

# 串口通信与LCD显示



专 业：生物医学工程

年 级：2022级

班 级：1班

姓 名：张台忍

学 号：3022202299

邮 箱：ztr8526@gmail.com

**2024年5月4日**

Contents

[串口通信与LCD显示 1](#_Toc166541617)

[一 实验目的 3](#_Toc166541618)

[二 实验设备 3](#_Toc166541619)

[三 实验内容 3](#_Toc166541620)

[1 建立工程 3](#_Toc166541621)

[2 需求分析 3](#_Toc166541622)

[3 硬件电路搭建 4](#_Toc166541623)

[4 代码编写 4](#_Toc166541624)

[5 的配置 6](#_Toc166541625)

[6 调试记录 8](#_Toc166541626)

[7 结果与讨论 8](#_Toc166541627)

# 一 实验目的

1. 掌握编写串口中断函数的方法
2. 掌握使用方法

# 二 实验设备

PC微机一台, 集成开发环境一套, 51单片机开发仪

# 三 实验内容

本实验流程图

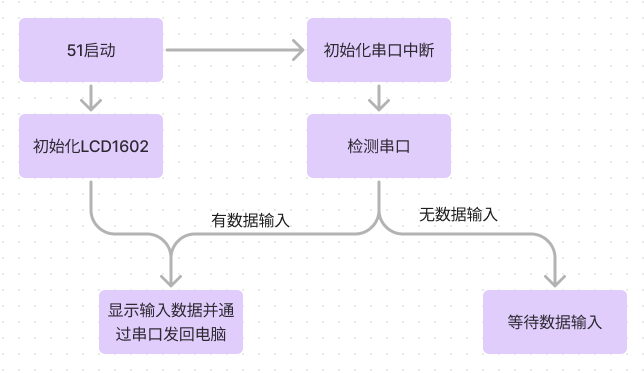


图 0 流程图

## 1 建立工程

在中, 参照实验的方法, 建立工程, 然后新建语言程序, 编译通过后进行接下来的操作.

## 2 需求分析

本次实验有两个需求, 描述如下:

1. 将接收到的字符回传给计算机, 其中小写字母转换为大写字母, 其他回传字符不变.
2. 将接收到的可显示字符(一律转为大写)显示在液晶屏上.

对于需求, 可以使用串口中断函数, 当有数据传入时, 触发串口中断, 中断函数中将接收到的数据传回电脑.

紧接着需求, 可以将需求中接收到的数据存起来, 然后调用的显示函数, 将字符显示在液晶屏上.

## 3 硬件电路搭建

参考原理图连接硬件电路, 连接图如下.

A close up of a circuit board

Description automatically generated

图1 硬件电路连接

本次实验硬件电路简单, 只需要把插上即可.

## 4 代码编写

完成硬件电路搭建后, 即可开始编写代码.

首先编写需求的代码. 使用串口中断之前应该先初始化串口并打开中断. 代码如下:

1. **void** UART\_Init()
2. {
3. SCON=0x50;      //配置serial control
4. PCON |= 0x80;
5. TMOD &= 0x0F;   //设置定时器模式
6. TMOD |= 0x20;   //设置定时器模式
7. TL1 = 0xF3;     //设定定时初值
8. TH1 = 0xF3;     //设定定时器重装值
9. ET1 = 0;        //禁止定时器1中断
10. TR1 = 1;        //启动定时器1
11. EA=1;
12. ES=1;           //打开中断
13. }

此代码是针对晶振为, 波特率为设置的初始化代码, 若是不同的晶振和波特率, 需要调整其中参数.

初始化完成后, 即可在文件中编写中断函数, 代码如下.

1. **char** str[32]="";
2. **static** **char** index=0;
3. **void** UARTFunction() interrupt 4
4. {
5. **if**(RI==1)
6. {
7. **char** rev=SBUF;
8. RI=0;
9. **if**(rev>=97 && rev<=122)       //小写转大写
10. {
11. UART\_SendByte(rev-32);  //回传
12. str[index]=rev-32;      //存储接收的数据
13. }
14. **else**
15. {
16. UART\_SendByte(rev);     //回传
17. str[index]=rev;         //存储接收的数据
18. }
19. index++;
20. }
21. }

这段代码使用存储接收到的数据, 使用来实现对不同位的定位. 代码中利用码大小写字母相差实现小写字母转大写字母. 另外, 用到了函数函数把接收到的字符又传回电脑. 下面是函数的具体实现.

1. **void** UART\_SendByte(unsigned **char** Byte)
2. {
3. SBUF=Byte;
4. **while**(TI==0);   //TI=1时发送
5. TI=0;           //清空
6. }

以上就实现了单片机接收字符并将小写字母转大写字母, 并且把接收到的传回电脑.

接下来是编写需求的代码.

需求要用到, 对于的配置较为复杂. 下节独立讨论, 此处先使用下节要实现的函数. 需求2在函数中实现, 具体代码如下.

1. **void** main()
2. {
3. LCD\_Init();
4. UART\_Init();
6. **while**(1)
7. {
8. **if**(index<17) LCD\_ShowString(1,1,str);//长度16以内显示在第一行
9. **else**
10. {
11. LCD\_ShowString(2,1,&str[16]);//长度超过16显示在第二行
12. **if**(index>32) index=0;//长度超过32索引置0
13. }
14. }
15. }

此代码将串口传入单片机的字符显示在屏幕上, 多次传入会在后续显示, 若传入的数据长度超过32, 会从首个字符开始覆盖.

接下来将实现对的配置.

## 5 的配置

本实验使用的是脚带背光的. 首先查阅数据手册, 对其引脚进行说明.

表1 引脚说明

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 编号 | 符号 | 引脚说明 | 编号 | 符号 | 引脚说明 |
| 1 | VSS | 电源地 | 9 | D2 | 数据 |
| 2 | VDD | 电源正极 | 10 | D3 | 数据 |
| 3 | VL | 液晶显示偏压 | 11 | D4 | 数据 |
| 4 | RS | 数据/命令选择 | 12 | D5 | 数据 |
| 5 | R/W | 读/写选择 | 13 | D6 | 数据 |
| 6 | E | 使能信号 | 14 | D7 | 数据 |
| 7 | D0 | 数据 | 15 | BLA | 背光源正极 |
| 8 | D1 | 数据 | 16 | BLK | 背光源负极 |

通过以上引脚功能, 即可实现对的操作. 以下是对应的功能表.

表2 指令

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 序号 | 指令 | RS | R/W | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| 1 | 清屏 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 2 | 归位 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | \* |
| 3 | 输入方式设置 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | I/D | S |
| 4 | 显示开关控制 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | D | C | B |
| 5 | 光标, 画面位移 | 0 | 0 | 0 | 0 | 0 | 1 | S/C | R/L | \* | \* |
| 6 | 功能设置 | 0 | 0 | 0 | 0 | 1 | DL | N | F | \* | \* |
| 7 | CGRAM地址设置 | 0 | 0 | 0 | 1 | A5 | A4 | A3 | A2 | A1 | A0 |
| 8 | DDRAM地址设置 | 0 | 0 | 1 | A6 | A5 | A4 | A3 | A2 | A1 | A0 |
| 9 | 读BF及AC值 | 0 | 1 | BF | AC6 | AC5 | AC4 | AC3 | AC2 | AC1 | AC0 |
| 10 | 写数据 | 1 | 0 | Data | | | | | | | |
| 11 | 读数据 | 1 | 1 | Data | | | | | | | |

通过赋值对应的引脚, 即可实现不同功能.

首先实现写入命令函数, 该函数实现对下达不同的指令. 代码如下.

1. //引脚配置：
2. sbit LCD\_RS=P2^6;
3. sbit LCD\_RW=P2^5;
4. sbit LCD\_EN=P2^7;
5. #define LCD\_DataPort P0
7. **void** LCD\_WriteCommand(unsigned **char** Command)
8. {
9. LCD\_RS=0;               //赋值RS
10. LCD\_RW=0;               //赋值R/W
11. LCD\_DataPort=Command;   //写入指令
12. LCD\_EN=1;               //使能LCD
13. LCD\_Delay();
14. LCD\_EN=0;               //关闭使能
15. LCD\_Delay();
16. }

通过这个代码可以对实现不同的指令写入.

再实现写入数据的函数. 此函数与上一函数类似. 只需要参照功能表赋值不同的和即可.

1. **void** LCD\_WriteData(unsigned **char** Data)
2. {
3. LCD\_RS=1;               //赋值RS
4. LCD\_RW=0;               //赋值R/W
5. LCD\_DataPort=Data;      //写入数据
6. LCD\_EN=1;               //使能LCD
7. LCD\_Delay();
8. LCD\_EN=0;               //关闭使能
9. LCD\_Delay();
10. }

由于有两行多列, 为了实现在不同的列显示字符的功能, 我们还需要操作光标, 使光标到达不同的位置开始显示. 下面的代码实现设置光标到不同的位置.

1. **void** LCD\_SetCursor(unsigned **char** Line,unsigned **char** Column)
2. {
3. **if**(Line==1)         //第一行
4. {
5. LCD\_WriteCommand(0x80|(Column-1));
6. }
7. **else** **if**(Line==2)    //第二行
8. {
9. LCD\_WriteCommand(0x80|(Column-1+0x40));
10. }
11. }

完成以上函数, 即可开始初始化, 只需要下达不同的指令即可, 代码如下.

1. **void** LCD\_Init()
2. {
3. LCD\_WriteCommand(0x38);//八位数据接口，两行显示，5\*7点阵
4. LCD\_WriteCommand(0x0c);//显示开，光标关，闪烁关
5. LCD\_WriteCommand(0x06);//数据读写操作后，光标自动加一，画面不动
6. LCD\_WriteCommand(0x01);//光标复位，清屏
7. }

以上就是的配置代码. 可利用上述代码实现不同的显示功能.

若要在某行某列显示字符, 可通过以下代码实现.

1. **void** LCD\_ShowChar(unsigned **char** Line,unsigned **char** Column,**char** Char)
2. {
3. LCD\_SetCursor(Line,Column);     //把光标放在line, column处
4. LCD\_WriteData(Char);            //写入数据显示
5. }

若要显示字符串, 加循环即可.

1. **void** LCD\_ShowString(unsigned **char** Line,unsigned **char** Column,**char** \*String)
2. {
3. unsigned **char** i;
4. LCD\_SetCursor(Line,Column);
5. **for**(i=0;String[i]!='\0';i++)
6. {
7. LCD\_WriteData(String[i]);
8. }
9. }

通过以上代码, 就实现了的配置和显示功能.

## 6 调试记录

调试过程中, 若只接收和发送单个字符, 实现起来特别简单. 因此我想实现对字符串的收发. 由于语言对字符串的支持不像其他高级语言一样完备, 对字符串的操作本质上还是操作字符. 以下是对不同的代码的改进.

单片机像电脑发送字符串代码.

1. **void** UART\_SendStr(unsigned **char** \*Str)
2. {
3. **while**(\*Str!=0)
4. {
5. SBUF=\*Str;
6. **while**(TI==0);
7. TI=0;
8. Str++;
9. }
10. }

本质就是遍历字符串, 将每一个字符都传回电脑.

接收字符串的代码在第节已经给出. 直接使用了鲁棒性更好的代码, 既可以接收字符, 又能接收字符串.

## 7 结果与讨论

实验结果现象如图所示.



图2 实验现象

A screenshot of a computer

Description automatically generated

图3 串口助手现象

本次实验实现了串口和的使用. 将串口接收到的数据显示在屏幕上. 进一步加深了对中断的理解, 并通过操作字符到操作字符串的扩展练习优化了单片机程序设计思路.