南通大学信息科学技术学院

嵌入式技术课程设计

报 告 书

设计题目	SPI + UART 聊天器
班级	计 171、172
姓名	刘乾、丁雨聪
学号	1714021057、1713021057
指导老师	李跃华
日期	2020.01.03

Contents

软硬件平台

功能概述

将两块实验板通过 SPI 通信，两块实验板通过 USB 连接至 PC 机，一方 PC 软件输入内容，一方 PC 软件能够显示。

软件开发平台

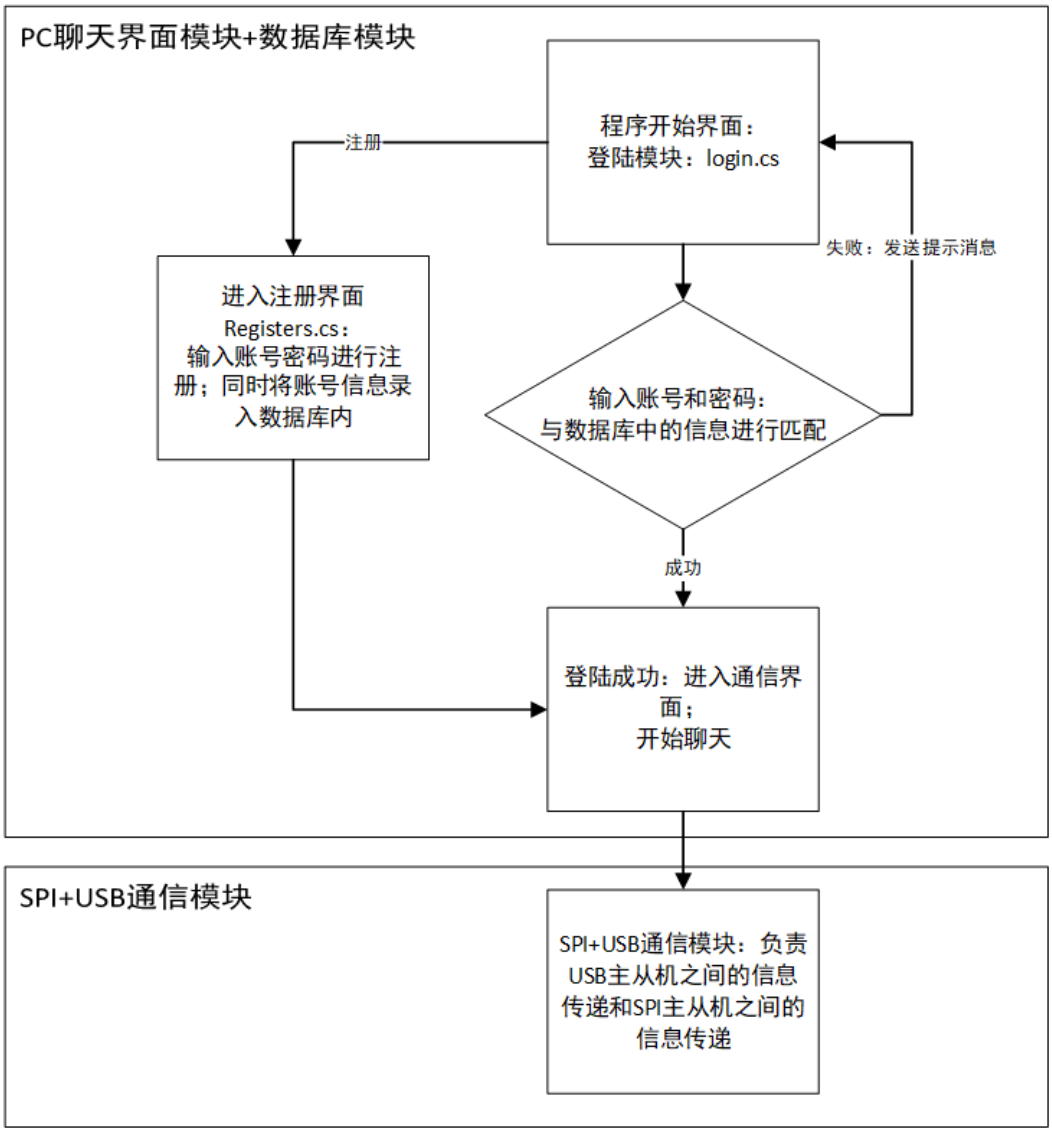
Visual Studio 2017/2019， Kinetis Design Studio 3.0.0 IDE

硬件开发平台

Freescale KL25 ARM Cortex-M0+KL 系列微控制器

总体设计

总体流程



引脚接法

（1）总共两块 MCU，分别取名为 KL251，KL252。

（2）将 KL251、KL252 通过 USB 线将实验板的 MicroUSB 口（标注 KL25 字样）与电脑相连，电脑起供电和传输作用。

（3）KL251 的 A14 口连接 KL252 的 B10 口，KL251 的 A15 口连接 K125T2 的 B11 口，KL251 的 A16 口，接 KL252 的 B16 口，KL251 的 A17 口接 KL252 的 B17 口。

（4）KL252 的 A14 口连接 KL251 的 B10 口，KL252 的 A15 口连接 KL251 的 B11 口，KL252 的 A16 口，接 KL251 的 B16 口，KL252 的 A17 口接 KL251 的 B17 口。

软件结构图

MCU 端软件

根据需求分析，对需要完成的功能进行分析：

（1） 初始化 USB，使 MCU 拥有传输数据的能力。

（2） 初始化 SPI0 和 SPI1，分别作为主机和从机。

（3） 在 USB 中断中发送数据。

（4） 主函数中当 SPI 输出数据时，SPI0 就会不断的发送时钟信号

（5） 然后从机在在 SPI 中断中接受数据。

PC 端软件

根据需求分析，对需要完成的功能进行设计。

（1） 用户打开聊天软件，在主界面选择登录还是注册，如果是登录，进入步骤（3）否则进入步骤（2）。

（2） 注册界面，用户需要填写用户名和密码，当与数据库已存在的用户名进行匹配，如果无重复，那么就注册成功，用户名和密码存入数据库中。否则提示失败。

（3） 在主登录界面，填写已经注册过的用户名和密码，用户可根据实际情况选择是否显示密码。当用户名和密码与数据库中已存在的数据匹配成功时，进入系统，否则提示错误。用户可以选择是否记录用户名，以便下次登录更加方便。

（4） 进入系统后。用户使用 USB 连接实验板和 PC 机成功之后，在软件左侧会显示系统 USB 设备信息和 USB 设备选择供用户选择。在发送框输入内容点击发送，在接收框可以得到数据，，用户可以选择，清空聊天窗口。系统左下角会显示用户信息，根据数据库信息会显示用户的头像、姓名。接收窗口中，在显示字符串前一行会显示系统当前时间，下一行会显示用户名展示用户发送的标识，方便用户观察聊天记录。

SPI + USB 双向通信设计

概述

本工程为测试 SPI 通信，从整体设计来看，我们需要完成以下任务

MCU：（刘乾完成）

（1） 编写 USB 中断服务例程

（2） 编写 SPI（8 线、4 线）中断服务例程

（3） 修改主函数

PC：（丁雨聪完成）

（1） PC 软件界面功能、样式设计

（2） 用户数据库建立与 PC 端通信

（3） 接收框接受信息格式、显示格式

（4） 图片收发功能

MCU 方软件设计

（1） 编写 USB 终端服务例程

USB 主机：主要负责发送： 接收从机数据命令：`cmdINTESTDATA` 发送数据给从机命令：`cmdOUTTESTDATA` 给 USB 从机来控制主从机之间的数据传输；

USB 从机：主要负责接收主机发送过来的命令，接收主机发送的数据或者向主机发送数据；

```
1.  public bool sendData(byte[] sendData, int count)
2.  {
3.      ErrorCode ec = ErrorCode.None;
4.      int bytesWritten;
5.      try
6.      {
7.          ec = writer.Write(sendData, 0, count, 100, out bytesWritten);
8.          if (ec != ErrorCode.None)
9.              throw new Exception(ec.ToString());
10.         return true;
11.     }
12.     catch (Exception ex)
13.     {
14.         Console.WriteLine("Error:" + ex.Message);
15.         return false;
16.     }
17.
18. }
19.
20.
21. public bool sendData(byte sendData)
22. {
23.     ErrorCode ec = ErrorCode.None;
24.     int bytesWritten;
25.     try
26.     {
```

```
        ec = writer.Write(sendData, 100, out bytesWritten);
        if (ec != ErrorCode.None)
            throw new Exception(ec.ToString());
        return true;
    }
    catch (Exception ex)
    {
        Console.WriteLine("Error:" + ex.Message);
        return false;
    }
}
```

（2） USB 接收数据

```
public string recvString()
        {
            byte[] readBuffer = new byte[64];
            int bytesRead;
            reader.Read(readBuffer, 100, out bytesRead);
            return Encoding.Default.GetString(readBuffer, 0, bytesRead);
        }

        public byte[] recvByte()
        {
            byte[] readBuffer = new byte[64];
            int bytesRead;
            reader.Read(readBuffer, 100, out bytesRead);

            if (bytesRead > 0)
            {
                byte[] recvDataBuffer = new byte[bytesRead];
                Array.Copy(readBuffer, recvDataBuffer, bytesRead);
                return recvDataBuffer;
            }
            else
            {
                return null;
            }
        }
```

（3） 编写 SPI 中断服务例程

[1] 主机与从机概念：一个 SPI 系统，由一个主机和一个或多个从机构成，主机启动一个与从机的同步通信，从而完成数据交换。
[2] 主出从入引脚 MOSI 与主入从出引脚 MISP,主出从入引脚是主机输出，从机出入数据线。主入从出引脚是主机输入，从机输出数据线。
[3] SPI 串行时钟引脚：SPI 串行时钟引脚用于控制主机与从机之间的数据传输。
[4] 时钟极性与时钟相位：时钟极性表示时钟信号在空闲时是高电平还是低电平。
[5] 从机选择引脚 SS：一些芯片带有从机选择引脚 SS，也称为片选引脚。

```
void SPI1_IRQHandler(void)
{
    uint_8 redata;
    DISABLE_INTERRUPTS;
    redata=SPI_receive1(SPI_1);    //接收主机发送过来的一个字节数据。
    if(redata!='0')
    {
        uart_send1(UART_2,redata);
    }
```

```
    ENABLE_INTERRUPTS;
}
```

（3）修改主函数

```
while(i<5 && !SPI_S_REG(SPI_baseadd(SPI_0))&SPI_S_SPRF_MASK)
     {                              //通信原理可参考 01_Doc 文本中的硬件说明
          SPI_send1(SPI_0,'0');     //为从机向主机发送数据提供时钟信号；此时主机发送的内容应被从机自动过滤。
          i++;                      //i：计数器，做限时用，一定时间后跳出 while，默认从机未向主机发送数据
      }
```

传输原理：

从主机 CPU 发出启动传输信号开始，将要传送的数据装入 8 位移位寄存器，并同时产生 8 个时钟信号，以此从 SCK 引脚送出，在 SCK 信号的控制下，主机中 8 位移位寄存器中的数据依次从 MOSI 引脚送出，到从机的 MOSI 引脚后送入它的 8 位移位寄存器。

SPI 的时序：

（1）空闲电平低电平，上升沿取数

（2）空闲电平低电平，下降沿取数

（3）空闲电平高电平，下降沿取数

（4）空闲电平高电平，上升沿取数

双向通信的两种方式

八线双向：

这种方式是最简单的，两个 MCU 同时充当主机和从机，发送数据只用到每个板子的主机模块；接收数据只用到每个板子的从机模块

源码：

```
#include "includes.h"   //包含总头文件
//USB 序列号
//注意：修改 USB 序列号时，要将设备名的第一个字节数据长度字段进行修改
uint_8 Serial_String[] = {
        0x10,       //设备名长度字段(包含该字段本身长度)
        0x03,       //设备名索引
        //以下是用户自定义设备名
        'H', 0x00, 'W', 0x00, 'W', 0x00, '_', 0x00, 'U', 0x00, 'S', 0x00, 'B',
        0x00, };

void Delay_ms(uint16_t u16ms) {
    uint32_t u32ctr;
    for (u32ctr = 0; u32ctr < ((48000 / 10) * u16ms); u32ctr++) {
        asm ("NOP");
    }
}

int main(void) {
```

```
19.     uint_32 mRuncount;      //主循环计数器
20. //  uint_8 i;
21.
22.     uint_8 j;
23.     //2. 关总中断
24.     DISABLE_INTERRUPTS;
25.     //3. 初始化外设模块
26.     light_init(RUN_LIGHT_BLUE, LIGHT_ON);        //蓝灯初始化
27.
28.     uart_init(UART_1, 9600);      //串口 1 初始化,波特率 9600
29.
30.     //把 SPI0 初始化为主机,波特率 6000，时钟极性 0，时钟相位 0
31.     SPI_init(SPI_0, 1, 6000, 0, 0);
32.     //把 SPI1 初始化为从机,波特率 6000，时钟极性 0，时钟相位 0
33.     SPI_init(SPI_1, 0, 6000, 0, 0);
34.
35.     //USB 设备初始化
36.     usb_init(Serial_String);
37.     //4. 给有关变量赋初值
38.     mRuncount = 0;              //主循环计数器
39.
40.     g_USBRecvLength = 0;        //USB 接收数据长度初始化
41.     g_USBSendLength = 0;        //USB 发送数据长度初始化
42.     g_USBSendFlag = 0;          //USB 执行发送标志初始化
43.     //5. 使能模块中断
44.     uart_enable_re_int(UART_1);      //uart 接收中断
45.
46.     SPI_enable_re_int(SPI_1);        //从机 SPI_1 的接收中断
47.
48.     for (;;) {
49.         //运行指示灯（RUN_LIGHT）闪烁-----------------------------------------------
50.         mRuncount++;                          //主循环次数计数器+1
51.         if (mRuncount >= RUN_COUNTER_MAX)  //主循环次数计数器大于设定的宏常数
52.         {
53.             mRuncount = 0;                    //主循环次数计数器清零
54.
55.         }
56.
57.         //以下加入用户程序-----------------------------------------------------------
58.         //USB 主机向 USB 设备发送要数据命令
59.         if (g_USBRecv[0] == cmdINTESTDATA && g_USBSendLength != 0) {
60.             g_USBRecv[0] = cmdNULL;
61.             g_USBSend[0] = cmdINTESTDATA;
62.             g_USBSendFlag = 1;
63.         }
64.
65.         //USB 主机发送数据，写入到发送数组中，以便主机要数据时进行发送
66.         else if (g_USBRecv[0] == cmdOUTTESTDATA && g_USBRecvLength != 0) {
67.             //禁止 USB 中断
68.             disable_irq(USB_INTERRUPT_IRQ);
69.
70.             //将接收数据的内容传输到发送缓存数组中进行保存，待发送
71.             for (j = 0; j < g_USBRecvLength; j++) {
72.                 spi_Send[j + 2] = g_USBRecv[j];     //Recv 数组存放到 SPI 缓存数组中
```

```
            }

            //组成帧的格式
            spi_Send[0] = 'P';
            spi_Send[1] = g_USBRecvLength + '0';
            spi_Send[g_USBRecvLength + 2] = 'C';
            spi_SendLength = g_USBRecvLength + 3;

            g_USBRecv[0] = cmdNULL;
            g_USBRecvLength = 0;
            //组帧成功，发送操作，再 SPI 从机的接收中断中执行


            //注释掉 SPI 主机发送程序

            int m = 0;
            for (m = 0; m < spi_SendLength; m++) {
                SPI_send1(SPI_0, spi_Send[m]);
                uart_send1(UART_1, spi_Send[m]);
            }

            //使能 USB 中断
            enable_irq(USB_INTERRUPT_IRQ);
        }

        //将 spi 从机接受的内容传递给 usb_send[]内，准备发送给 USB 主机
                if (spi_flag == 1) {
                    uint_8 t = 0;
                    for (t = 0; t < spi_RecvLength ; t++) {
                        g_USBSend[t] = spi_Recv[t + 2];
                    }
                    g_USBSendLength = spi_RecvLength;
                    spi_flag = 0;
                }

            }        //主循环 end_for
                     //主循环结束
    ======================================================
       }
       #2: isr.c
       void USB0_IRQHandler(void) {
           uint_8 isr_type;
           DISABLE_INTERRUPTS;     //关总中断
           //1. 获取中断类型
           isr_type = usb_get_isr();

           //2. 若不是令牌完成中断，调用相应处理程序
           if (isr_type != USB_TOKDNE_INT) {
               usb_isr_handler(isr_type);     //调用非令牌完成中断的处理程序
               goto USB0_IRQ_exit;
           }

           //3. 是令牌完成中断，执行块数据传输或设备枚举

           //(3.1)清收发 ODD 区，并指定 EVEN 区
           FLAG_SET(USB_CTL_ODDRST_SHIFT, USB0_CTL);

           //(3.2)若是设备枚举请求，进行设备枚举
```

```
131.            if ((USB0_STAT >> 4) == 0) {
132.                usb_enumerate();
133.                goto USB0_IRQ_exit;
134.            }
135.            //(3.3)若不是设备枚举请求，进行数据收发操作
136.
137.            //从端点 2 发送数据至主机
138.            if ((USB0_STAT & 0xF8) == mEP2_IN) {
139.                //若执行发送标志被置 1，则执行发送数据函数
140.                if (g_USBSendFlag == 1) {
141.                    //调用发送数据处理函数
142.                    usb_send(g_USBSend, &g_USBSendLength);
143.
144.                    if (g_USBSendLength == 0)    //剩余未发送的数据长度为 0
145.                            {
146.                        g_USBSendFlag = 0;    //发送完成，执行发送标志清 0
147.                    }
148.                    goto USB0_IRQ_exit;
149.                }
150.                //若无数据需要发送，则向 USB 主机发送一个确认包
151.                BDTtable[bEP2IN_ODD].Cnt = 0;
152.                vEP2State ^= 0x40;
153.                BDTtable[bEP2IN_ODD].Stat._byte = vEP2State;
154.
155.                //从端点 3 接收主机发来的数据
156.            } else if ((USB0_STAT & 0xF8) == mEP3_OUT) {
157.                usb_recv(g_USBRecv, &g_USBRecvLength);
158.                send_flag = 1;
159.            }
160.            USB0_IRQ_exit:
161.
162.            FLAG_SET(USB_ISTAT_TOKDNE_SHIFT, USB0_ISTAT);
163.            ENABLE_INTERRUPTS;    //开总中断
164.            return;
165.        }
166.
167.        void SPI1_IRQHandler(void) {
168.
169.            uint_8 redata;
170.            DISABLE_INTERRUPTS;
171.            redata = SPI_receive1(SPI_1);    //接收主机发送过来的一个字节数据。
172.
173.            if (CreateFrame(redata, spi_Recv) != 0) {
174.                //组帧成功
175.                spi_RecvLength = spi_Recv[1];
176.                spi_flag = 1;
177.            }
178.
179.            ENABLE_INTERRUPTS;
180.        }
```

四线双向

设计思路：

这种思路是有一定难度的，但是最有价值的，两个 MCU 设定唯一的主机和从机；通过利用 SPI 双机通信的特点来进行从机向主机的信息传递；

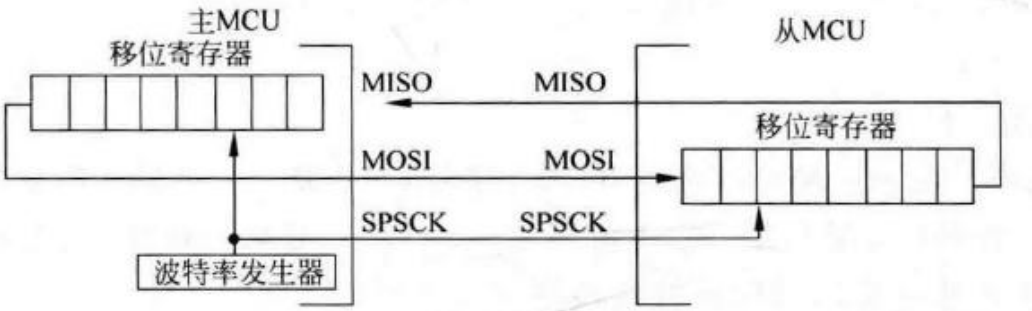


图 11-1 SPI 全双工主-从连接

主机发送从机过程跟上面的八线双向相同，但是从机发往主机则利用主机发送的时序信号‘0’来将从机移位寄存器中的待发送数据按字节逐次发送给主机；

1：（主机/从机--接收/发送） 数据的信号控制

通过定义全局变量 uint_8 rec_flag(主机); uint_8 spi_flag(从机)来控制；每次接收到 USB 发送数据后通过 USB 中断设置为 1；

1）主机 rec_flag 信号设置为 1 代表：

1、不提供从机发主机的时序信号 ‘0’

2、主机接收中断开始判断从机移位寄存器发送过来的内容

2）从机 spi_flag 信号设置为 1 代表：

从机开始根据主机发送的时序信号开始将数据内容逐字节发送到主机

源码

```
void SPI0_IRQHandler(void) {

    //仅在从机发送数据时 rec_flag == 0 调用
    uint_8 redata;
    DISABLE_INTERRUPTS;

    redata = SPI_read1(SPI_0);    //接收主机发送过来的一个字节数据。
    if (rec_flag == 0) {//SPI 主机不处于发送数据时间
        if (redata == '0') {    //若是时序信号则不进行操作

        } else if (CreateFrame(redata, spi_Recv) != 0) {
            //组帧成功
            spi_RecvLength = spi_Recv[1];
            spi_flag = 1;
        }
    }
    ENABLE_INTERRUPTS;
}


#2 USB 从机

void SPI1_IRQHandler(void) {

    uint_8 redata;
    DISABLE_INTERRUPTS;
```

```
    redata = SPI_receive1(SPI_1);     //接收主机发送过来的一个字节数据。

    //SPI 从机发送模块
    if (send_flag == 1) {
        if (redata == '0') {
            int m = 0;
            if (m < spi_SendLength) {
                //收到一个时序信号，后将待发送内容转入移位寄存器
                //待一个时序信号发送过来后，循环发送至 SPI 主机的移位寄存器中
                SPI1_D = spi_Send[m];
                m++;
            }
            if (m == spi_SendLength) {
                send_flag = 0;//发送完毕后，SPI 从机发送标志重新清零
            }

        }
    }

    //SPI 从机接收模块
    if (CreateFrame(redata, spi_Recv) != 0) {
        //组帧成功
        spi_RecvLength = spi_Recv[1];
        spi_flag = 1;
    }

    ENABLE_INTERRUPTS;
}
```

2：主机发往从机的 时序信号控制

1）通过主机 main.c 中对的 rec_flag 标志的判断：

为 1 则不发送时序标志；为 0 则发送时序标志

源码

```
for (;;) {
        //运行指示灯（RUN_LIGHT）闪烁-----------------------------------------------
        mRuncount++;                          //主循环次数计数器+1
        if (mRuncount >= RUN_COUNTER_MAX)  //主循环次数计数器大于设定的宏常数
        {
            mRuncount = 0;                     //主循环次数计数器清零
        }
        //以下加入用户程序-------------------------------------------------------------

        //SPI 主机只要不发送数据，就一直为 SPI 从机提供时序信号‘0’，根据 rec_flag 标志判
    断
        if (rec_flag == 0) {
            SPI_send1(SPI_0, '0');
        }

        //USB 主机向 USB 设备发送要数据命令
        if (g_USBRecv[0] == cmdINTESTDATA && g_USBSendLength != 0) {
            g_USBRecv[0] = cmdNULL;
            g_USBSend[0] = cmdINTESTDATA;
            g_USBSendFlag = 1;
        }
```

```
21.
22.         //USB 主机发送数据，写入到发送数组中，以便主机要数据时进行发送
23.         else if (g_USBRecv[0] == cmdOUTTESTDATA && g_USBRecvLength != 0) {
24.             //禁止 USB 中断
25.             disable_irq(USB_INTERRUPT_IRQ);
26.
27.             //将接收数据的内容传输到发送缓存数组中进行保存，待发送
28.             for (j = 0; j < g_USBRecvLength; j++) {
29.
30.                 spi_Send[j + 2] = g_USBRecv[j];      //Recv 数组存放到 SPI 缓存数组中
31.
32.             }
33.
34.             //组成帧的格式
35.             spi_Send[0] = 'P';
36.             spi_Send[1] = g_USBRecvLength + '0';
37.             spi_Send[g_USBRecvLength + 2] = 'C';
38.             spi_SendLength = g_USBRecvLength + 3;
39.
40.             g_USBRecv[0] = cmdNULL;
41.             g_USBRecvLength = 0;
42.
43.
44.
45.             int m = 0;
46.             for (m = 0; m < spi_SendLength; m++) {
47.                 SPI_send1(SPI_0, spi_Send[m]);
48.             }
49.
50.             //发送一个时序信号，清空 SPI 从机移位寄存器的原有的内容
51.             //保证下一次 SPI 主机中断接收的非‘0’内容一定是 SPI 从机发送的有效数据
52.             SPI_send1(SPI_0, '0');
53.
54.             //SPI 主机发送数据完毕，重新设置 SPI 主机接收数据信号标志为 0;
55.             rec_flag = 0;
56.
57.             //使能 USB 中断
58.             enable_irq(USB_INTERRUPT_IRQ);
59.         }
60.
61.         //将 spi 主机接受的内容传递给 usb_send[]内，准备发送给 USB 主机
62.         if (spi_flag == 1) {
63.             uint_8 t = 0;
64.             for (t = 0; t < spi_RecvLength; t++) {
65.                 g_USBSend[t] = spi_Recv[t + 2];
66.             }
67.             g_USBSendLength = spi_RecvLength;
68.             spi_flag = 0;
69.         }
70.
71.     }
```

3: 从机对发送数据的组帧处理

方法：

跟 SPI 主机发送到 SPI 从机对发送数据的 main.c 里的组帧方法相同，只不过将发送到主机的 SPI 的操作放在了 SPI 从机的接收中断函数里；

PC 方软件设计

登录界面功能样式设计

1. 显示密码

```
                //功能：显示密码，默认不显示
        private void checkBox1_CheckedChanged(object sender, EventArgs e)
        {
            if(this.checkBox1.Checked)
            {
                this.txtPassword.PasswordChar = default(char);
            }
            else
            {
                this.txtPassword.PasswordChar = '*';
            }
        }
```

2. 登录按钮，用户名异常，数据库匹配功能

```
        //功能：实现登录判断功能（MySql 数据库）
        private void btnLogin_Click(object sender, EventArgs e)
        {
            string userName = this.txtUserName.Text;
            string userPassword = this.txtPassword.Text;

            if(userName.Equals("")||userPassword.Equals(""))
            {
                MessageBox.Show("用户名或密码不能为空！");
            }
            else
            {
                string strcon = "server=localhost;database=login;uid=root;pwd=12345
";
                MySqlConnection con = new MySqlConnection(strcon);
                try
                {
                    con.Open();
                    string sql = "select count(*) from user where userName = '" + u
serName + "' and userPassWord ='" + userPassword + "'";
                    MySqlCommand com = new MySqlCommand(sql, con);
                    if(Convert.ToInt32(com.ExecuteScalar())>0)
                    {
                        MessageBox.Show("登录成功！");
                        this.DialogResult = DialogResult.OK;
                        PublicVar.g_flag = 1;
                        this.Dispose();
                        this.Close();
                    }
                    else
                    {
                        MessageBox.Show("用户名或密码错误！");
                    }

                }
                catch  (MySqlException ex)
                {
                    Condole.WriteLine(ex.Message);
```

```
                }
            }
        }
```

3. 注册页面的跳转

```
//功能：实现点击注册按钮后在 program 中判断跳转注册界面
        private void btnRegister_Click(object sender, EventArgs e)
        {
            //string userName = this.txtUserName.Text.Trim();
            //string userPassword = this.txtPassword.Text.Trim();
            this.DialogResult = DialogResult.OK;
            PublicVar.g_flag = 2;
        }
```

注册界面功能样式设计

1. 提交用户名和密码至数据库

```
//功能：1、数据库匹配名称是否重复
        //      2、插入用户输入的账号密码
        //      3、以当前用户登入聊天系统
        private void btSubmit_Click(object sender, EventArgs e)
        {
            string userName = this.textBox1.Text.Trim();
            string userPassword = this.textBox2.Text.Trim();
            if (userName.Equals("") || userPassword.Equals(""))
            {
                MessageBox.Show("用户名或密码不能为空！");
            }
            else
            {
                string strcon = "server=localhost;database=login;uid=root;pwd=12345";
                MySqlConnection con = new MySqlConnection(strcon);
                try
                {
                    con.Open();
                    string sqlCheckUsername = "select count(*) from user where userName = '" + userName + "'";
                    MySqlCommand com = new MySqlCommand(sqlCheckUsername, con);
                    if (Convert.ToInt32(com.ExecuteScalar()) > 0)
                    {
                        MessageBox.Show("用户名重复！\n\r 请重新输入！");
                    }
                    else
                    {
                        string sqlInsert = "insert into user value('" + userName + "','" + userPassword + "')";
                        MySqlCommand com1 = new MySqlCommand(sqlInsert, con);
                        if(Convert.ToInt32(com1.ExecuteScalar())>0)
                        {
                            //MessageBox.Show("注册成功！");
                            //this.DialogResult = DialogResult.OK;
                            //this.Dispose();
                            //this.Close();
                        }
                        MessageBox.Show("注册成功！");
                        this.DialogResult = DialogResult.OK;
```

```
                        PublicVar.g_flag = 2;
                        PublicVar.g_shutdown = 1;
                        this.Dispose();
                        this.Close();
                    }
                }
                catch
                {
                    MessageBox.Show("打开数据库失败！");
                }
            }
        }
```

聊天窗口的功能样式设计

```
 private void BtnSCISend_Click(object sender, EventArgs e)
        {

            bool Flag;
            int i, count = 0;
            int len;

            System.Collections.ArrayList SendData = new
                System.Collections.ArrayList();

            if (!sci.IsOpen)
            {
                return;
            }
            if (this.TbSCISend.Text == string.Empty)
            {
                return;
            }

            len =System.Text.Encoding.Default.GetBytes(this.TbSCISend.Text).Length;

            PublicVar.g_SendByteArray = new byte[len];
            PublicVar.g_SendByteArray =
                System.Text.Encoding.Default.GetBytes(this.TbSCISend.Text);
            Flag = sci.SCISendData(ref PublicVar.g_SendByteArray);
            this.TbSCISend.Text = "";
        }
```

处理发送的消息

```
//功能：拼接接收到且转化后的数据（时间+换行+字符+换行）
    private void SCIUpdateRevtxtbox(Object textbox, string text)
    {
      System.DateTime currentTime = new System.DateTime();
      currentTime = System.DateTime.Now;
      string strY = currentTime.ToString("f");
      if (((TextBox)textbox).InvokeRequired)
      {
        handleinterfaceupdatedelegate InterFaceUpdate = new
          handleinterfaceupdatedelegate(SCIUpdateRevtxtbox);
        this.Invoke(InterFaceUpdate, new object[] { textbox, text });
      }
      else
      {
```

```
            ((TextBox)textbox).AppendText(strY);
            ((TextBox)textbox).AppendText("\r\n");
            ((TextBox)textbox).Text += text;
            ((TextBox)textbox).AppendText("\r\n");
            ((TextBox)textbox).AppendText("\r\n");
            ((TextBox)textbox).SelectionStart =
                                ((TextBox)textbox).Text.Length;
            ((TextBox)textbox).ScrollToCaret();
        }
    }
```

字符串序列化

```
//功能：字符串序列化操作，将字符串转化成二进制流保存
    private void checkBox2_CheckedChanged(object sender, EventArgs e)
    {
        User user = new User();
        FileStream fs = new FileStream("data.bin", FileMode.OpenOrCreate);
        BinaryFormatter bf = new BinaryFormatter();
        user.LoginID = txtUserName.Text.Trim();
        if (checkBox2.Checked)
            user.Pwd = txtPassword.Text.Trim();
        else
            user.Pwd = "";
        if (users.ContainsKey(user.LoginID))
        {
            users.Remove(user.LoginID);
        }
        users.Add(user.LoginID, user);
        bf.Serialize(fs, users);
        fs.Close();
    }
```

字符串反序列化

```
//功能：窗口加载时读取二进制流并反序列化为字符串
    private void login_Load(object sender, EventArgs e)
    {
        Dictionary<string, User> users = new Dictionary<string, User>();
        FileStream fs = new FileStream("data.bin", FileMode.OpenOrCreate);
        if (fs.Length > 0)
        {
            BinaryFormatter bf = new BinaryFormatter();
            //读取存在 Data.bin 里的用户消息
            users = bf.Deserialize(fs) as Dictionary<string, User>;
            //添加 textbox(账号和密码)
            foreach(User user in users.Values)
            {
                this.txtUserName.Text = user.LoginID;
                this.txtPassword.Text = user.Pwd;
            }

        }
        fs.Close();
    }
```

发送图片功能

```
public static string ChangeImageToString(Image image)
    {
      try
      {
        System.IO.MemoryStream ms = new System.IO.MemoryStream();
        image.Save(ms, System.Drawing.Imaging.ImageFormat.Png);
        byte[] arr = new byte[ms.Length];
        ms.Position = 0;
        ms.Read(arr, 0, (int)ms.Length);
        ms.Close();
        string pic = Convert.ToBase64String(arr);
        return pic;

      }
      catch (Exception)
      {
        return "Fail to change bitmap to string!";
      }
    }

    public static Image ChangeStringToImage(string pic)
    {
      try
      {
        byte[] imageBytes = Convert.FromBase64String(pic);
        //读入 MemoryStream 对象
        MemoryStream memoryStream = new MemoryStream(imageBytes, 0, imageBytes.Length);
        memoryStream.Write(imageBytes, 0, imageBytes.Length);
        //转成图片
        Image image = Image.FromStream(memoryStream);

        return image;
      }
      catch (Exception)
      {
        Image image = null;
        return image;
      }
    }
    public static void Base64ToImage(string base64)
    {
      base64 = base64.Replace("", "").Replace("", "").Replace("", "").Replace("", "");//将 base64 头部信息替换
      byte[] bytes = Convert.FromBase64String(base64);
      MemoryStream memStream = new MemoryStream(bytes);
      Image mImage = Image.FromStream(memStream);
      Bitmap bp = new Bitmap(mImage);
      bp.Save(Application.StartupPath + "\\Pic\\lq.jpg" + DateTime.Now.ToString("yyyyMMddHHss") + ".jpg", System.Drawing.Imaging.ImageFormat.Jpeg);//注意保存路径
    }

    public static string ImageToBase64(string fileFullName)
    {
      try
      {
        Bitmap bmp = new Bitmap(fileFullName);
        MemoryStream ms = new MemoryStream();
        bmp.Save(ms, System.Drawing.Imaging.ImageFormat.Jpeg);
        byte[] arr = new byte[ms.Length]; ms.Position = 0;
        ms.Read(arr, 0, (int)ms.Length); ms.Close();
        return Convert.ToBase64String(arr);
      }
      catch (Exception ex)
```

```
        {
            throw ex;
        }
    }

    public static bool IsBase64String(string s)
    {
        s = s.Trim();
        return (s.Length % 4 == 0) && Regex.IsMatch(s, @"^[a-zA-Z0-9\+/]*={0,3}$", RegexOptions.None);
    }



    private void BtnsendPicture_Click(object sender, EventArgs e)
    {
        int len;
        bool Flag;
        string str;

        byte[] sendData = new byte[100000];
        this.TSSLState.Text = "过程提示: 执行发送数据...";

        sendData[0] = (byte)PublicProperty.eCmd.cmdOUTTESTDATA;

        str = ChangeImageToString(Image.FromFile(Application.StartupPath + "\\Pic\\lq.jpg"));

        len = System.Text.Encoding.Default.GetBytes(str).Length;

        byte[] tempArray = System.Text.Encoding.Default.GetBytes(str);
        //byte[] userNameArray = System.Text.Encoding.Default.GetBytes(PublicVar.userName + ": ");



        //PublicVar.g_SendByteArray = new byte[len];
        //PublicVar.g_SendByteArray = System.Text.Encoding.Default.GetBytes(str);

        for (int i = 0; i < tempArray.Length; i++)
        {
            sendData[i + PublicProperty.FRAMEHEADLENGTH] = tempArray[i];
        }

        Flag = usb.sendData(sendData, tempArray.Length + PublicProperty.FRAMEHEADLENGTH);

        System.DateTime currentTime = new System.DateTime();
        currentTime = System.DateTime.Now;
        string strY = currentTime.ToString("f");
        strY += "\r\n";
        strY += (PublicVar.userName + ": ");
        strY += str;
        strY += "\r\n\r\n";

        txtRecv.Text += strY;

        this.pictureBox2.Image = Image.FromFile(Application.StartupPath + "\\Pic\\lq.jpg");

        if (Flag)
        {
            this.TSSLState.Text += "数据发送成功!";
        }
        else
        {
            this.TSSLState.Text += "数据发送失败!";
        }

        PublicVar.g_pictureFlag = 1;
```

```
147.     }
```

设计过程中的技术难点分析

1、多线程控件的消息传递。因为串口接受处理函数属于一个单独的线程，不属于 main 的主线程函数，而接受区域的 textbox 是在主线程中创建的，所以当我们直接用 serialPort.read()读取回字符串然后追加到文本框的时候，串口助手运行时没有反应，会报异常。所以我们使用 invoke 方式，这种方式专门被用于解决从不是创建控件的线程访问它。

2、串口接受函数必须满足一帧发送前进行统一编码，在接收后统一转码处理，要不然会出现中文无法发送和中文乱码的情况。

3、由于本次 C#额外添加用户名和密码的数据库读取和存储功能,用到 MySql 数据库,其中数据库与 C#的连接和数据库表的创建,插入等实现存在一定难度。

4、在准备实现保存用户名时起初不清楚怎么将字符串保存且保存到哪里，后来通过网上查询了解到字符串的序列化与反序列化操作，其原理是在字符串保存到本地文件时，将其转化为二进制流并保存到二进制文件中。在从二进制流里进行反序列化操作即可将其转化为字符串，从而在文本上显示。从而实现记住用户名功能。

5、发送框消息展示，发送格式为“时间-发送人-发送内容”，需要对发送数据格式进行重新设计，将发送人信息等进行重新组帧，并在接收时对文件重新解析并显示。

6、发送图片需要克服许多困难，将图片用 base64 格式转换为字符串形式，以字节流形式进行发送，因为板子和杜邦线质量问题，传输过程中可能会字节缺失的情况，并且受限于硬件情况，只能进行小体积图片慢速传输。

C#模块设计

PC 方界面显示

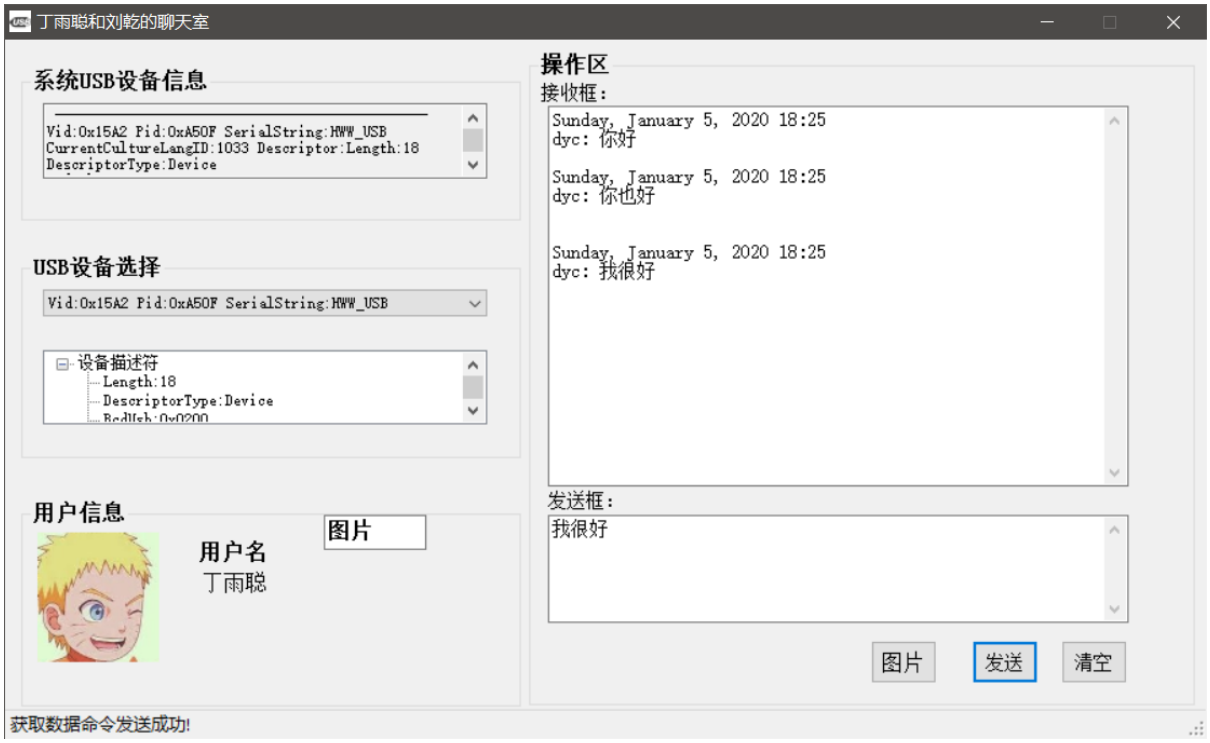
1. 主界面显示

2. 注册界面展示

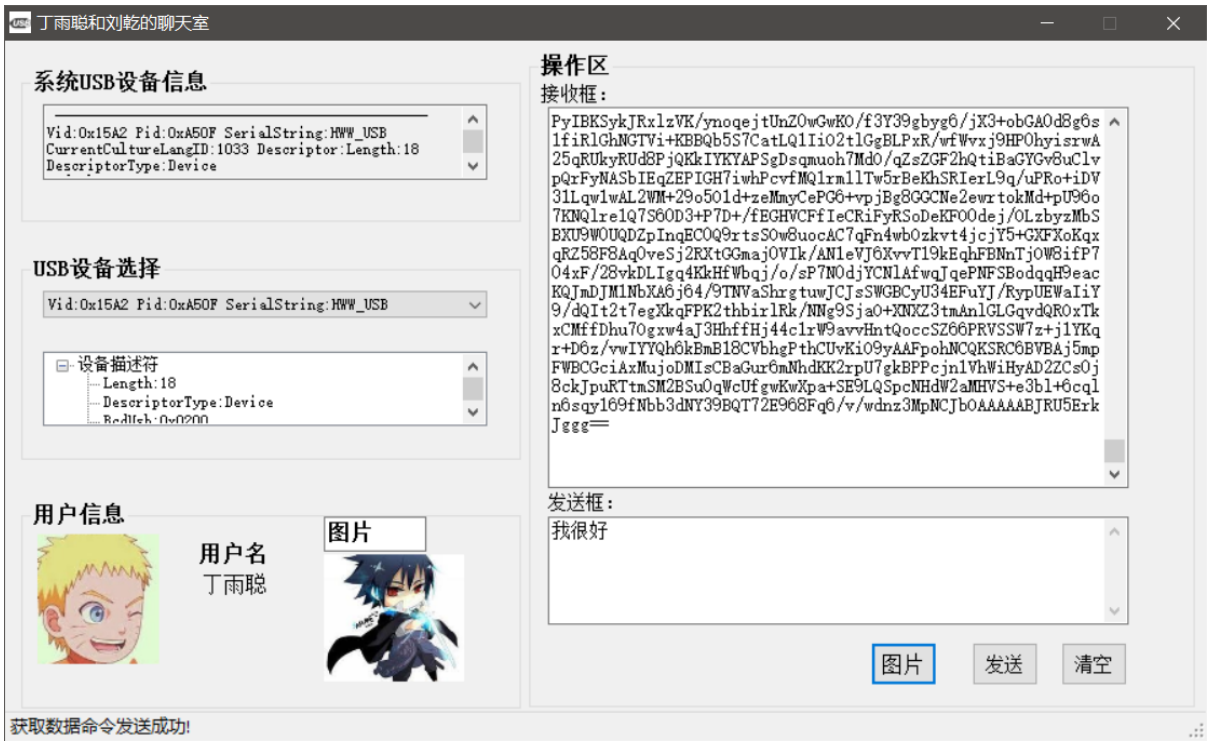
3. 聊天界面显示


丁雨聪和刘乾的聊天室
系统USB设备信息
已连接的设备:0
USB设备选择
没有找到USB设备.
用户信息
图片
用户名
丁雨聪
操作区
接收框:
发送框:
图片
发送
清空
发送获取命令异常!
丁雨聪和刘乾的聊天室
系统USB设备信息
已连接的设备:0
USB设备选择
没有找到USB设备.
用户信息
图片
用户名
刘乾
操作区
接收框:
发送框:
图片
发送
清空
发送获取命令异常!

4. 发送消息展示



5. 发送图片显示（并将转换后的字符串显示在接受框内供调试使用）



实验总结

通过这次课设深入理解了 USB 通信和 SPI 主从双向通信的原理；一开始学习的时候确实花了相当大的时间。

不同于以往，这次课设综合了两个模块，需要对两个模块的整体结构有一个很好的了解才能着手设计整体的通讯。

特别是在主从双向通讯的时候，一开始只完成了八线双向，往后深入做四线双向通信的时候遇到相当大的困难，我通过向身边优秀的同学请教，一点一点的尝试，才学会了。

C#编程因为 net.Framework 版本问题，导致了和数据库连接出现了很多问题，解决了 C#版本问题后对上位机界面进行编辑，最后进行图片发送，发送图片需要将图片转换为 Base64 编码以字节流形式发送，因为 KL25 硬件问题传输会造成传输部分数据丢失，思考了很多时间，接收端对接收的内容进行判断是否为 Base64 编码，是 Base64 就调用 ImagetoString 对 Base64 解码将图片放到 PictureBox 中显示。

甚至到现在，我还有很多很多的知识没学到。但是我相信：有了这种经验，以后尽管遇到困难，我也不会退缩了，问题没能解决之前都很困难，最好的办法就是解决它，自然问题就不困难了。