测 Flash 是否准备就绪
入口参数: 无
出口参数:
    0 : 没有检测到正确的 Flash
    1 : Flash 准备就绪
*************************************************/
void FlashCheckID(void)
{
    SPI_CE_Low();                   //SPI 片选使能
    SPI_WriteByte(SFC_RDID);        //发送读取 ID 命令
    SPI_WriteByte(0x00);            //空读 3 个字节
    SPI_WriteByte(0x00);
    SPI_WriteByte(0x00);
    PM25LV040_ID1 = SPI_ReadByte();  //读取制造商 ID1
    PM25LV040_ID = SPI_ReadByte();   //读取设备 ID
    PM25LV040_ID2 = SPI_ReadByte();  //读取制造商 ID2
    SPI_CE_High();                  //SPI 片选禁能

    printf("制造商 ID1 = 0x%02X",PM25LV040_ID1);
    printf("\r\n       ID2 = 0x%02X",PM25LV040_ID2);
    printf("\r\n  设备 ID = 0x%02X\r\n",PM25LV040_ID);

    if((PM25LV040_ID1 == 0x9d) && (PM25LV040_ID2 == 0x7f))      B_FlashOK = 1;
                                                                //检测是否为 PM25LVxx 系列的 Flash
    else if(PM25LV040_ID == 0x12)       B_FlashOK = 2;    //检测是否为 W25X4x 系列的 Flash
    else if(PM25LV040_ID == 0x13)       B_FlashOK = 3;    //检测是否为 W25X8x 系列的 Flash
    else if(PM25LV040_ID == 0x17)       B_FlashOK = 4;    //检测是否为 W25X128 系列的 Flash
    else                                B_FlashOK = 0;    //没有检测到指定型号的 Flash

    if(!B_FlashOK)    printf("未检测到指定型号的 Flash!\r\n");
    else
    {
        if(B_FlashOK == 1)
        {
            printf("检测到 PM25LV040!\r\n");
        }
        else if(B_FlashOK == 2)
        {
```

```
            printf("检测到 W25X40CL!\r\n");
        }
        else if(B_FlashOK == 3)
        {
            printf("检测到 W25Q80BV!\r\n");
        }
        else if(B_FlashOK == 4)
        {
            printf("检测到 W25Q128FV!\r\n");
        }
    }
}

/*************************************************
检测 Flash 的忙状态
入口参数: 无
出口参数:
    0 : Flash 处于空闲状态
    1 : Flash 处于忙状态
*************************************************/
u8 CheckFlashBusy(void)
{
    u8    dat;

    SPI_CE_Low();           //SPI 片选使能
    SPI_WriteByte(SFC_RDSR);      //发送读取状态命令
    dat = SPI_ReadByte();   //读取状态
    SPI_CE_High();          //SPI 片选禁能

    return (dat);           //状态值的 Bit0 即为忙标志
}

/*************************************************
使能 Flash 写命令
入口参数: 无
出口参数: 无
*************************************************/
void FlashWriteEnable(void)
{
    while(CheckFlashBusy() > 0);      //Flash 忙检测
    SPI_CE_Low();                     //SPI 片选使能
    SPI_WriteByte(SFC_WREN);          //发送写使能命令
    SPI_CE_High();                    //SPI 片选禁能
}

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

```
擦除扇区, 一个扇区 4KB
入口参数: 无
出口参数: 无
*************************************************/
void FlashSectorErase(u32 addr)
{
    if(B_FlashOK)                   //判断是否检测到指定型号 Flash
    {
        FlashWriteEnable();         //使能 Flash 写命令
        SPI_CE_Low();               //SPI 片选使能
        if(B_FlashOK == 1)          //判断 Flash 类型，不同型号擦除命令不同
        {
            SPI_WriteByte(SFC_SECTORER1);   //发送扇区擦除命令
        }
        else
        {
            SPI_WriteByte(SFC_SECTORER2);   //发送扇区擦除命令
        }
        SPI_WriteByte(((u8 *)&addr)[1]);            //设置起始地址
        SPI_WriteByte(((u8 *)&addr)[2]);
        SPI_WriteByte(((u8 *)&addr)[3]);
        SPI_CE_High();                              //SPI 片选禁能
    }
}

/*************************************************
从 Flash 中读取数据
入口参数:
    addr   : 地址参数
    buffer : 缓冲从 Flash 中读取的数据
    size   : 数据块大小
出口参数:
    无
*************************************************/
void SPI_Read_Nbytes(u32 addr, u8 *buffer, u16 size)
{
    if(size == 0)       return;         //操作长度为 0 则退出
    if(!B_FlashOK)      return;         //没检测到指定型号 Flash 则退出
    while(CheckFlashBusy() > 0);        //Flash 忙检测

    SPI_CE_Low();                       //SPI 片选使能
    SPI_WriteByte(SFC_READ);            //发送读命令

    SPI_WriteByte(((u8 *)&addr)[1]);        //设置起始地址
    SPI_WriteByte(((u8 *)&addr)[2]);
    SPI_WriteByte(((u8 *)&addr)[3]);
```

```
    do{
        *buffer = SPI_ReadByte();    //读取 1 个字节内容存入缓冲区
        buffer++;
        }while(--size);              //连续读取指定个数内容
    SPI_CE_High();                   //SPI 片选禁能
}

/*********************************************
写数据到 Flash 中
入口参数:
    addr   : 地址参数
    buffer : 缓冲需要写入 Flash 的数据
    size   : 数据块大小
出口参数: 无
*********************************************/
void SPI_Write_Nbytes(u32 addr, u8 *buffer, u8 size)
{
    if(size == 0)        return;        //操作长度为 0 则退出
    if(!B_FlashOK)       return;        //没检测到指定型号 Flash 则退出
    while(CheckFlashBusy() > 0);        //Flash 忙检测

    FlashWriteEnable();                 //使能 Flash 写命令

    SPI_CE_Low();                       //SPI 片选使能
    SPI_WriteByte(SFC_PAGEPROG);        //发送页编程命令
    SPI_WriteByte(((u8 *)&addr)[1]);    //设置起始地址
    SPI_WriteByte(((u8 *)&addr)[2]);
    SPI_WriteByte(((u8 *)&addr)[3]);
    do{
        SPI_WriteByte(*buffer++);       //连续页内写
        addr++;
        if ((addr & 0xff) == 0) break;  //地址跨页则退出
        }while(--size);
    SPI_CE_High();                      //SPI 片选禁能
}
```

c) uart.c -- 初始化 UART 接口

```
//uart.c
//初始化 UART 接口

#include "config.h"          //头文件中已包含 ai8051u.h 以及其他头文件

void Uart1_Init(void)        //115200bps@40.000MHz
```

```
{
    S1_S1 = 0;              //UART1 通道选择, 00: P3.0 P3.1, 01: P3.6 P3.7, 10: P1.6 P1.7, 11: P4.3 P4.4
    S1_S0 = 1;              //UART1 选择  P3.6 P3.7  作为串口收发通道

    SCON = 0x50;            //8 位数据,可变波特率
    AUXR |= 0x40;           //定时器时钟 1T 模式
    AUXR &= 0xFE;           //串口 1 选择定时器 1 为波特率发生器
    TMOD &= 0x0F;           //设置定时器模式
    TL1 = 0xA9;             //设置定时初始值
    TH1 = 0xFF;             //设置定时初始值
    ET1 = 0;                //禁止定时器中断
    TR1 = 1;                //定时器 1 开始计时
}

void Uart1_Send(u8 dat)     //通过串口 1 发送数据
{
    SBUF = dat;             //写入需要发送的数据
    while(!TI);             //判断发送是否完成
    TI = 0;                 //清除发送完成标志
}

char putchar(char c)        //将  printf  函数映射到  UART1  接口
{
    Uart1_Send(c);      //调用串口 1 发送函数（如果要映射到串口 2 的话，这里修改为串口 2 发送函数）
    return c;
}
```

d) key.c -- 按键扫描以及按键事件处理

```
//key.c
//按键扫描以及按键事件处理

#include "config.h"                 //头文件中已包含 ai8051u.h, ai_usb.h 以及其他头文件

bit   Key1_Flag;
bit   Key2_Flag;
bit   Key2_Short_Flag;

bit   Key1_Function;
bit   Key2_Short_Function;
bit   Key2_Long_Function;

u16  Key1_cnt;
u16  Key2_cnt;
```

```
union LongType temp;

void KeyScan(void)
{
    //单纯短按按键
    if(!KEY1)
    {
        if(!Key1_Flag)      //判断按键是否已经有效触发，如果已经处于触发状态则不再检测，避免重复触发
        {
            Key1_cnt++;                 //按键计数器加 1
            if(Key1_cnt >= 50)          //50ms 防抖
            {
                Key1_Flag = 1;          //设置按键状态，防止重复触发
                Key1_Function = 1;      //设置按键操作标志
            }
        }
    }
    else
    {
        Key1_cnt = 0;       //按键计数器清零
        Key1_Flag = 0;      //按键释放
    }

    //长按短按按键
    if(!KEY2)
    {
        if(!Key2_Flag)      //判断按键是否已经有效触发，如果已经处于触发状态则不再检测，避免重复触发
        {
            Key2_cnt++;  //按键计数器加 1
            if(Key2_cnt >= 1000)            //长按 1s
            {
                Key2_Short_Flag = 0;        //清除短按标志
                Key2_Flag = 1;              //设置按键状态，防止重复触发
                Key2_Long_Function = 1;     //设置长按操作标志
            }
            else if(Key2_cnt >= 50)         //50ms 防抖
            {
                Key2_Short_Flag = 1;        //设置短按标志
            }
        }
    }
    else
    {
        if(Key2_Short_Flag)             //判断是否短按
        {
```

```
            Key2_Short_Flag = 0;        //清除短按标志
            Key2_Short_Function = 1;    //设置短按操作标志
        }
        Key2_cnt = 0;       //按键计数器清零
        Key2_Flag = 0;      //按键释放
    }
}

void KeyEvents(void)
{
    u8 buf[4];

    if(Key1_Function)           //判断 KEY1 是否按下
    {
        Key1_Function = 0;      //清按键标志位
        FlashCheckID();         //检测 Flash 型号

        SPI_Read_Nbytes(FLASH_ADDR,temp.cdata,4);     //读取 4 字节内容
        printf("读取内容：0x%08lx\r\n",temp.ldata);          //打印读取内容，printf 打印 32 位数据需要加"l"
        temp.ldata++;               //数值加 1
        //temp.ldata = 0x12345678;  //设置写入内容
    }

    if(Key2_Short_Function)         //判断 KEY2 是否短按
    {
        Key2_Short_Function = 0;    //清按键标志位

        FlashSectorErase(FLASH_ADDR);     //擦除 Flash
        printf("写入内容：0x%08lx\r\n",temp.ldata);          //打印写入内容，printf 打印 32 位数据需要加"l"
        SPI_Write_Nbytes(FLASH_ADDR,temp.cdata,4);     //写 4 个字节内容

        SPI_Read_Nbytes(FLASH_ADDR,buf,4);             //读取 4 字节内容
        printf("读取内容：0x%02x%02x%02x%02x\r\n",buf[0],buf[1],buf[2],buf[3]);       //打印读取内容

        temp.ldata++;               //数值加 1
    }

    if(Key2_Long_Function)          //判断 KEY2 是否长按
    {
        Key2_Long_Function = 0;     //清按键标志位
        printf("擦除\xfd Flash\r\n");
        FlashSectorErase(FLASH_ADDR);     //擦除 Flash
        temp.ldata = 0;             //重置写入内容
    }
}
```

e) timer.c -- 定时器初始化以及定时器中断服务程序

```
//timer.c
//定时器初始化以及定时器中断服务程序
//定时器 0:1ms 定时，产生 1ms 的标志位

#include "config.h"                 //头文件中已包含 ai8051u.h, ai_usb.h 以及其他头文件

bit    f1ms;                        //1ms 标志位

void Timer0_Init(void)              //1 毫秒@40.000MHz
{
    AUXR |= 0x80;                   //定时器 0 时钟 1T 模式
    TMOD &= 0xF0;                   //设置定时器 0 为模式 0
    TL0 = 0xC0;                     //设置定时器 0 初始值
    TH0 = 0x63;                     //设置定时器 0 初始值
    TF0 = 0;                        //清除 TF0 标志
    TR0 = 1;                        //定时器 0 开始计时
    ET0 = 1;                        //使能定时器 0 中断

    f1ms = 0;                       //初始化 1ms 标志位
}

void Timer0_Isr(void) interrupt 1   //定时器 0 中断服务程序
{
    f1ms = 1;                       //设置 1ms 标志位
}
```

f) config.h -- 项目配置的头文件

```
#ifndef __CONFIG_H__                //防止头文件被重复包含
#define __CONFIG_H__

#define HIRC            40000000UL
#define FOSC            40000000UL
#define SYSCLK          FOSC
#define MAIN_Fosc       FOSC

#include <ai8051u.h>                //包含外部头文件
#include <stdio.h>
#include <intrins.h>

#include "def.h"                    //包含项目头文件
#include "timer.h"
```

```
#include "key.h"
#include "spi.h"
#include "uart.h"

void SYS_Init(void);                    //函数声明

#endif
```

g) spi.h -- 项目配置文件的头文件

```
#ifndef __SPI_H__               //防止头文件被重复包含
#define __SPI_H__

sbit    SPI_CE   = P4^0;        //PIN1
sbit    SPI_SO   = P4^2;        //PIN2
sbit    SPI_SI   = P4^1;        //PIN5
sbit    SPI_SCK = P4^3;         //PIN6

#define  SFC_WREN               0x06     //串行 Flash 命令集
#define  SFC_WRDI               0x04
#define  SFC_RDSR               0x05
#define  SFC_WRSR               0x01
#define  SFC_READ               0x03
#define  SFC_FASTREAD           0x0B
#define  SFC_RDID               0xAB
#define  SFC_PAGEPROG           0x02
#define  SFC_RDCR               0xA1
#define  SFC_WRCR               0xF1
#define  SFC_SECTORER1          0xD7     //PM25LV040  扇区擦除指令
#define  SFC_SECTORER2          0x20     //W25Xxx  扇区擦除指令
#define  SFC_BLOCKER            0xD8
#define  SFC_CHIPER             0xC7

#define  FLASH_ADDR             0x00

#define  SPI_CE_High()          SPI_CE = 1    // set CE high
#define  SPI_CE_Low()           SPI_CE = 0    // clear CE low

void SPI_Init();                        //函数声明
void FlashCheckID(void);                //函数声明
void FlashSectorErase(u32 addr);
void SPI_Read_Nbytes( u32 addr, u8 *buffer, u16 size);
void SPI_Write_Nbytes(u32 addr, u8 *buffer,    u8 size);

#endif
```

h) uart.h -- 项目配置文件的头文件

```
#ifndef __UART_H__          //防止头文件被重复包含
#define __UART_H__

void Uart1_Init();          //函数声明
void Uart1_Send(u8 dat);    //函数声明

#endif
```

i) key.h -- 项目配置文件的头文件

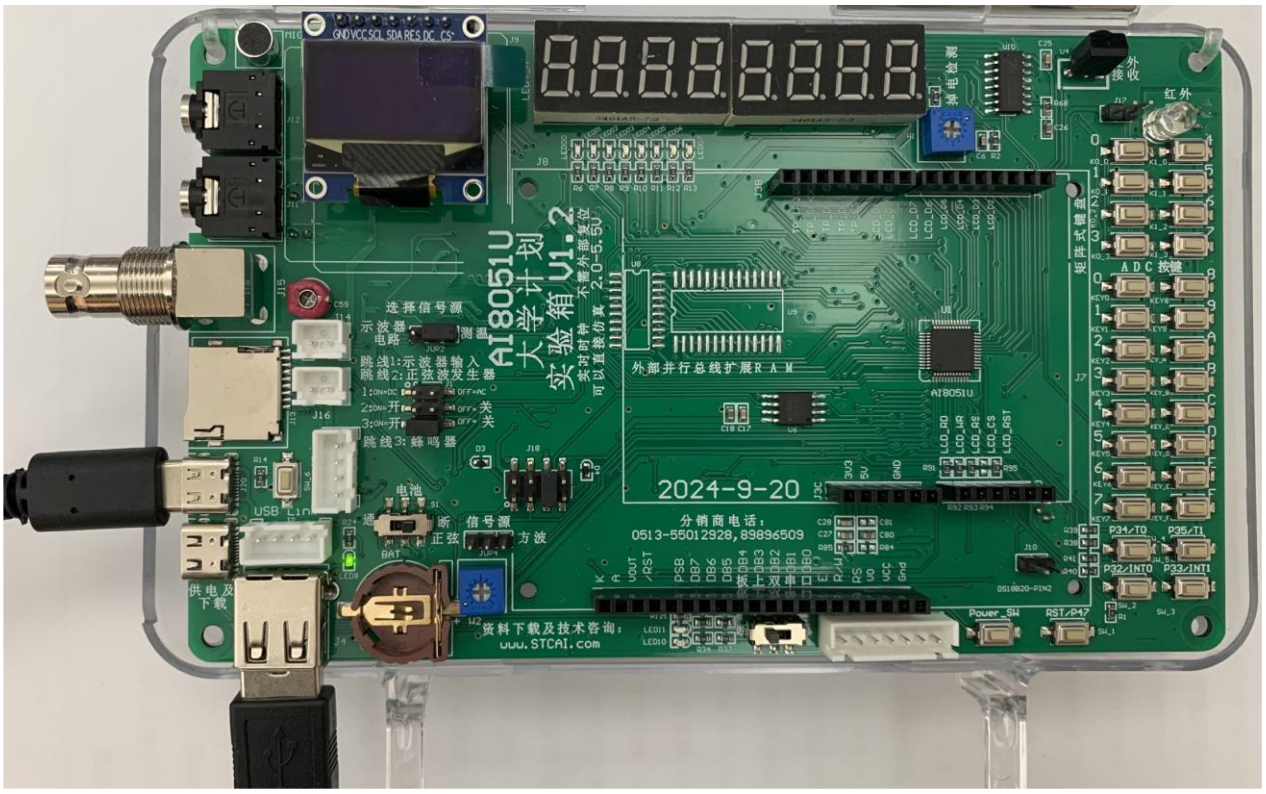
```
#ifndef __KEY_H__           //防止头文件被重复包含
#define __KEY_H__

sbit KEY1 = P3^2;
sbit KEY2 = P3^3;

union LongType
{
    u32 ldata;
    u8 cdata[4];
};

extern union LongType temp;

void KeyScan ();            //函数声明
void KeyEvents();           //函数声明

#endif
```

j) timer.h -- 项目配置文件的头文件

```
#ifndef __TIMER_H__         //防止头文件被重复包含
#define __TIMER_H__

void Timer0_Init(void);     //函数声明

extern bit f1ms;            //外部变量声明

#endif
```

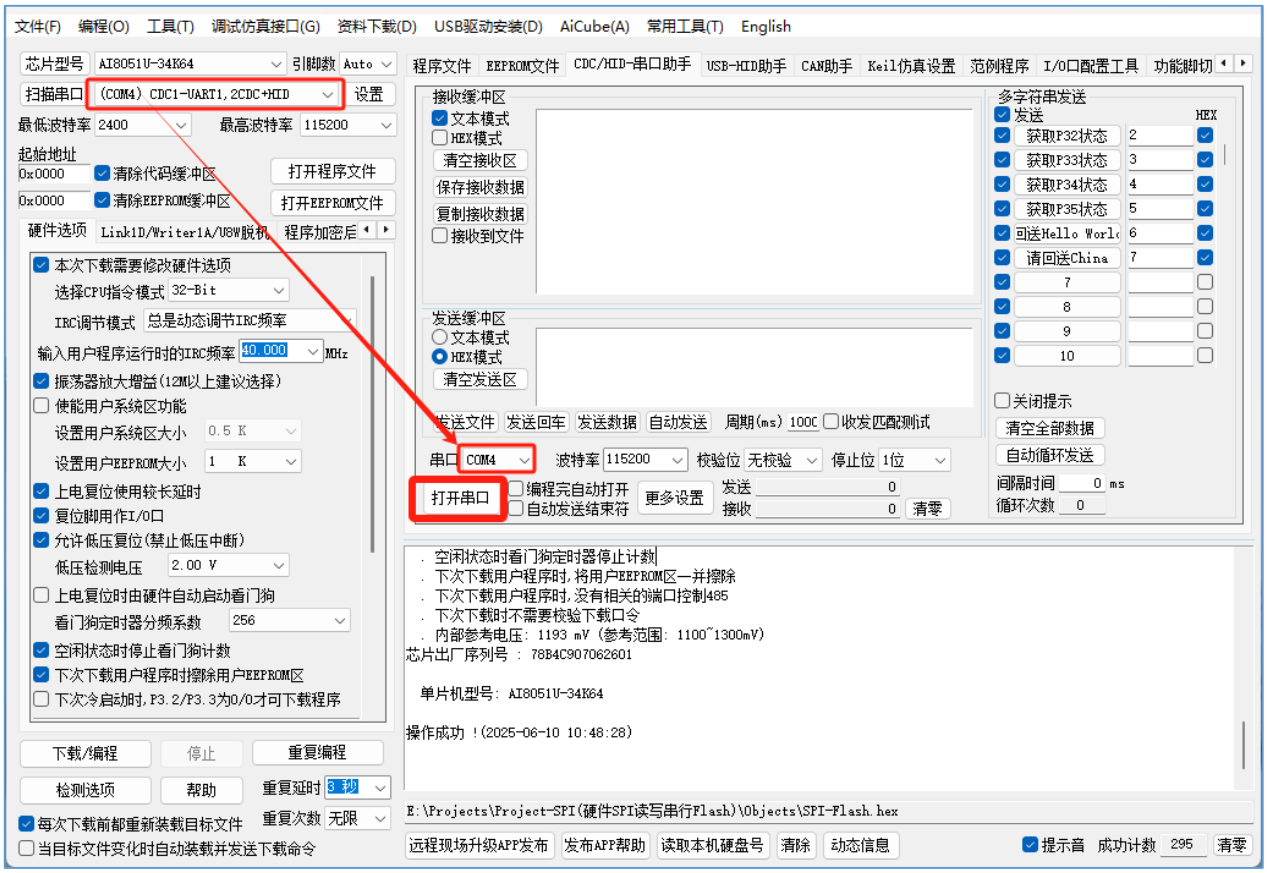
25.9.4.　程序下载

录入代码，保存，编译。用 TypeA-TypeC 线与 Ai8051U 实验箱的 J20 接口，短接 J18C 跳线，如下图：



运行 AIapp-ISP 软件系统，按步骤打开并下载程序“SPI-Flash.hex”，如下图：

在 AIapp-ISP 软件系统界面，点击“CDC/HID-串口助手”选项卡，串口选择 CDC1 对应的 COM 口，找到“打开串口”按钮，如下图：



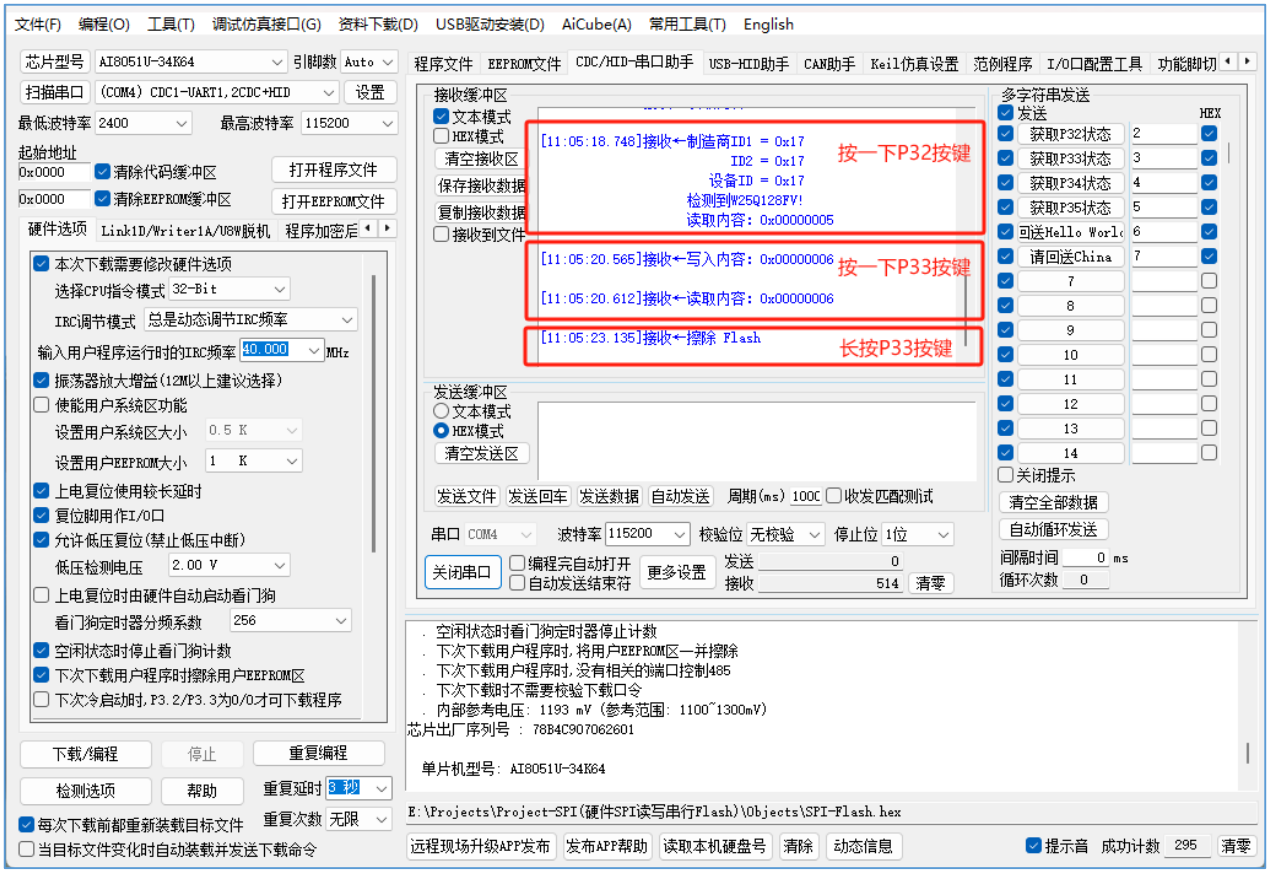
一切准备就绪！

25.9.5. 观察实验现象，验证实验效果

在上述 AIapp-ISP 软件系统界面：

- 点击“打开串口”按钮
- 按一下 P32 按键：读取串行 Flash ID，以及 Flash 指定地址数据。
- 按一下 P33 按键（短按）：擦除指定地址 Flash 扇区，并写入计数值，然后读出，通过串口打印操作结果。
- 长按 P33 按键：擦除指定地址 Flash 扇区，重置计数值。

观察接收缓冲区，如下图：



完成实验验证。