通信、信工、安全专业单片机实验（二）

【实验要求】 通过串口与 LED 控制交互

说明：实验（二）的 3 道题目设计源代码分别在文件夹 exp2.1、exp2.2、exp2.3.1 中。此外，文件夹 exp2.3 是仅有呼吸效果的呼吸灯的源代码，未加入串口的控制。

【实验内容】

（1）在第一个实验中，实现了使用四个 LED 灯来显示计数的状态。在该实验中，使用串口显示当前的计数状态和信息，比如：是递增/递减计数、当前计数的值。（60 分）

注：在显示计数状态时，不能重复显示。

【实验结果】

（1）保存设计工程，以及设计源代码，源代码每行给出注释。

（2）实验报告，包含设计思路，设计过程等，遇到的问题，以能说明本次实验的内容即可。

（3）实验报告和设计工程保存在一个目录下，文件夹命名规则（学号+班级+名字）

注：对于安全专业学生，只要求实现书上按键扫描并通过串口显示的例子即可，实验结果要求同前。

1．设计思路

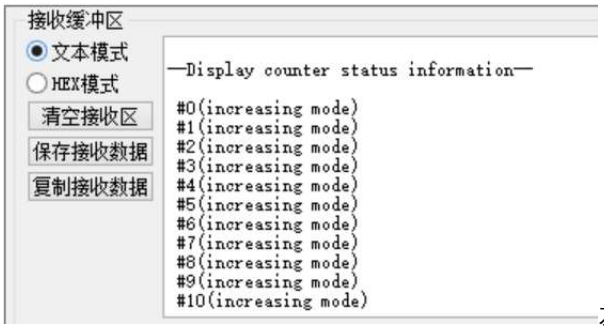
参照书本第 10 章的例子 10-1 和 10-2。在如何触发串口发送计数状态的问题上，我选择了参照例子 10-2 的方法，记录了旧的计数状态，在 while(1)的循环里判断只要计数值或者计数方向发生变化，就会发送计数状态。

2．调试过程及遇到问题的解决办法

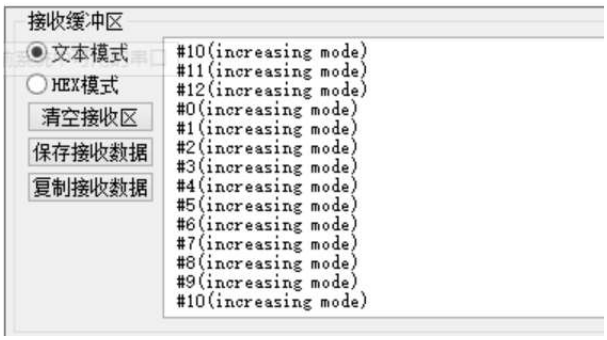
1) 关于定时器 0 和定时器 1 的协调问题

因为定时器 0 用来计数，计数频率的计算公式为:计数频率=[(IRC 时钟频率/clk_div 的分频系数)/12] /(65536-[RL_TH0,RL_TL0])，定时器 1 用来作为波特率时钟，[RL_TH1,RL_TL1]=65536-SYSclk/(串口 1 的波特率 x4)，现在串口 1 的波特率需要为 115200，综合考虑之下，选择了 IRC 时钟频率=11.0592MHz，CLK_DIV=0x03;//分频系数为 8，SYSclk=主时钟频率/8，这样使得[RL_TH0,RL_TL0]=0，则 13 进制计数器的计数频率=1.75Hz，[RL_TH1,RL_TL1] =65536-3=65533。

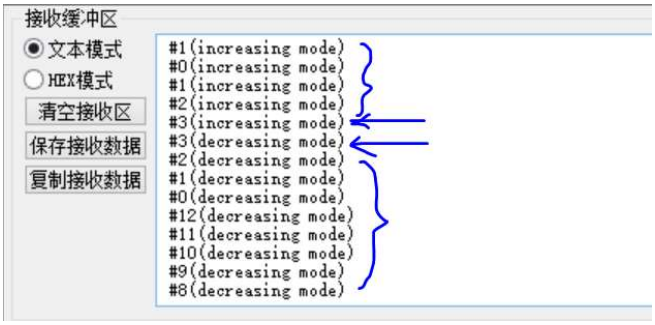
2．实验结果：成功实现所有功能。单片机主频率为 11.0592MHz。



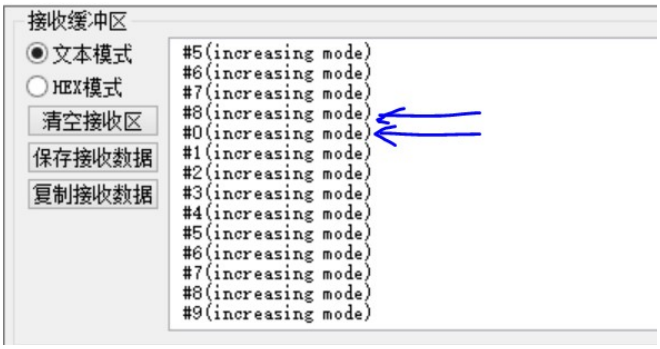
左边 2 幅图显示的是连续递增计数



下图显示的是 INT1 改变计数方向后显示的计数状态



下图显示的是 INT0 进行计数清零后显示的计数状态



4．源代码及注释

```
#include "STC15F2K60S2.H"
#define TIMS 0//声明定时器 0 的计数初值
#define FOSC 11059200L//声明单片机的工作频率
#define BAUD 115200//声明串口 1 的波特率

//IRC=11.0592MHz
bit busy=0; //声明 bit 型变量
xdata char menu[]={"\r\n--Display counter status information--\r\n"}; //声明字符数组 menu

int i_old=0,i=0;
int flag_old=0,flag=0;
int j=0;

void SendData(unsigned char dat) //声明 SendData 函数
```

```
{
    while(busy);//判断是否发送完，没有则等待
    SBUF=dat;//否则，将数据 dat 写入 SBUF 寄存器
    busy=1;//将 busy 标志置 1
}
void SendString(char *s) //声明 SendString 函数
{
    while(*s!='\0')//判断是否是字符串的结尾
        SendData(*s++);//如果没有结束，调用 SendData 发送数据
}
void SendStatus()//声明 SendStatus 函数
{
    SendString("\r\n #");//发送字符串信息
    if (i<10)//如果计数值小于 10，即 0~9
        SendData(i+0x30);//转化为对应的 ASCII 码，调用 SendData 发送
    else if(i==10) //如果计数值为 10
        SendString("10");//调用 SendString 函数，发送字符串 10
    else if(i==11) //如果计数值为 11
        SendString("11");//调用 SendString 函数，发送字符串 11
    else//如果计数值为 12
        SendString("12");//调用 SendString 函数，发送字符串 12
    if(flag==0)//如果 flag=0，即为递增计数
        SendString("(increasing mode)");//显示递增计数的信息
    else//如果 flag=1，即为递减计数
        SendString("(decreasing mode)"); //显示递减计数的信息
}
void uart1() interrupt 4//声明串口 1 中断服务程序 uart1
{
    if(RI)//通过 RI 标志，判断是否接收到数据
        RI=0;//如果 RI 为 1，则软件清零 RI
    if(TI) //通过 TI 标志，判断是否发送完数据
        TI=0; //如果 TI 为 1，则软件清零 TI
      busy=0;//将 busy 标志清零
}
void timer_0() interrupt 1//定时器 0 中断服务程序实现自动计数
{
            if(j==0)
            {
            if(flag==0) //由外部中断 1 的服务程序改变 flag，以实现递增或递减计数
                i=i+1;
          else
                i=i-1;
            }
            if(i==13) i=0;
```

```
if(i<0) i=12;
if(i==0)
{
	P17=1;
	P16=1;
	P47=1;
	P46=1;
}
else if(i==1)
{
	P17=0;
	P16=1;
	P47=1;
	P46=1;
}
else if(i==2)
{
	P17=1;
	P16=0;
	P47=1;
	P46=1;
}
else if(i==3)
{
	P17=0;
	P16=0;
	P47=1;
	P46=1;
}
else if(i==4)
{
	P17=1;
	P16=1;
	P47=0;
	P46=1;
}
else if(i==5)
{
	P17=0;
	P16=1;
	P47=0;
	P46=1;
}
else if(i==6)
```

```
{
	P17=1;
	P16=0;
	P47=0;
	P46=1;
}
else if(i==7)
{
	P17=0;
	P16=0;
	P47=0;
	P46=1;
}
else if(i==8)
{
	P17=1;
	P16=1;
	P47=1;
	P46=0;
}
else if(i==9)
{
	P17=0;
	P16=1;
	P47=1;
	P46=0;
}
else if(i==10)
{
	P17=1;
	P16=0;
	P47=1;
	P46=0;
}
else if(i==11)
{
	P17=0;
	P16=0;
	P47=1;
	P46=0;
}
else if(i==12)
{
	P17=1;
```

```
                P16=1;
                P47=0;
                P46=0;
            }
            else ;
j=0;
}

servivce_int0() interrupt 0//由外部中断 0 的服务程序计数清 0
{
    j=1; //使得定时器 0 的中断服务程序跳过开头的改变计数值的语句，先显示 0 值
    i=0;
}
servivce_int1() interrupt 2//由外部中断 1 的中断服务程序对标志变量 flag 取反，来改变计数方向
{
    flag=!flag;
}
void main()
{
    //初始化、配置 led 灯及其引脚驱动模式
    P1M0&=0x3F;
    P1M1&=0x3F;
    P4M0&=0x3F;
    P4M1&=0x3F; //配置引脚驱动模式为准双向
    P17=1;
    P16=1;
    P47=1;
    P46=1;
    //初始化、配置定时器 1
    CLK_DIV=0x03;//分频，SYSclk=主时钟频率/8
    TL0=TIMS;
    TH0=TIMS>>8; //装入计数初值
    AUXR=0x40; //最高位置 0，SYSclk/12 作定时器 0 时钟，次高位置 1，SYSclk 不分频，作定时器 1 时钟
    //AUXR2&=0xFE;
    TMOD=0x00; //设置 GATE=0，定时器 0 为模式 0（16 位自动重加载模式），定时器 1 为模式 0（16 位自动重加载模式），作为串口 1 的波特率时钟
    TR0=1; //设置 TR0=1，启动定时器 0
    ET0=1; //允许定时器 0 中断


    TL1=(65536-((FOSC/8/4)/BAUD));
    TH1=(65536-((FOSC/8/4)/BAUD))>>8; //装入计数初值的低 8 位和高 8 位
```

```
    TR1=1; //设置 TR0=1，启动定时器 1


    IT0=1; //只允许 INT0 下降沿触发
    IT1=1; //只允许 INT1 下降沿触发
    EX0=1; //允许外部中断 0
    EX1=1; //允许外部中断 1

    SCON=0x50; //串口 1 模式 1，使能串行接收，禁止多机
    ES=1;//允许串口 1 中断
    EA=1; //CPU 允许中断
    SendString(&menu);//向串口助手的接收缓冲区发送 提示信息字符串
    SendStatus();//显示初始的计数状态为：计数值为 0，递增计数
    while(1)
    {
        if(i!=i_old||flag!=flag_old)//如果新的计数值和旧的计数值不一样，或者新的计数方向和旧的计数方向不一样，则发送出改变后的计数信息。
        {
            i_old=i;//把新的计数值保存为旧的计数值
            flag_old=flag; //把新的计数方向保存为旧的计数方向
            SendStatus();//在串口调试界面中显示当前变化的计数值及递增/递减计数状态
        }
    }
}
```

（2）在第一个实验中，使用按键控制递增/递减的方向，在该实验中，自定义串口通信的数据格式，实现通过串口来改变计数器的计数方向。（25 分）

1．设计思路

参照书本第 10 章的例子 10-1，将原来 led 取反改为 flag 取特定的值即可。我设定，如果通过串口接收到的数据为 1，则正向计数，为 2，则反向计数。在下载了程序之后，断电再重新启动，就可以看到程序的正常运行结果。

2．调试过程及遇到问题的解决办法

因为沿用了第 1 题的代码，以及在例子 10-1 的思路上，主要在主程序的 while 循环里改动即可，所以实验过程比较顺畅。

3．实验结果：成功实现通过串口来改变计数器的计数方向。主频率为 **11.0592MHz**。

4．源代码及注释

```
#include "STC15F2K60S2.H"
#define TIMS 0//声明定时器 0 的计数初值
#define FOSC 11059200L//声明单片机的工作频率
#define BAUD 115200//声明串口 1 的波特率

//IRC=11.0592MHz
bit busy=0; //声明 bit 型变量
xdata char menu[]={ "\r\n------main menu----------"//声明字符数组 menu
                    "\r\n        input 1:   Count increasingly "
                    "\r\n        input 2:   Count decreasingly   "
                        "\r\n        other   :   Exit Program"
                        "\r\n------end menu----------"
                  };
int i_old=0,i=0;
int flag_old=0,flag=0;
int j=0;

void SendData(unsigned char dat) //声明 SendData 函数
{
    while(busy);//判断是否发送完，没有则等待
    SBUF=dat;//否则，将数据 dat 写入 SBUF 寄存器
    busy=1;//将 busy 标志置 1
}
void SendString(char *s) //声明 SendString 函数
{
    while(*s!='\0')//判断是否是字符串的结尾
        SendData(*s++);//如果没有结束，调用 SendData 发送数据
}
void uart1() interrupt 4//声明串口 1 中断服务程序 uart1
{
    if(RI)//通过 RI 标志，判断是否接收到数据
```

```
        RI=0;//如果 RI 为 1，则软件清零 RI
    if(TI) //通过 TI 标志，判断是否发送完数据
        TI=0; //如果 TI 为 1，则软件清零 TI
      busy=0;//将 busy 标志清零
}
void timer_0() interrupt 1//定时器 0 中断服务程序实现自动计数
{
            if(j==0)
            {
            if(flag==0) //由外部中断 1 的服务程序改变 flag，在这里实现递增或递减计数
                i=i+1;
          else
                i=i-1;
            }
            if(i==13) i=0;
            if(i<0) i=12;
            if(i==0)
            {
                P17=1;
                P16=1;
                P47=1;
                P46=1;
            }
            else if(i==1)
            {
                P17=0;
                P16=1;
                P47=1;
                P46=1;
            }
            else if(i==2)
            {
                P17=1;
                P16=0;
                P47=1;
                P46=1;
            }
            else if(i==3)
            {
                P17=0;
                P16=0;
                P47=1;
                P46=1;
            }
```

```
else if(i==4)
{
	P17=1;
	P16=1;
	P47=0;
	P46=1;
}
else if(i==5)
{
	P17=0;
	P16=1;
	P47=0;
	P46=1;
}
else if(i==6)
{
	P17=1;
	P16=0;
	P47=0;
	P46=1;
}
else if(i==7)
{
	P17=0;
	P16=0;
	P47=0;
	P46=1;
}
else if(i==8)
{
	P17=1;
	P16=1;
	P47=1;
	P46=0;
}
else if(i==9)
{
	P17=0;
	P16=1;
	P47=1;
	P46=0;
}
else if(i==10)
{
```

```
                P17=1;
                P16=0;
                P47=1;
                P46=0;
            }
            else if(i==11)
            {
                P17=0;
                P16=0;
                P47=1;
                P46=0;
            }
            else if(i==12)
            {
                P17=1;
                P16=1;
                P47=0;
                P46=0;
            }
            else ;
j=0;
}

servivce_int0() interrupt 0//由外部中断 0 的服务程序计数清 0
{
    j=1; //使得定时器 0 的中断服务程序跳过开头的改变计数值的语句，先显示 0 值
    i=0;
}
servivce_int1() interrupt 2//由外部中断 1 的服务程序改变 flag，来改变计数方向
{
    flag=!flag;
}
void main()
{
    //初始化、配置 led 灯及其引脚驱动模式
    P1M0&=0x3F;
    P1M1&=0x3F;
    P4M0&=0x3F;
    P4M1&=0x3F; //配置引脚驱动模式为准双向
    P17=1;
    P16=1;
    P47=1;
    P46=1;
    //初始化、配置定时器 1
```

```
    CLK_DIV=0x03;//分频，SYSclk=主时钟频率/8
    TL0=TIMS;
    TH0=TIMS>>8; //装入计数初值
    AUXR=0x40; //最高位置 0，SYSclk/12 作定时器 0 时钟，次高位置 1，SYSclk 不分频，作定时器 1 时钟
    //AUXR2&=0xFE;
    TMOD=0x00; //设置 GATE=0，定时器 0 为模式 0（16 位自动重加载模式），定时器 1 为模式 0（16 位自动重加载模式），作为串口 1 的波特率时钟
    TR0=1; //设置 TR0=1，启动定时器 0
    ET0=1; //允许定时器 0 中断


    TL1=(65536-((FOSC/8/4)/BAUD));
    TH1=(65536-((FOSC/8/4)/BAUD))>>8; //装入计数初值的低 8 位和高 8 位
    TR1=1; //设置 TR0=1，启动定时器 1


    IT0=1; //只允许 INT0 下降沿触发
    IT1=1; //只允许 INT1 下降沿触发
    EX0=1; //允许外部中断 0
    EX1=1; //允许外部中断 1

    SCON=0x50; //串口 1 模式 1，使能串行接收，禁止多机
    ES=1;//允许串口 1 中断
    EA=1; //CPU 允许中断
    SendString(&menu);//向串口助手的接收缓冲区发送 操作菜单字符串信息
    while(1)
    {
        if(RI==1)//如果接收到上位机发送的数据
         {
            c=SBUF;//从 SBUF 缓冲区读数据到变量 c
            if(c==0x31)//判断如果接受到的数据是字符 1
                flag=0;//正向计数
            else if(c==0x32) //判断如果接受到的数据是字符 2
                flag=1; //反向计数
            else //对于其它输入
            {
                SendString("\r\n Exit Program");//串口上打印 Exit Program 信息
            }
        }
    }
}
```

（3）通过串口，设置参数，实现可变“呼吸灯”的功能，比如频率、亮度等（15 分）。

1．设计思路

参照书本第 9 章的例子 9-4 的控制占空比的思想，以及我之前在 DSP 课程中做的实验——用定时器产生 PWM 信号控制电机的程序。

经过不断尝试，我的参数为：

主频率为 11.0592MHz

一次呼或者吸维持 1/3.5 秒

每个亮度维持 C=1/3.5/10 秒

每次计数间隔 T=C/10

定时器频率=350Hz

定时器初值 TIMES=64219

后来在调试过程中，又把定时器初值 TIMES 改为了 64000，

用串口改变呼吸灯的呼吸频率，只要改变定时器的初值即可。

2．调试过程及遇到问题的解决办法

这个实验是到目前为止，我做的单片机实验中最耗时的，遇到的问题也很多。写出来的程序虽然看上去不长，但是调试过程费时费力。

1) 开始我试图通过硬件在线调试，使用“run to cursor line”进入中断，查看关键变量的变化情况，但是我无法进入指定的中断服务程序中的语句，一选择“run to cursor line”，程序的调试按钮就变灰，卡住了。后来经过老师的点拨，我觉得是我的中断的初始化设置有问题，后来修改主程序里的关于中断的寄存器设置后，这个问题就解决了。

2) 后来是我写的呼吸灯的产生程序的逻辑有问题，导致呼吸灯呼吸不稳定、“频闪”现象，后来因为这个呼吸由暗变亮，又由亮变暗的逻辑比较复杂，我一边硬件在线调试，一边对我的程序逻辑进行检查和改正。为了验证我的呼吸灯产生程序的逻辑，我特意把呼吸的频率变慢了，通过看一个呼吸周期中，亮的时间的长短，来检查实际效果是否符合我的预期效果。这样调试了 1~2 个小时，我的呼吸灯的逻辑就捋通了，呼吸也更加顺畅和平稳了。

3．实验结果：成功实现用串口改变呼吸灯的呼吸频率。本程序的主频率为 **11.0592MHz**，将呼吸频率分为 3 档，慢中快，当串口发送数据为 1 时，呼吸速度慢，当串口发送数据为 2 时，呼吸速度中等，当串口发送数据为 3 时，呼吸速度最快。

4．源代码及注释

```
#include "STC15F2K60S2.H"
#define FOSC    11059200L//声明单片机的工作频率
#define BAUD 115200//声明串口 1 的波特率

sfr TH2      =0xD6;
sfr TL2      =0xD7;
bit busy=0; //声明 bit 型变量
int uN=0;
//设置初始的占空比为 0%，其中 uN=0、1、2~9 分别对应占空比为 0%、10%、20%~90%
int nCount=0;//  设置初始的计数值为 0
int dir=0;//设置初始的呼吸方向是由亮变暗
int start=1;//这是一个开关变量，用来控制呼吸方向改变的时刻和条件
unsigned int TIMS=64000;//计数器 0 的初始值，用来调整呼吸灯的呼吸频率

xdata char menu[]={"\r\n------main menu----------------"
                                "\r\n        input 1:   breathe slowly "
                                "\r\n        input 2:   breathe faster "
                                "\r\n        input 3:   breathe fastestly "
                                "\r\n        other   :   Exit Program"
                                "\r\n------end menu----------------"
                  };//声明字符数组 menu

void SendData(unsigned char dat) //声明 SendData 函数
{
      while(busy); //判断是否发送完，没有则等待
      SBUF=dat; //否则，将数据 dat 写入 SBUF 寄存器
      busy=1; //将 busy 标志置 1
}
void SendString(char *s) //声明 SendString 函数
{
      while(*s!='\0') //判断是否是字符串的结尾
            SendData(*s++);//如果没有结束，调用 SendData 发送数据
}
void uart1() interrupt 4//声明串口 1 中断服务程序 uart1
{
      if(RI) //通过 RI 标志，判断是否接收到数据
            RI=0; //如果 RI 为 1，则软件清零 RI
      if(TI) //通过 TI 标志，判断是否发送完数据
            TI=0; //如果 TI 为 1，则软件清零 TI
         busy=0; //将 busy 标志清零
}

void timer_0() interrupt 1//定时器 0 中断服务程序改变变量 nCount 的值。
```

```
{
    if(dir==0)//判断当前的呼吸方向，是由暗变亮（占空比由小变大）还是由亮变暗（占空比由大变小），dir=0 时，占空比由大变小
    {
            P47=( nCount<uN )?0:1;
           //当前计数值小于占空比的时候引脚电平为 0，否则为 1
            nCount++; nCount=nCount%10;//计数值 nCount 的变化范围为 0~9
            if(nCount==0)
            {
                uN++;//当计数值恢复到 0 时，占空比增加 10%
                uN=uN%10;//占空比 uN 的变化范围为 0~9
                start=0;//从第 1 轮 nCount 循环结束起，start 开关变量开启，允许等待符合条件时改变呼吸的方向
            }
    }
     else// dir=1 时，占空比由小变大
     {
            P47=( nCount<uN )?1:0;
           //当前计数值小于占空比的时候引脚电平为 1，否则为 0
            nCount++; nCount=nCount%10;
            if(nCount==0)
            {
                uN++;
                uN=uN%10;
                start=0; //从第 1 轮 nCount 循环结束起，start 开关变量开启，允许等待符合条件时改变呼吸的方向
             }
    }
     if(start!=1&&uN==0)
     {
        dir=!dir;//改变呼吸的方向
        start=1; //start 开关变量关闭，禁止在第 1 轮 nCount 计数循环期间改变呼吸的方向，因为此期间的 uN=0，但是却不希望呼吸方向改变。
     }
}

void main()
{
    unsigned char c;//声明字符型变量
    P4M0&=0x7F;
    P4M1&=0x7F; //配置引脚 P47 的驱动模式为准双向
    P47=1;
    SCON=0x50;//串口 1 模式 1，使能串行接收，禁止多机
```

```
    CLK_DIV=0x01;//主时钟 2 分频
    TL0=TIMS;
    TH0=TIMS>>8; //装入计数初值
    TL2=(65536-((FOSC/2/4)/BAUD));
    TH2=(65536-((FOSC/2/4)/BAUD))>>8; //装入计数初值的低 8 位和高 8 位

    AUXR=0x15;//允许定时器 2，不分频，选择定时器 2 作为波特率发生器
    TMOD=0x00; //设置 GATE=0，定时器 0 模式 0（16 位自动重加载模式）

    TR0=1; //设置 TR0=1，启动定时器 0
    ET0=1; //允许定时器 0 中断
    ES=1; //允许串口 1 中断
    EA=1; //CPU 允许中断
    SendString(&menu); //向串口助手的接收缓冲区发送 操作菜单的内容
    while(1)
    {
        if(RI==1) //如果接收到上位机发送的数据
         {
            c=SBUF; //从 SBUF 缓冲区读数据到变量 c
            if(c==0x31) //判断如果接受到的数据是字符 1，慢速呼吸
            {
                TIMS=64000;//设置定时器的计数初值
                TL0=TIMS;
                TH0=TIMS>>8;//将计数初值装入 TH0 和 TL0 寄存器
            }
            else if(c==0x32) //判断如果接受到的数据是字符 2，中速呼吸
            {
                TIMS=64500; //设置定时器的计数初值
                TL0=TIMS;
                TH0=TIMS>>8; //将计数初值装入 TH0 和 TL0 寄存器
            }
            else if(c==0x33) //判断如果接受到的数据是字符 3，快速呼吸
            {
                TIMS=65000; //设置定时器的计数初值
                TL0=TIMS;
                TH0=TIMS>>8; //将计数初值装入 TH0 和 TL0 寄存器
            }
            else //对于其他任何输入
            {
                SendString("\r\n Exit Program"); //串口上打印 Exit Program 信息
            }
        }
    }
}
```