设计题目（五） 单片机红外、串行异步收发的应用

一、设计要求

使用 STC 单片机上的红外接收器和配套的红外遥控器，实现对 STC 单片机实验箱上的资源进行控制和交互。

（1）STC 单片机能正确接收到红外遥控器的编码信息，并显示（不限制显示介质，串口或 1602）

（2）能控制 LED

（3）能实现更复杂的显示交互和控制功能

二、设计代码与注释

main.c 文件：

```
#include "reg51.h"
#include "intrins.h"
#include "math.h"
#include "stdio.h"
#define FOSC    6000000L
#define BAUD 115200

char scankey();                          //声明遥控器按键识别函数

sfr AUXR    =0x8E;
sfr AUXR1 =0xA2;
sfr AUXR2 =0x8F;
sfr TH2      =0xD6;
sfr TL2      =0xD7;
sfr P0M1    =0x94;
sfr P0M0    =0x93;
sfr P2M1    =0x95;
sfr P2M0    =0x96;
sfr P3M1    =0xB1;
sfr P3M0    =0xB2;
```

```
sfr P4 = 0XC0;
sbit P36    =P3^6;
sbit P46    =P4^6;
sbit P47    =P4^7;
bit busy=0;
unsigned char irdata[4]={0,0,0,0};      //定义字符型数组，存放每次红外解码的编码信息
unsigned char cal[20];                  //计算器辅助寄存数组
bit flag0=0;

void SendData(unsigned char dat)
//串口传送函数（传送单字符）
{
    while(busy);
    SBUF=dat;
    busy=1;
}

void SendString(char *s)
//串口字符串传送函数
{
    while(*s!='\0')
        SendData(*s++);
}

unsigned int high_level_time()
//高电平计量函数（解码）
{
    TL0=0;
    TH0=0;
    TR0=1;
```

```
    while(P36==1)
    {
        if(TH0>=0xEE)
            break;
    }
    TR0=0;
    return(TH0*256+TL0);
}

unsigned int low_level_time()
//低电平计量函数（解码）
{
    TL0=0;
    TH0=0;
    TR0=1;
    while(P36==0)
    {
        if(TH0>=0xEE)
            break;
    }
    TR0=0;
    return(TH0*256+TL0);
}

void int2() interrupt 10
//外部中断 INT2 的 ISR，实现红外信号解码
{
    unsigned char i,j;
    unsigned int count=0;
    unsigned char dat=0;
```

```
    AUXR2&=0x00;
    count=low_level_time();        //首先测量低电平宽度（时长）

  if(count<0 || count>5000)     return;

  count=high_level_time();        //之后测量高电平宽度（时长）
  if(count<2000 || count>4500)     return;

    for(i=0;i<4;i++)              //进入正式解码状态（4 字节编码）
    {
    P36=1;                        //拉高 P3.6 电平
        dat=0;
        for(j=0;j<8;j++)
          {
              count=low_level_time();
                if(count<150 || count>400)
                return;
                count=high_level_time();
                  if(count>150 && count<400)
                        dat>>=1;
    //符合“0”编码条件，在最高位存入 0（8 位字符型变量进行左移运算，最高位自动补 0）
                  else if(count>400 && count<1000)
                  {
                      dat>>=1;
                      dat|=0x80;
    //符合“1”编码条件，在最高位存入 1（8 位字符型变量进行左移运算，最高位自动补 0，再对最高位置位）
                  }
                  else return;
              }
```

```
                irdata[i]=dat; //解码出的每一字节编码存入编码数组
        }
         flag0=1;          //标志位置位，表示完整完成一次红外解码
}

void uart1() interrupt 4
//串口传送/接收的 ISR
{
    if(RI)
        RI = 0;
    if(TI)
        TI = 0;
      busy=0;
}

void getresult(unsigned int op1,unsigned int op2, char p)
//计算器实现函数（加、减、乘、除、阶乘）
{
    unsigned int res=1,count;
    unsigned char resc[20];
                    if(p == '+')    res = op1 + op2;
                    else if(p == '-')
                    {
                       if(op1 >= op2)
                         res = op1 - op2;
                    if(op1 < op2) {SendData('-'); res = op2 - op1;}
                    }
                    else if(p == '*')   res = op1 * op2;
                    else if(p == '/')   res = op1 / op2;
                    else if(p == '!')
                        for(count =1 ; count<=op1; count++)
```

```
                    res = count*res;
                sprintf(resc,"%u",res);
                SendString(resc);
                SendString("\r\n");
}

void main()
{
    unsigned char opra,key;
    unsigned int op1,op2,i,j,flag;
    P36=1;
    P2M1 = 0x00;
    P2M0 = 0x00;
    P3M1 = 0x00;
    P3M0 = 0x00;
    TMOD = 0x00;
    SCON = 0x40;        //串口 1 工作在模式 1
    AUXR = 0x40;        //设置定时器 1 为串口 1 波特率发生器
    TL1 = (65536-((FOSC/4)/BAUD));
    TH1 = (65536-((FOSC/4)/BAUD))>>8;
    TR1 = 1;
    ES = 1;
    AUXR2 |= 0x10;      //使能外部中断 INT2
    EA=1;
    SendString("\r\n--begin-----\r\n");
     while(1)
       {
           AUXR2|=0x10;    //每次从 INT2 的 ISR 意外退出，重
新打开 INT2 中断
          if(flag0==1)
          {
```

```
                AUXR2&=0x00;
                flag0=0;
                //SendString("\r\n--Received IR data is-----\r\n");
                //for(k=0;k<4;k++)        //通过串口输出红外编码
                  //SendData(irdata[k]);
                //SendString("\r\n");
                key = scankey();          //读取编码对应按键信息
                if(key == '$') {i = 0; SendString("\r\n--OUT--\r\n");
continue;}
    //当按键信息为'$'，串口输出提示信息，并跳过本次循环
                if(key == '@') SendString("Control LED\r\n");
    //当按键信息为'@'，控制 LED 灯
                else
    //不为以上两种情况，则说明返回的是计算器相关按键信息，所以进入计算器模式
                  {
                  if(!i) SendString("Open Calculator\r\n");
    //如果是（在本次循环内）第一次进入计算器模式，串口输出提示信息
                  cal[i] = key; //将每个返回字符存入辅助寄存数组
                  i++;
                  SendData(key);
                  }
    if((key == '+')||(key == '-')||(key == '*')||(key == '/')||(key == '!'))
                  {
                  op1 = cal[0]-48;
                  for(j = 1; j < i-1; j++)   op1 = (cal[j]-48) + op1*10;
                  //提取操作数 1
                  opra = key;          //运算符暂存
                  flag = i;
                  }
```

```
                else if(key == '=')
                    {
                     if(opra != '!') //如果运算符不是阶乘
                        {
                            op2 = cal[flag]-48;
        for(j = flag+1; j < i-1; j++)    op2 = (cal[j]-48) + op2*10;
        //提取操作数 2
                            getresult(op1,op2,opra);
                        }
                        else if(opra == '!')   //如果运算符是阶乘
                             getresult(op1,0,opra);
                            i = 0;
                            op1 = op2 = 0;
                        }
            }
    }
}
//遥控器按键信息识别函数
char scankey(){
    if(irdata[3] == 0xBA) {P46 = !P46; return '@';}//控制 LED 亮灭
    if(irdata[3] == 0xB8) {P47 = !P47; return '@';}
    if(irdata[3] == 0xE9) return '0';
    if(irdata[3] == 0xF3) return '1';
    if(irdata[3] == 0xE7) return '2';
    if(irdata[3] == 0xA1) return '3';
    if(irdata[3] == 0xF7) return '4';
    if(irdata[3] == 0xE3) return '5';
    if(irdata[3] == 0xA5) return '6';
    if(irdata[3] == 0xBD) return '7';
    if(irdata[3] == 0xAD) return '8';
    if(irdata[3] == 0xB5) return '9';
```

```
    if(irdata[3] == 0xF8) return '-';
    if(irdata[3] == 0xEA) return '+';
    if(irdata[3] == 0xF6) return '=';
    if(irdata[3] == 0xBB) return '/';
    if(irdata[3] == 0xBC) return '*';
    if(irdata[3] == 0xB9) return '!';
    return '$';
}
```

三、设计思路

1.**串口通信**:

在本设计实验中，我使用了串口 1，使其工作在模式 1（即工作在异步接发器状态），设置波特率 115200，由定时器 1 作为波特率发生器，所以需将定时器 1 设置为工作在模式 0，且设定定时器 1 的速度为系统时钟，而波特率的计算公式为:

波特率 = 定时器 1 的溢出率/4

所以，初始化定时器 1 的计数初值(TH1,TL1)的计算公式为:

【TH1,TL1】 = 65536-SYSclk/(串口 1 的波特率*4)

单片机通过串口发送数据的时候，要将需发送数据写入 SBUF 缓冲寄存器，当发送完一串数据，自动将发送中断请求标志位 TI 置位，进入中断（串口 1 中断，中断号 4）。当 TI 在中断中被软件清 0 后，即可继续发送下一组数据串。

2.**红外解码**:

首先简要回顾红外通信中的一些重要概念:

1. 原始信号脉冲首先被调制在一定的载波上，再通过红外光的方式传送，在发送端与接收端被分别调制/解调。
2. 信号脉冲使用 PPM（脉冲位置调制），每一个二进制的 1 或 0，通过脉冲的时间间隔区分。
3. 信号的数据格式包含：起始码、用户码、数据码、数据反码。真正有效的为后三类编码，一共 32 位。这 4 字节编码也是我们需要解码出的部分。
4. 对于信号脉冲的解码，通过计量高低电平的宽度（时长）来判断（定时器 0 作为计数器）。

所以，对于单片机的红外解码功能，我总结有两个环节：硬件部分，是单片机外设的红外接收器将接收的红外信号转化为电平脉冲信号，通过 P3.6 端口发送给单片机；软件部分，在单片机内通过软件方式，计量高低电平时长的判断方法解码出最终的编码

数据。

而当红外信号能被完整解码出来之后，再将缓存的编码数据利用串口发送至 PC 显示就是水到渠成了。

3.**简单计算器**:

设计的要求中提到了要能“实现更复杂的显示交互和控制功能”。显然，如果能利用红外编码信息触发相应的其它单片机功能，就可以使用遥控器直接产生与单片机的交互和控制。一种直观的思路是利用遥控器的按键特点和单片机片内资源实现简易的计算器功能。我的设计目标是，逐次按下遥控器的数字键、约定的运算符键等，可以在 PC 端的串口上显示出一行数学算术式，并给出对应的运算结果。而按下其它计算器非相关按键，则会直接返回初始状态，使单片机重新等待红外输入。

我的简易计算器**实现原理**如下:

创建计算辅助字符数组 cal[ ]，在计算器模式下，遥控器每按下一个计算器相关按键，就将按键信息对应的字符（在其它函数中提前约定）（该字符可能为个位数字“0”~“9”或运算符“+”、“-”、“*”、“/”、“!”或算式输入结束标志“=”）送入 cal[ ]中保存。当检测到当前存入 cal[ ]的字符为运算符，则提取 cal[ ]中该字符之间的各位数字字符，结合成对应的操作数 1（整型），同理，当检测到当前存入字符为“=”，则提取运算符与“=”间的各位数字字符，组合成对应的操作数 2（整型）（如果运算符为阶乘 “!”，则仅提取操作数 1）。以上操作的目的是使得计算器的操作数可以有多位输入，而**输入的限制范围则是每个操作数以及运算结果（均为整型）所对应的存储类型的溢出界限值**，比如，对于 16 位无符号整型，无溢出范围: 0~65535；对于 16 位有符号整型，无溢出范围：-32768~+32767。如果选用长整型存储，则范围还能扩大。

在我设计的单片机软件逻辑中，有 getresult()函数，当操作数 1/2 以及运算符已经得到，就调用该函数，在其中直接进行相关整型数运算，最后利用 sprintf()函数将整型数转换为字符串，通过串口输出，在 PC 端显示。

综上所述，我对本题设计的整体功能及可执行操作叙述如下:

1. 按下遥控器任一按键，通过串口连接，在 PC 端显示出按键对应的用户码、数据码、数据反码共 4 字节数据。
2. 按下“CH+”、“CH-”，进入 **LED 控制模式**，分别控制 LED9/10 的亮灭；
3. 按下数字键（“0”~“9”）、运算符（“+”、“-”、“*”、“/”、“!”）或算式输入结束标志（“=”），进入**计算器模式**，对于计算器模式而言，完整的**工作周期**是以**按下数字键开始，填写完整 1**

个算式（包括操作数、运算符），到按下“=”以终结算式，输出结果并跳至屏幕下一行;

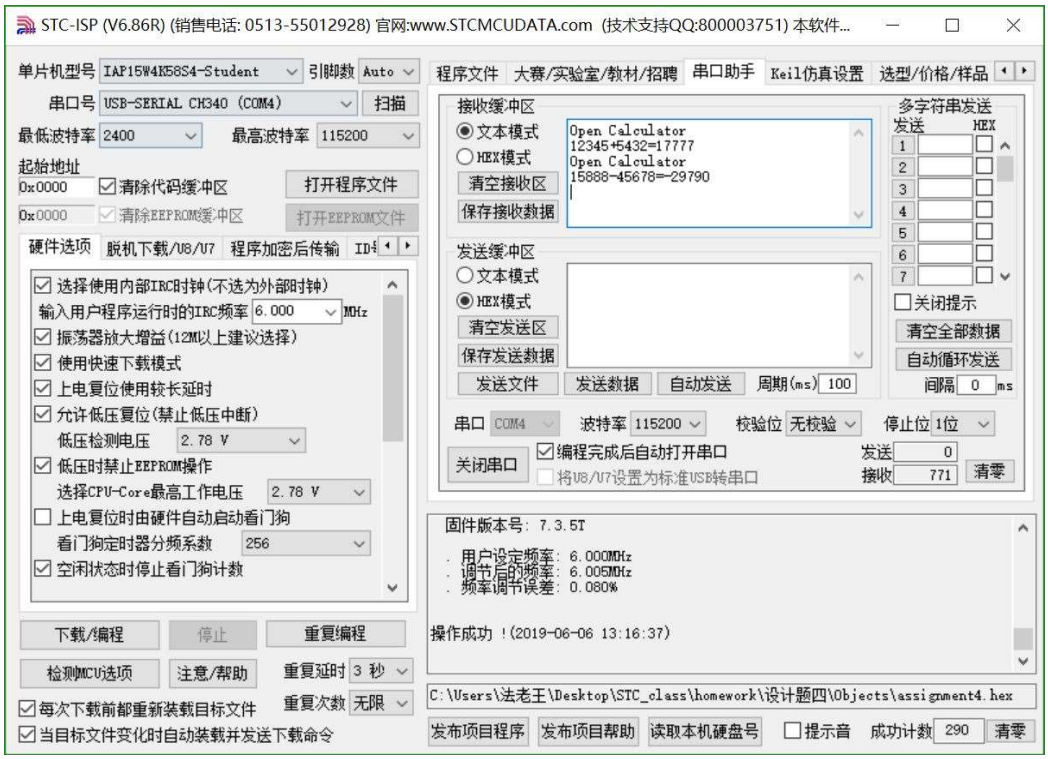
4. 在计算器模式下，可随时按下该模式与 LED 控制模式中**未定义的按键**退出当前**工作周期**，进入新的周期。
5. 计算器模式与 LED 控制模式不冲突，可并行，当执行在计算器状态，也可随时控制 LED。

四、具体实现展示

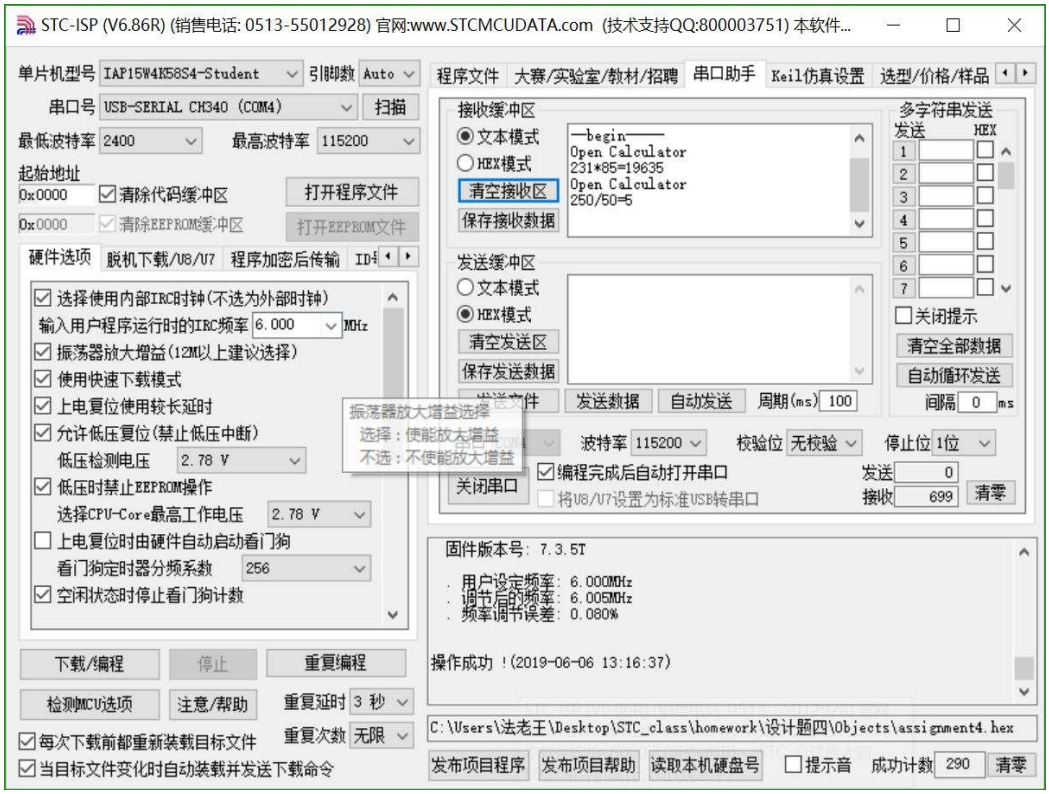
解码展示

按下按键，通过串口在 PC 端显示出对应的编码 4 字节（用户码、数据码、数据反码），图中为按键“0”~”9”对应的解码结果。

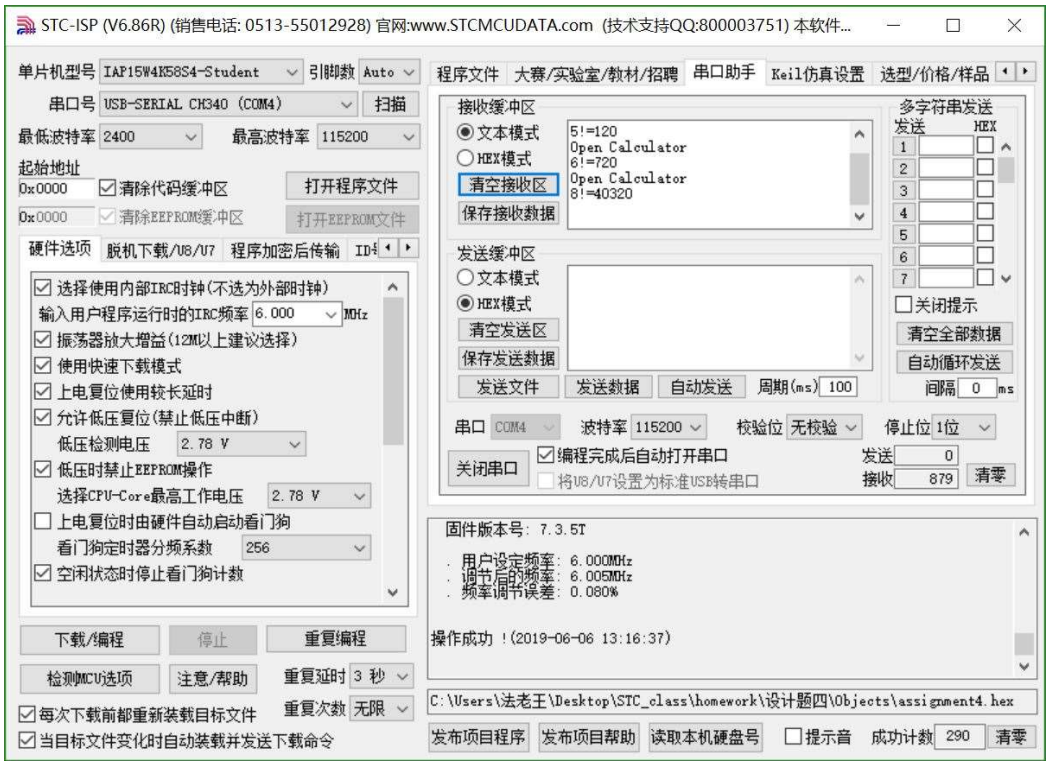
计算器模式的加、减法演示



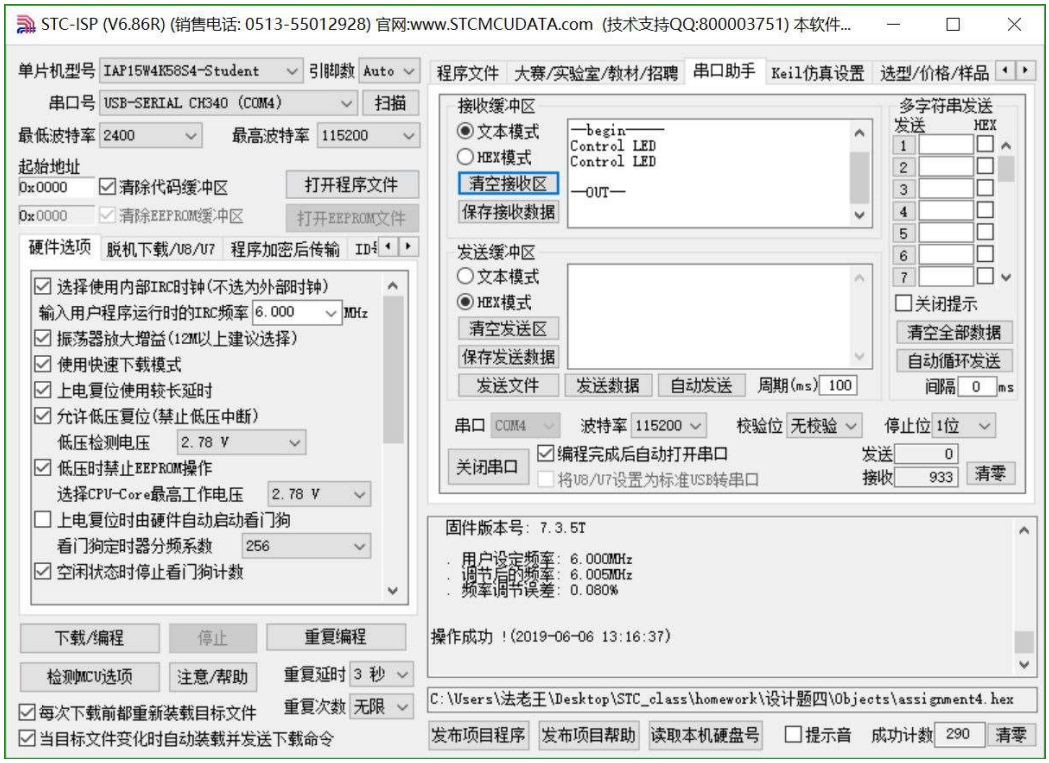
计算器模式的乘、除法演示



计算器模式的阶乘演示



模式的退出演示



五、遇到的问题和设计重点

1. 红外软件解码判断语句的调整:

在软件解码部分，使用定时器 0 作为计数器，测量高/低电平的长度。

利用定时器 0 计数时钟为系统时钟 12 分频，所以有公式:

12 * count / 6,000,000 = T

T 对应电平持续时间的估计边界，count 对应定时器 0 的计数值。

我通过书本给出的电平持续时间范围计算，得到与书上代码中一致的计数值范围。但是在实际测试中，该范围并不能将信号脉冲解码出来。所以，我尝试适当放宽估测的范围，最后在本报告中设置的参数范围下成功解码并串口输出显示。

2. 计算器实现过程中的数据类型选择:

我一开始，在 getresult()函数中的 sprintf()（整型转字符型）语句中，使用%d 格式说明符，在这种情况下，我计算 8!（8 的阶乘）即发生溢出。后面我仔细查找资料，发现 d 表示有符号整数，所以 8! 的结果 40320>32767(16 位有符号整型数的上溢界限)。经过调整，我使用%u，即将无符号整型数转为字符串形式，解决了该问题。通过这个遇到的问题，我了解到在代码算术逻辑中，考虑到数据类型的必要性。