学生学号	0122015710114	实验课成绩	

武汉理工大学

学 生 实 验 报 告 书

实验课程名称	单片机及嵌入式系统原理
开 课 学 院	信息工程学院
指导教师姓名	周伟
学 生 姓 名	胡姗
学生专业班级	信息 2001

2022 -- 2023 学年 第 一 学期

实验教学管理基本规范

实验是培养学生动手能力、分析解决问题能力的重要环节；实验报告是反映实验教学水平与质量的重要依据。为加强实验过程管理，改革实验成绩考核方法，改善实验教学效果，提高学生质量，特制定实验教学管理基本规范。

1、 本规范适用于理工科类专业实验课程，文、经、管、计算机类实验课程可根据具体情况参照执行或暂不执行。

2、 每门实验课程一般会包括许多实验项目，除非常简单的验证演示性实验项目可以不写实验报告外，其他实验项目均应按本格式完成实验报告。

3、 实验报告应由实验预习、实验过程、结果分析三大部分组成。每部分均在实验成绩中占一定比例。各部分成绩的观测点、考核目标、所占比例可参考附表执行。各专业也可以根据具体情况，调整考核内容和评分标准。

4、 学生必须在完成实验预习内容的前提下进行实验。教师要在实验过程中抽查学生预习情况，在学生离开实验室前，检查学生实验操作和记录情况，并在实验报告第二部分教师签字栏签名，以确保实验记录的真实性。

5、 教师应及时评阅学生的实验报告并给出各实验项目成绩，完整保存实验报告。在完成所有实验项目后，教师应按学生姓名将批改好的各实验项目实验报告装订成册，构成该实验课程总报告，按班级交课程承担单位（实验中心或实验室）保管存档。

6、 实验课程成绩按其类型采取百分制或优、良、中、及格和不及格五级评定。

附表：实验考核参考内容及标准

	观测点	考核目标	成绩组成
实验预习	1. 预习报告 2. 提问 3. 对于设计型实验，着重考查设计方案的科学性、可行性和创新性	对实验目的和基本原理的认识程度，对实验方案的设计能力	20%
实验过程	1. 是否按时参加实验 2. 对实验过程的熟悉程度 3. 对基本操作的规范程度 4. 对突发事件的应急处理能力 5. 实验原始记录的完整程度 6. 同学之间的团结协作精神	着重考查学生的实验态度、基本操作技能；严谨的治学态度、团结协作精神	30%
结果分析	1. 所分析结果是否用原始记录数据 2. 计算结果是否正确 3. 实验结果分析是否合理 4. 对于综合实验，各项内容之间是否有分析、比较与判断等	考查学生对实验数据处理和现象分析的能力；对专业知识的综合应用能力；事实求实的精神	50%

实验课程名称：单片机及嵌入式系统原理

实验项目名称	双机通信实验			实验成绩	
实 验 者	胡姗	专业班级	信息 2001	组　　别	无
同 组 者	无			实验日期	2022 年 12 月 3 日

第一部分：实验预习报告（包括实验目的、意义，实验基本原理与方法，主要仪器设备及耗材，实验方案与技术路线等）

一、实验目的

1、理解 UART 串口通信的基本原理和通信过程。

2、掌握通信的底层操作原理。

3、学会通过配置寄存器，实现串口通信的基本操作过程。

二、实验基本原理

单片机的 P3.0 和 P3.1 是专门用来进行 UART 串行通信的，P3.0 叫做 RXD，P3.1 脚叫做 TXD；RXD 是串行接收引脚，TXD 是串行发送引脚。在 UART 通信时，低位先发，高位后发，TXD 首先拉低电平，持续 1/baud 时间(baud 是发送二进制数据位的速率),发送一位数据，持续 1/baud，再发送一位数据，直到全部发送完毕，拉高 TXD 电平，发送结束。

单片机的 P2 口与按键相连，当按键按下时，相应的 KeyIn 口为低电平，当按键松开时，相应的 KeyIn 口为高电平。扫描按键时，每次让矩阵按键的一个 KeyOut 输出低电平，其他三个输出高电平，再判断所有 KeyIn 的状态，然后再让下一个 KeyOut 输出低电平，其他三个输出高电平，判断所有 KeyIn 电平状态，不断循环，就可以判断哪个按键按下。

三、实验内容

使用变量标识半双工的通信状态，开始时只允许 PC 向单片机发送数据，单片机接收到数据后，才能向 PC 发送数据，然后以此循环。PC 发送给单片机的数据通过 UART 串口收集，然后在液晶屏上显示出来。单片机给 PC 发送数据时，使用按键输入需要发送的数据，并在液晶屏上显示，然后按下回车发送数据。当单片机接收到 PC 发送的数据时，向 PC 发回接受确认。

第二部分：实验过程记录（可加页）（包括实验原始数据记录，实验现象记录，实验过程发现的问题等）

首先当程序运行时使能总中断，调用函数使定时器 T0 定时 1ms，定时器 T1 使用模式 2 自动重装载模式，确定 UART 波特率为 9600，初始化液晶屏后，默认通信状态标识 step=0，启动 UART 驱动，等待 PC 向单片机发送数据，若接收到 PC数据 step=1，启动按键矩阵驱动，等待单片机向 PC 发送数据，发送后 step=0，然后不断循环。

```
void main()
{
 EA=1;
 ConfigTimer0(1);
 ConfigUART(9600);
 InitLcd1602();
 while(1)
 {
 if(step==0)
 {
 step=UartDriver();
 }
 else if(step==1)
 {
 step=KeyDriver();
 }
 }
}
```

定时器 T0 每次中断时，给 T0 重新赋值，调用函数监控 UART 状态，检测按键状态。

```
void InterruptTimer0() interrupt 1
{

    TH0=T0RH;
    TL0=T0RL;
    UartRxMonitor(1);
    KeyScan();
}
```

如果监控到 UART 的 RXD 空闲 30ms，一帧数据接收完毕，接收完成标识 flagFrame=1。

```
void UartRxMonitor(unsigned char ms)
{
    static unsigned char cntbkp=0;
    static unsigned char idletmr=0;
    if(cntRxd>0)
    {
```

```
            if(cntbkp!=cntRxd)
            {
                cntbkp=cntRxd;
                idletmr=0;
            }
            else
            {
                if(idletmr<30)
                {
                    idletmr+=ms;
                    if(idletmr>=30)
                    {
                        flagFrame=1;
                    }
                }
            }
        }
        else
        {
            cntbkp=0;
        }
}
```

启动 UART 驱动时，如果 flagFrame=1，给 flagFrame 赋值 0，并将接收到的一帧数据显示在液晶屏第一行上，然后将 step 赋值 1，返回接收确认。

```
unsigned char UartDriver()
{
    unsigned char len;
    unsigned char pdata buf[40];
    if(flagFrame)
    {
        flagFrame=0;
        len=UartRead(buf,sizeof(buf));
        UartAction(buf,len);
        return 1;
    }
    else
    {
        return 0;
    }
}
void UartAction(unsigned char *buf,unsigned char len)
{
    buf[len]='\0';
    LcdShowStr(0,0,buf);
```

```
    if(len<16)
    {
        LcdAreaClear(len,0,16-len);
    }
    UartWrite("GetInformation",sizeof("GetInformation"));
}
```

当按键矩阵驱动启动时，按下的数字在液晶屏第二行显示，按下回车后将这些数字发送，发送完成给 step 赋值 0。

```
unsigned char KeyDriver()
{
    unsigned char i,j;
    unsigned char step;
    static unsigned char pdata backup[4][3]={
        {1,1,1},{1,1,1},{1,1,1},{1,1,1}
    };
    for(i=0;i<4;i++)
    {
        for(j=0;j<3;j++)
        {
            if(backup[i][j]!=KeySta[i][j])
            {
                if(backup[i][j]!=0)
                {
                    step=KeyAction(KeyCodeMap[i][j]);
                }
                else{step=1;}
                backup[i][j]=KeySta[i][j];
            }
            else{step=1;}
        }
    }
    return step;
}
unsigned char KeyAction(unsigned char keycode)
{
    unsigned char flag;
    if((keycode>='0')&&(keycode<='9'))
    {
        NumKeyAction(keycode-'0');
        flag=1;
    }
    else if(keycode==0x0D)
    {
        unsigned char len;
```

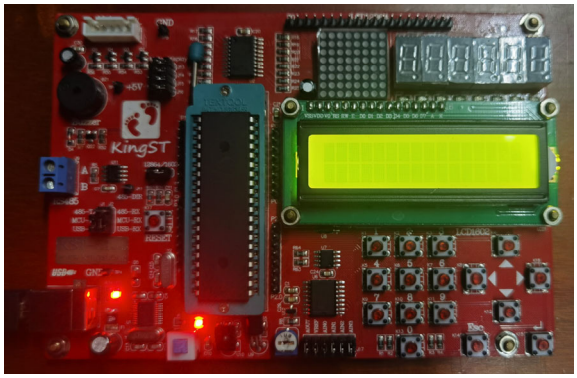
```
            unsigned char str[16];
            len=LongToString(str,num);
            UartWrite(str,len);
            Reset();
            LcdShowStr(16-sizeof("Sented"),1,"Sented");
            flag=0;
        }
        else if(keycode==0x1B)
        {
            Reset();
            LcdShowStr(15,1,"0");
            flag=1;
        }
        else{flag=1;}
        return flag;
}
```

教师签字_____

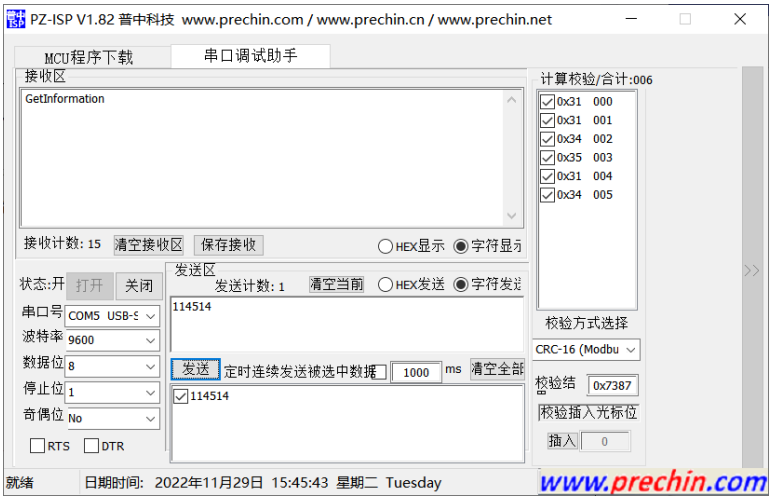
第三部分　结果与讨论（可加页）

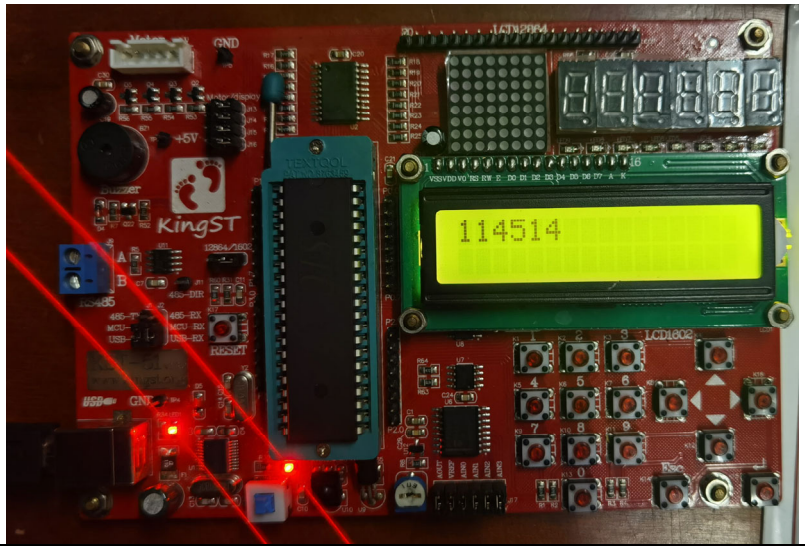
一、实验结果分析（包括数据处理、实验现象分析、影响因素讨论、综合分析和结论等）

最开始单片机液晶屏清空，只能 PC 给单片机发送数据，单片机按键按下不响应。

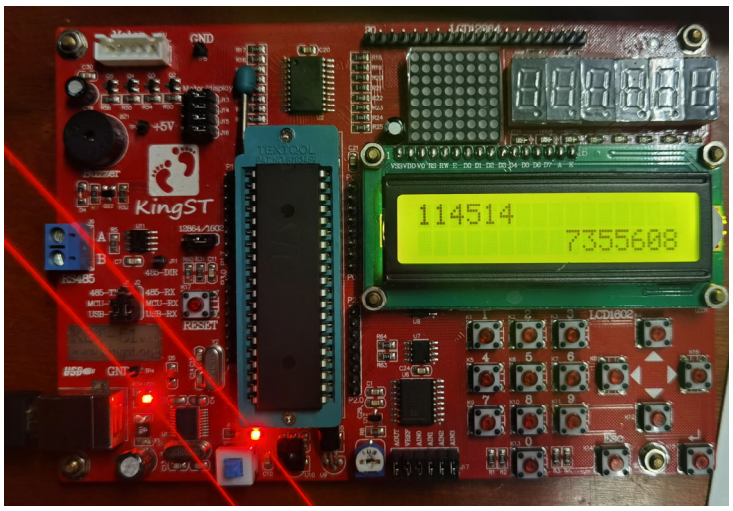


PC 给单片机发送数据，数据在液晶屏第一行显示，并且 PC 接收到单片机的确认

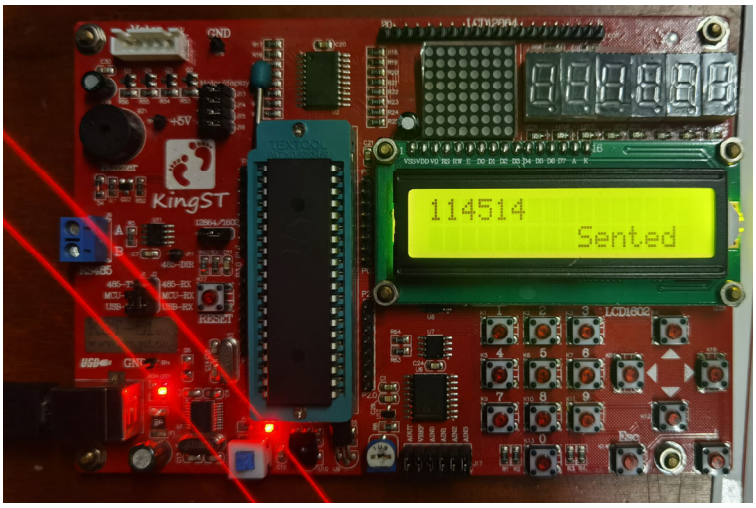


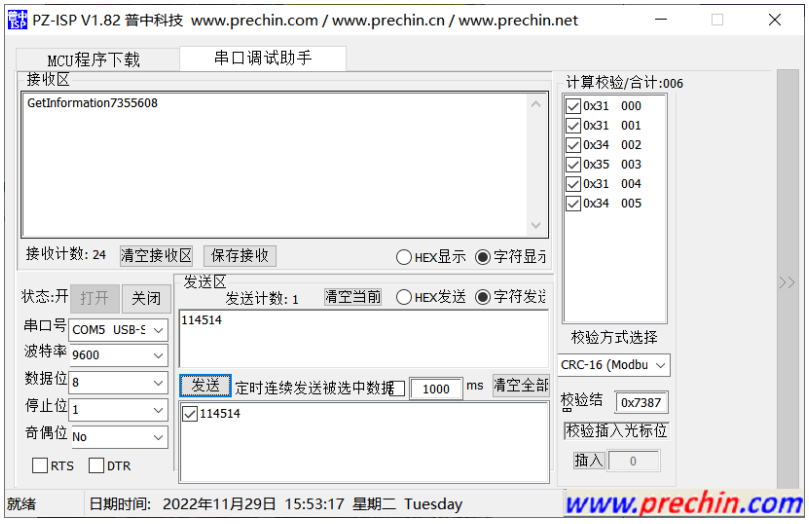


PC 给单片机成功发送数据后，只能单片机给 PC 发送数据，此时 PC 再次发送数据，单片机无响应。按下单片机按键，会在液晶屏第二行显示。



单片机按下回车键后，数据发送，并在液晶屏第二行显示 Sented，之后只能 PC 向单片机发送数据，按动单片机按键无响应。等到下一次单片机向 PC 发送信息时，需要按下 ESC 清空液晶屏第二行，才可用按键输入数据。





二、思考题

请总结单片机 UART 串口通信的实现流程。

答：首先确定波特率 baud，使用定时器 T1 作为波特率发生器，将串口控制寄存器 SCON 设置成模式 1，启动 UART 中断。使用定时器 T0 定时 1ms，每 1ms 检测一次 UART 总线状态，判断数据接收是否完成，接收完成时，清零接收中断标志位 RI，对数据进行下一步处理。如果发送数据，发送完成时，清零发送中断标志位 TI。

附录：单片机程序如下

Lcd1602.c 文件：

```
#include <reg52.h>
#define LCD1602_DB P0
sbit LCD1602_RS=P1^0;
sbit LCD1602_RW=P1^1;
sbit LCD1602_E=P1^5;
void LcdWaitReady()
{
    unsigned char sta;
    LCD1602_DB=0xFF;
    LCD1602_RS=0;
    LCD1602_RW=1;
    do
    {
        LCD1602_E=1;
        sta=LCD1602_DB;
        LCD1602_E=0;
    }while(sta&0x80);
}
void LcdWriteCmd(unsigned char cmd)
{
    LcdWaitReady();
    LCD1602_RS=0;
    LCD1602_RW=0;
    LCD1602_DB=cmd;
    LCD1602_E=1;
    LCD1602_E=0;
}
void LcdWriteDat(unsigned char dat)
{
    LcdWaitReady();
    LCD1602_RS=1;
    LCD1602_RW=0;
    LCD1602_DB=dat;
    LCD1602_E=1;
    LCD1602_E=0;
```

```
}
void LcdSetCursor(unsigned char x,unsigned char y)
{
	unsigned char addr;
	if(y==0)
	{
		addr=0x00+x;
	}
	else
	{
		addr=0x40+x;
	}
	LcdWriteCmd(addr|0x80);
}
void LcdShowStr(unsigned char x,unsigned char y,unsigned char *str)
{
	LcdSetCursor(x,y);
	while(*str!='\0')
	{
		LcdWriteDat(*str++);
	}
}
void LcdAreaClear(unsigned char x,unsigned char y,unsigned char len)
{
	LcdSetCursor(x,y);
	while(len--)
	{
		LcdWriteDat(' ');
	}
}
void InitLcd1602()
{
	LcdWriteCmd(0x38);
	LcdWriteCmd(0x0C);
	LcdWriteCmd(0x06);
	LcdWriteCmd(0x01);
}
```

Keyboard.c 文件

```
#include <reg52.h>
sbit KEY_IN_1=P2^4;
sbit KEY_IN_2=P2^5;
sbit KEY_IN_3=P2^6;
sbit KEY_OUT_1=P2^3;
sbit KEY_OUT_2=P2^2;
sbit KEY_OUT_3=P2^1;
sbit KEY_OUT_4=P2^0;
unsigned char code KeyCodeMap[4][3]={
    {'1','2','3'},
    {'4','5','6'},
    {'7','8','9'},
    {'0',0x1B,0x0D}
};
unsigned char pdata KeySta[4][3]={
    {1,1,1},{1,1,1},{1,1,1},{1,1,1}
};
extern unsigned char KeyAction(unsigned char keycode);
unsigned char KeyDriver()
{
    unsigned char i,j;
    unsigned char step;
    static unsigned char pdata backup[4][3]={
        {1,1,1},{1,1,1},{1,1,1},{1,1,1}
    };
    for(i=0;i<4;i++)
    {
        for(j=0;j<3;j++)
        {
            if(backup[i][j]!=KeySta[i][j])
            {
                if(backup[i][j]!=0)
                {
                    step=KeyAction(KeyCodeMap[i][j]);
                }
                else{step=1;}
                backup[i][j]=KeySta[i][j];
            }
```

```
                else{step=1;}
            }
        }
        return step;
}
void KeyScan()
{
        unsigned char i;
        static unsigned char keyout=0;
        static unsigned char keybuf[4][3]={
            {0xFF,0xFF,0xFF},{0xFF,0xFF,0xFF},
            {0xFF,0xFF,0xFF},{0xFF,0xFF,0xFF}
        };
        keybuf[keyout][0]=(keybuf[keyout][0]<<1)|KEY_IN_1;
        keybuf[keyout][1]=(keybuf[keyout][1]<<1)|KEY_IN_2;
        keybuf[keyout][2]=(keybuf[keyout][2]<<1)|KEY_IN_3;
        for(i=0;i<3;i++)
        {
            if((keybuf[keyout][i]&0x0F)==0x00)
            {
                KeySta[keyout][i]=0;
            }
            else if((keybuf[keyout][i]&0x0F)==0x0F)
            {
                KeySta[keyout][i]=1;
            }
        }
        keyout++;
        keyout&=0x03;
        switch(keyout)
        {
            case 0:KEY_OUT_4=1;KEY_OUT_1=0;break;
            case 1:KEY_OUT_1=1;KEY_OUT_2=0;break;
            case 2:KEY_OUT_2=1;KEY_OUT_3=0;break;
            case 3:KEY_OUT_3=1;KEY_OUT_4=0;break;
            default: break;
        }
}
```

Uart.c 文件

```
#include <reg52.h>
bit flagFrame=0;
bit flagTxd=0;
unsigned char cntRxd=0;
unsigned char pdata bufRxd[64];
extern void UartAction(unsigned char *buf,unsigned char len);
void ConfigUART(unsigned int baud)
{
    SCON=0x50;
    TMOD&=0x0F;
    TMOD|=0x20;
    TH1=256-(11059200/12/32)/baud;
    TL1=TH1;
    ET1=0;
    ES=1;
    TR1=1;
}
void UartWrite(unsigned char *buf,unsigned char len)
{
    while(len--)
    {
        flagTxd=0;
        SBUF=*buf++;
        while(!flagTxd);
    }
}
unsigned char UartRead(unsigned char *buf,unsigned char len)
{
    unsigned char i;
    if(len>cntRxd)
    {
        len=cntRxd;
    }
    for(i=0;i<len;i++)
    {
        *buf++=bufRxd[i];
    }
    cntRxd=0;
```

```
    return len;
}
void UartRxMonitor(unsigned char ms)
{
    static unsigned char cntbkp=0;
    static unsigned char idletmr=0;
    if(cntRxd>0)
    {
        if(cntbkp!=cntRxd)
        {
            cntbkp=cntRxd;
            idletmr=0;
        }
        else
        {
            if(idletmr<30)
            {
                idletmr+=ms;
                if(idletmr>=30)
                {
                    flagFrame=1;
                }
            }
        }
    }
    else
    {
        cntbkp=0;
    }
}
unsigned char UartDriver()
{
    unsigned char len;
    unsigned char pdata buf[40];
    if(flagFrame)
    {
        flagFrame=0;
        len=UartRead(buf,sizeof(buf));
        UartAction(buf,len);
```

```
        return 1;
    }
    else
    {
        return 0;
    }
}
void InterruptUART() interrupt 4
{
    if(RI)
    {
        RI=0;
        if(cntRxd<sizeof(bufRxd))
        {
            bufRxd[cntRxd++]=SBUF;
        }
    }
    if(TI)
    {
        TI=0;
        flagTxd=1;
    }
}
```

Main.c 文件

```
#include <reg52.h>
unsigned char T0RH=0;
unsigned char T0RL=0;
unsigned char step=0;
unsigned long num=0;
void ConfigTimer0(unsigned int ms);
extern unsigned char UartDriver();
extern unsigned char KeyDriver();
extern void KeyScan();
extern void ConfigUART(unsigned int baud);
extern void UartRxMonitor(unsigned char ms);
extern void UartWrite(unsigned char *buf,unsigned char len);
extern void InitLcd1602();
extern void LcdShowStr(unsigned char x,unsigned char y,unsigned char *str);
```

```
extern void LcdAreaClear(unsigned char x,unsigned char y,unsigned char len);
void main()
{
	EA=1;
	ConfigTimer0(1);
	ConfigUART(9600);
	InitLcd1602();
	while(1)
	{
		if(step==0)
		{
			step=UartDriver();
		}
		else if(step==1)
		{
			step=KeyDriver();
		}
	}
}
void UartAction(unsigned char *buf,unsigned char len)
{
	buf[len]='\0';
	LcdShowStr(0,0,buf);
	if(len<16)
	{
		LcdAreaClear(len,0,16-len);
	}
	UartWrite("GetInformation",sizeof("GetInformation"));
}
void Reset()
{
	num=0;
	LcdAreaClear(0,1,16);
}
unsigned char LongToString(unsigned char *str,unsigned long dat)
{
	unsigned char i=0;
	unsigned char len=0;
	unsigned char buf[16];
```

```
    do
    {
        buf[i++]=dat%10;
        dat/=10;
    }while(dat>0);
    len+=i;
    while(i-->0)
    {
        *str++=buf[i]+'0';
    }
    *str++='\r';
    *str='\n';
    len=len+2;
    return len;
}
void NumKeyAction(unsigned char n)
{
    unsigned char len;
    unsigned char str[16];
    num=num*10+n;
    len=LongToString(str,num);
    len=len-2;
    LcdShowStr(16-len,1,str);
}
unsigned char KeyAction(unsigned char keycode)
{
    unsigned char flag;
    if((keycode>='0')&&(keycode<='9'))
    {
        NumKeyAction(keycode-'0');
        flag=1;
    }
    else if(keycode==0x0D)
    {
        unsigned char len;
        unsigned char str[16];
        len=LongToString(str,num);
        UartWrite(str,len);
        Reset();
```

```
            LcdShowStr(16-sizeof("Sented"),1,"Sented");
            flag=0;
        }
        else if(keycode==0x1B)
        {
            Reset();
            LcdShowStr(15,1,"0");
            flag=1;
        }
        else{flag=1;}
        return flag;
}
void ConfigTimer0(unsigned int ms)
{
        unsigned long tmp;
        tmp=11059200/12;
        tmp=(tmp*ms)/1000;
        tmp=65536-tmp;
        tmp=tmp+33;
        T0RH=(unsigned char)(tmp>>8);
        T0RL=(unsigned char)tmp;
        TMOD&=0xF0;
        TMOD|=0x01;
        TH0=T0RH;
        TL0=T0RL;
        ET0=1;
        TR0=1;
}
void InterruptTimer0() interrupt 1
{
        TH0=T0RH;
        TL0=T0RL;
        UartRxMonitor(1);
        KeyScan();
}
```