嵌入式系统设计手札三

**手札说明：我听取老师上次的建议后，修改手札内容组成如下：主要内容时每次实验，将其用实验报告的形式来完成；第二部分就是书本每章提纲挈领的总结和重要概念的简单提要。**

**不再像以前那样书本内容所占篇幅过大。**

## 一：实验部分：

### 实验任务：

1）设计实验：（说明：本次实验我将老师给的实验要求改了一下，通过字符控制灯的颜色）通过串口发送程序发送字符：‘r’,‘g’,‘b’ 分别控制红绿蓝三种灯的亮）。

2）进阶实验：（说明：因为本次实验直接涉及到组帧功能，所以将设计实验中的open close一起放到进阶实验当中实现）设计createframe() 函数实现在接收中断服务例程中每次接受单字节字符输入并进行组帧，通过对每次组帧成功后保存的字符串进行判断来控制不同颜色的小灯的亮灭。

并且在C#的串口通信窗口程序中封装八个按钮控制灯的亮暗和颜色切换

### 实验环境：

MCU编程环境IDE: KDS

C#窗口程序IDE:VS2017

### 实验目的：

通过完成两个实验来学习并使用的串口通信构件的基本功能和自主添加额外的新函数实现新的功能进行组帧识别来控制不同颜色的灯的亮暗；

#### 实验任务一：

初步学习了串口通信构件中的接收中断服务例程的使用：实现MCU接受单字节字符内容的输入和发送。

#### 实验任务二：

进一步学习了如何通过使用组帧的方式来用MCU串口接受单字节字符输入并通过creatframe（）函数来组成字符串帧并进行识别判断来控制七种颜色灯切换和实现关灯控制操作；

#### 实验任务三：

将控制小灯颜色的功能封装在C#编写的串口通信窗口程序中

### 实验内容：

（实验中问题解决方法和思路的具体阐述）

#### 实验任务一：

思路：调用接收中断服务例程**void** **UART1\_IRQHandler**(**void**)；在接受每次单字节字符输入后用开关语句进行选择，控制不同颜色的灯的亮灭。

##### 源码：

//串口2接收中断服务例程

**void** **UART2\_IRQHandler**(**void**) {

uint\_8 ch;

uint\_8 flag;

DISABLE\_INTERRUPTS; //关总中断

**if** (uart\_get\_re\_int(UART\_2)) {

ch = uart\_re1(UART\_2, &flag); //调用接收一个字节的函数，清接收中断位

**if** (flag) {

uart\_send1(UART\_2, ch);

**switch** (ch) {//使用开关语句进行条件判断来控制不同的灯的亮暗；

**case** 'b': {

light\_change(LIGHT\_BLUE);

light\_control(LIGHT\_RED, LIGHT\_OFF);

light\_control(LIGHT\_GREEN, LIGHT\_OFF);

uart\_send\_string(UART\_2, "blue\_UART\_2!\r\n");

**break**;

}

**case** 'r': {

light\_change(LIGHT\_RED);

light\_control(LIGHT\_BLUE, LIGHT\_OFF);

light\_control(LIGHT\_GREEN, LIGHT\_OFF);

uart\_send\_string(UART\_2, "red\_UART\_2!\r\n");

**break**;

}

**case** 'g': {

light\_change(LIGHT\_GREEN);

light\_control(LIGHT\_BLUE, LIGHT\_OFF);

light\_control(LIGHT\_RED, LIGHT\_OFF);

uart\_send\_string(UART\_2, "green\_UART\_2!\r\n");

**break**;

}

**default**: {

}

}

}

}

ENABLE\_INTERRUPTS;

}

##### main.c：

//说明见工程文件夹下的Doc文件夹内Readme.txt文件

//======================================================================

#include "includes.h" //包含总头文件

int main(void) {

//1. 声明主函数使用的变量

uint\_32 mRuncount; //主循环计数器

uint\_32 mLight\_chang\_num; //灯状态变换次数

//2. 关总中断

DISABLE\_INTERRUPTS;

//3. 初始化外设模块

light\_init(LIGHT\_RED, LIGHT\_OFF); //红灯初始化

light\_init(LIGHT\_BLUE, LIGHT\_OFF); //蓝灯初始化

light\_init(LIGHT\_GREEN, LIGHT\_OFF);

uart\_init(UART\_0, 9600); //初始化串口0，波特率为9600

uart\_init(UART\_1, 9600); //初始化串口1，波特率为9600

uart\_init(UART\_2, 9600); //初始化串口2，波特率为9600

//uart\_init(UART\_Debug, 9600); //初始化UART\_Debug，波特率为9600

//串口发送初始化提示

uart\_send\_string(UART\_0, "使用uart\_send\_string发送：Hello Uart\_0!\r\n");

uart\_send\_string(UART\_1, "使用uart\_send\_string发送：Hello Uart\_1!\r\n");

uart\_send\_string(UART\_2, "使用uart\_send\_string发送：Hello Uart\_2!\r\n");

printf("使用printf函数发送：Hello Uart\_2!\r\n");

//4. 给有关变量赋初值

mRuncount = 0; //主循环计数器

mLight\_chang\_num = 0; //灯状态变换次数

//5. 使能模块中断

uart\_enable\_re\_int(UART\_0); //使能串口0接收中断

uart\_enable\_re\_int(UART\_1); //使能串口1接收中断

uart\_enable\_re\_int(UART\_2); //使能串口2接收中断

//6. 开总中断

ENABLE\_INTERRUPTS; //开总中断

//进入主循环

//主循环开始==================================================================

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

//light\_change(LIGHT\_BLUE); //蓝色运行指示灯（RUN\_LIGHT\_BLUE）状态变化

mLight\_chang\_num++; //灯状态变换次数+1

printf("灯状态变换次数mLight\_chang\_num=%d\n",mLight\_chang\_num);

uart\_send\_string(UART\_0, "UART\_0!\r\n");

uart\_send\_string(UART\_1, "UART\_1!\r\n");

uart\_send\_string(UART\_2, "UART\_2!\r\n");

}\*/

//以下加入用户程序--------------------------------------------------------

} //主循环end\_for

//主循环结束==================================================================

}

##### 实验小结：

（关于这次实验所遇到的问题和解决的方法）

这次试验是初步学习使用UART构件中的串口的接收中断服务例程来实现字符的接受和转发；而且结合了前几次实验所学的GPIO控件的使用：控制小灯的亮暗；

发现的问题：一次只能接受一个字节的字符输入，及不论一次输入多少字符发送，只要含有能改变灯的颜色的字符在输入的内容中，则会改变灯的颜色，不能精准的判断命令的正确与否，所以需要设计一个能精准识别命令来控制小灯的亮暗的函数——组帧createframe() ；

#### 实验任务二：

思路：通过组帧函数来对每次输入的字符进行组帧——字符串buffer[64]，每次组帧成功后通过if-else if …else进行条件判断，依次进行判断，对帧内容符合编码要求的字符串 转入控制灯亮的程序；

##### 源码：

**#define** FrameHead (0x50) //帧头 ASCII码对应P

**#define** FrameTail (0x43) //帧尾 ASCII码对应C

uint8\_t **CreateFrame**(uint\_8 Data, uint\_8 \* buffer) {

**static** uint\_8 frameLen = 0; //帧的计数器

uint\_8 frameFlag; //组帧状态

frameFlag = 0; //组帧状态初始化

//根据静态变量frameCount组帧

**switch** (frameLen) {

**case** 0: //第一个数据

{

**if** (Data == FrameHead) //收到数据是帧头FrameHead

{

buffer[0] = Data;

frameLen++;

frameFlag = 0; //组帧开始

}

**break**;

}

**case** 1: //第二个数据，该数据是随后接收的数据个数

{

buffer[1] = Data - 0x30;

frameLen++;

**break**;

}

**default**: //其他情况

{

//第二位数据是有效数据长度,根据它接收余下的数据直到帧尾前一位

**if** (frameLen >= 2 && frameLen <= (buffer[1] + 1)) {

buffer[frameLen] = Data;

frameLen++;

**break**;

}

//若是末尾数据则执行

**if** (frameLen >= (buffer[1] + 2)) {

**if** (Data == FrameTail) //若是帧尾

{

buffer[frameLen] = Data; //将帧尾存入缓冲区

frameFlag = 1; //组帧成功

}

frameLen = 0; //计数清0，准备重新组帧

**break**;

}

}

} //switch\_END

**return** frameFlag; //返回组帧状态

}

##### 实验小结：

组帧函数是利用老师给的组帧函数，并没有做过多的修改。但是我仔细学习了老师源码，对源码的分析如下:老师首先限定了帧的格式：首尾P,C,第二位控制字段。对错误帧也运用设置了framelen这个变量来进行检验——每次若帧头帧尾不符合要求则自动指针指向buffer第一位即起到了检验输入的字符串是否满足正确的帧格式的作用；

##### 源码：

//串口一的终端服务例程

void UART1\_IRQHandler(void) {

uint\_8 ch;

uint\_8 flag;

uint\_8 buffer[64];//函数内部的局部变量，用来保存录入的正确格式的帧

DISABLE\_INTERRUPTS; //关总中断

if (uart\_get\_re\_int(UART\_1)) {

ch = uart\_re1(UART\_1, &flag); //调用接收一个字节的函数，清接收中断位

if(CreateFrame(ch,buffer)!=0){

if (buffer[2] == 'r' && buffer[3] == 'e' && buffer[4] == 'd') {

light\_control(LIGHT\_BLUE, LIGHT\_OFF);

light\_control(LIGHT\_GREEN, LIGHT\_OFF);

light\_control(LIGHT\_RED, LIGHT\_ON);

}

else if (buffer[2] == 'b' && buffer[3] == 'l' && buffer[4] == 'u'

&& buffer[5] == 'e') {

light\_control(LIGHT\_GREEN, LIGHT\_OFF);

light\_control(LIGHT\_RED, LIGHT\_OFF);

light\_control(LIGHT\_BLUE, LIGHT\_ON);

}

else if (buffer[2] == 'g' && buffer[3] == 'r' && buffer[4] == 'e'

&& buffer[5] == 'e' && buffer[6] == 'n') {

light\_control(LIGHT\_BLUE, LIGHT\_OFF);

light\_control(LIGHT\_RED, LIGHT\_OFF);

light\_control(LIGHT\_GREEN, LIGHT\_ON);

}

else if (buffer[2] == 'c' && buffer[3] == 'l' && buffer[4] == 'o'

&& buffer[5] == 's' && buffer[6] == 'e') {

light\_control(LIGHT\_BLUE, LIGHT\_OFF);

light\_control(LIGHT\_RED, LIGHT\_OFF);

light\_control(LIGHT\_GREEN, LIGHT\_OFF);

}

else if (buffer[2] == 'v' && buffer[3] == 'i' && buffer[4] == 'o'

&& buffer[5] == 'l' && buffer[6] == 'e'

&& buffer[7] == 't') {

light\_control(LIGHT\_GREEN, LIGHT\_OFF);

light\_control(LIGHT\_BLUE, LIGHT\_ON);

light\_control(LIGHT\_RED, LIGHT\_ON);

}

else if (buffer[2] == 'w' && buffer[3] == 'h' && buffer[4] == 'i'

&& buffer[5] == 't' && buffer[6] == 'e') {

light\_control(LIGHT\_BLUE, LIGHT\_ON);

light\_control(LIGHT\_RED, LIGHT\_ON);

light\_control(LIGHT\_GREEN, LIGHT\_ON);

}

else if (buffer[2] == 'c' && buffer[3] == 'y' && buffer[4] == 'a'

&& buffer[5] == 'n') {

light\_control(LIGHT\_RED, LIGHT\_OFF);

light\_control(LIGHT\_BLUE, LIGHT\_ON);

light\_control(LIGHT\_GREEN, LIGHT\_ON);

}

else if(buffer[2] == 'y' && buffer[3] == 'e' && buffer[4] == 'l'

&& buffer[5] == 'l' && buffer[6] == 'o'

&& buffer[7] == 'w') {

light\_control(LIGHT\_BLUE, LIGHT\_OFF);

light\_control(LIGHT\_GREEN, LIGHT\_ON);

light\_control(LIGHT\_RED, LIGHT\_ON);

}

else {

}

}

if (flag) {

uart\_send1(UART\_1, ch); //向原串口发回一个字节

}

}

ENABLE\_INTERRUPTS;

}

##### 实验小结：

这个终端服务例程就是比较简单了，只是每次调用终端服务例程时调用组帧函数createframe() 通过返回值framflag 是否为1进行判断：是否组帧成功。一旦成功则进行帧判断，条件语句判断灯的情况。

#### 实验任务三：

在C#编写的串口通信窗口程序项目中添加八个按钮分别实现对应的功能，创建八个按钮，在每个按钮的方法里分别通过已经封装好的发送函数，向MCU串口里发送对应的帧来控制小灯。最后在打开/关闭串口按钮中添加this.button\*.Enable=false/true；来控制按钮的有效与否。

##### 源码：

private void button1\_Click(object sender, EventArgs e)

{

if (!sci.IsOpen)

{

//状态条进行提示

this.TSSLState.Text += "请先打开串口!";

return;

}

else

{

//创建容器存储发送数据

PublicVar.g\_SendByteArray = new byte[10];

PublicVar.g\_SendByteArray =

System.Text.Encoding.Default.GetBytes("P3redC");

sci.SCISendData(ref PublicVar.g\_SendByteArray);

this.TSSLState.Text = "红灯亮";

}

}

private void button2\_Click(object sender, EventArgs e)

{

if (!sci.IsOpen)

{

//状态条进行提示

this.TSSLState.Text += "请先打开串口!";

return;

}

else

{

PublicVar.g\_SendByteArray = new byte[10];

PublicVar.g\_SendByteArray =

System.Text.Encoding.Default.GetBytes("P5greenC");

sci.SCISendData(ref PublicVar.g\_SendByteArray);

this.TSSLState.Text = "绿灯亮";

}

}

private void button3\_Click(object sender, EventArgs e)

{

if (!sci.IsOpen)

{

//状态条进行提示

this.TSSLState.Text += "请先打开串口!";

return;

}

else

{

PublicVar.g\_SendByteArray = new byte[10];

PublicVar.g\_SendByteArray =

System.Text.Encoding.Default.GetBytes("P4blueC");

sci.SCISendData(ref PublicVar.g\_SendByteArray);

this.TSSLState.Text = "蓝灯亮";

}

}

private void button4\_Click(object sender, EventArgs e)

{

if (!sci.IsOpen)

{

//状态条进行提示

this.TSSLState.Text += "请先打开串口!";

return;

}

else

{

PublicVar.g\_SendByteArray = new byte[10];

PublicVar.g\_SendByteArray =

System.Text.Encoding.Default.GetBytes("P6violetC");

sci.SCISendData(ref PublicVar.g\_SendByteArray);

this.TSSLState.Text = "紫灯亮";

}

}

private void button5\_Click(object sender, EventArgs e)

{

if (!sci.IsOpen)

{

//状态条进行提示

this.TSSLState.Text += "请先打开串口!";

return;

}

else

{

PublicVar.g\_SendByteArray = new byte[10];

PublicVar.g\_SendByteArray =

System.Text.Encoding.Default.GetBytes("P6yellowC");

sci.SCISendData(ref PublicVar.g\_SendByteArray);

this.TSSLState.Text = "黄灯亮";

}

}

private void button6\_Click(object sender, EventArgs e)

{

if (!sci.IsOpen)

{

//状态条进行提示

this.TSSLState.Text += "请先打开串口!";

return;

}

else

{

PublicVar.g\_SendByteArray = new byte[10];

PublicVar.g\_SendByteArray =

System.Text.Encoding.Default.GetBytes("P5whiteC");

sci.SCISendData(ref PublicVar.g\_SendByteArray);

this.TSSLState.Text = "白灯亮";

}

}

private void button7\_Click(object sender, EventArgs e)

{

if (!sci.IsOpen)

{

//状态条进行提示

this.TSSLState.Text += "请先打开串口!";

return;

}

else

{

PublicVar.g\_SendByteArray = new byte[10];

PublicVar.g\_SendByteArray =

System.Text.Encoding.Default.GetBytes("P4cyanC");

sci.SCISendData(ref PublicVar.g\_SendByteArray);

this.TSSLState.Text = "青紫灯亮";

}

}

private void button8\_Click(object sender, EventArgs e)

{

if (!sci.IsOpen)

{

//状态条进行提示

this.TSSLState.Text += "请先打开串口!";

return;

}

else

{

PublicVar.g\_SendByteArray = new byte[10];

PublicVar.g\_SendByteArray =

System.Text.Encoding.Default.GetBytes("P5closeC");

sci.SCISendData(ref PublicVar.g\_SendByteArray);

this.TSSLState.Text = "关灯";

}

}



##### 实验小结：

学习了C#的GUI编程。通过这次实验让我初步了解了怎样通过串口程序实现窗口界面程序和MCU来进行通信。

### 实验总结：

这三个小实验花了不少时间，让我发现自身的编程能力还有待加强。对MCU的理解还是不深入，是要花大功夫才能有所进步的。实验过程中要有耐心，不能浮躁。对待实验态度要认真。