**北京邮电大学课程设计报告**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **课程设计**  **名称** | **微机系统与**  **接口技术** | | **学 院** | **计算机学院** | **指导教师** | **周锋** |
| **班 级** | **班内序号** | **学 号** | | **学生姓名** | **成绩** | |
|  |  |  | |  |  | |
|  |  |  | |  |  | |
| **课**  **程**  **设**  **计**  **内**  **容** | **实验目的**：进一步加深和巩固对所学知识点的理解和掌握；综合运用所学知识、工程原理来分析和完成本课程设计；理解计算机软件、硬件如何配合工作，能对计算机硬件进行功能扩展；掌握接口电路设计过程和驱动程序的编写方法；学会如何区分系统中软件、硬件故障和调试手段；培养计算机软硬件的综合设计能力和调试能力。  **基本内容**：(软件仿真)4x4小键盘接口设计；异步串行通信接口设计；4位数码管显示器设计。  (程序)键盘扫描程序；串行接口通信程序；4位七段数码管显示程序；    **实验方法**：在 Proteus 软件环境下，采用软件完成电路的搭建，采用汇编语言完成程序设计。  **团队分工**： 一人独立完成。 | | | | | |
| **学生**  **课程设计**  **报告**  （附页） | 见附页 | | | | | |
| **课**  **程**  **设**  **计**  **成**  **绩**  **评**  **定** | 遵照实践教学大纲并根据以下四方面综合评定成绩：  1、课程设计目的任务明确，选题符合教学要求，份量及难易程度  2、团队分工是否恰当与合理  3、综合运用所学知识，提高分析问题、解决问题及实践动手能力的效果  4、是否认真、独立完成属于自己的课程设计内容，课程设计报告是否思路清晰、文字通顺、书写规范  **评语**:        **成绩**:  指导教师签名：  年 月 日 | | | | | |

**1 实验目的**

1. 进一步加深和巩固对所学知识点的理解和掌握；

2. 综合运用所学知识、工程原理来分析和完成本课程设计；

3. 理解计算机软件、硬件如何配合工作，能对计算机硬件进行功能扩展；

4. 掌握接口电路设计过程和驱动程序的编写方法；

5. 学会如何区分系统中软件、硬件故障和调试手段；

6. 培养计算机软硬件的综合设计能力和调试能力。

**2 实验内容**

1. **硬件(软件仿真)部分**
2. 4x4小键盘接口设计（利用8255 C口）
3. 异步串行通信接口设计（8254为波特率发生器）

c. 4位数码管显示器设计

**2. 软件部分**

a. 键盘扫描程序。将按键0~F转换成0~15的二进制值

b. 串行接口通信程序。发送/接收均为查询方式

c. 4位七段数码管显示程序。左移位方式显示0~9、A~F。将串行接口发送端和接收端环回，当在4x4小键盘上敲击按键时，经串口发送、接收后将对应键值显示在4位数码管上。

**3 实验原理**

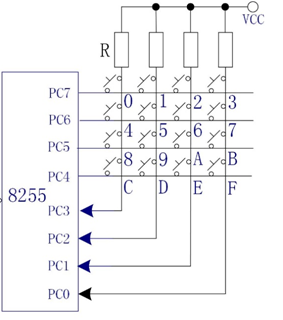
**3.1 设计思想**

**3.1.1 主体框架**



1. 发送数据：从8255C口进行键盘扫描读入按键，将其发送给8251。
2. 接收数据：从 8251接收到发送来的按键值，将其转为显示需要的七段码在液晶显示器LCD上进行移位显示。
3. 主函数为一个大循环，采用查询方式，依次查询是否有键被按下，发送是否准备好，接收是否准备好，条件满足则执行相应操作，不满足则进行下一个查询，而不是原地踏步地查询。循环执行主程序，直至程序终止。

**3.1.2 键盘扫描**



1. 8255C口高四位PC7-PC4输出，低四位PC3-PC0输入，接键盘的行3-列0。
2. 采用行扫描法，使键盘上某一行线为低电平，而其余行接高电平，然后读取列值；如果列值中有某位为低电平，则表明当前行与当前列交点处的键被按下；否则扫描下一行，直到扫完全部的行线为止。
3. 分别用寄存器记录当前行和当前列，并用 行数X 4+列数 计算键值。

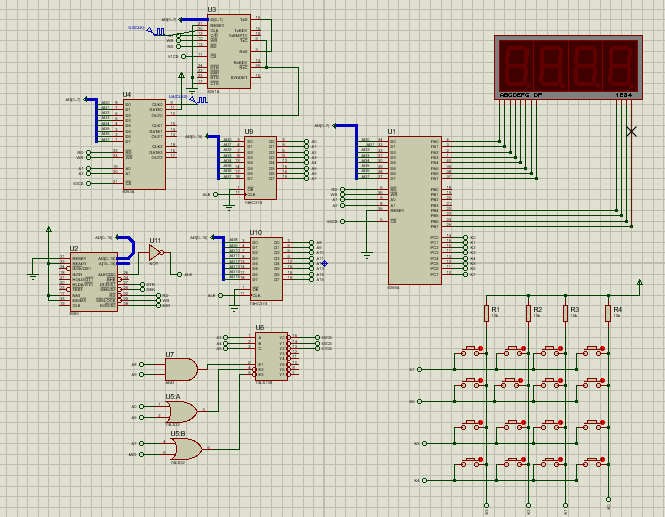
**3.1.3 数码管显示**

1. 8255B口输出，高四位PB7-PB4接LED位码S3-S0，选择某一个数码管。

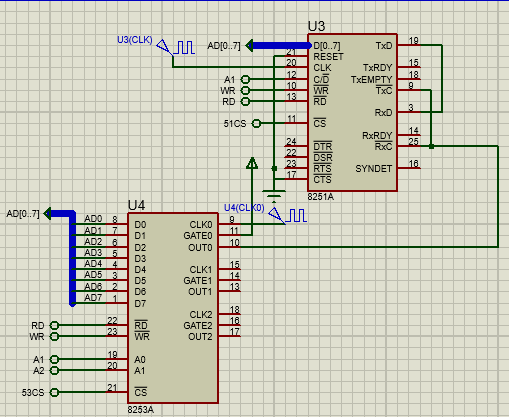
2. 8255A口输出，PA7-PA0接LED的DP-A，将七段译码值输出到数码管，并进行显示。

**3.2 实验原理图**

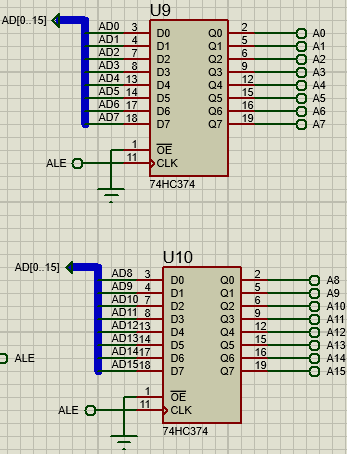
**3.2.1 系统接线图**



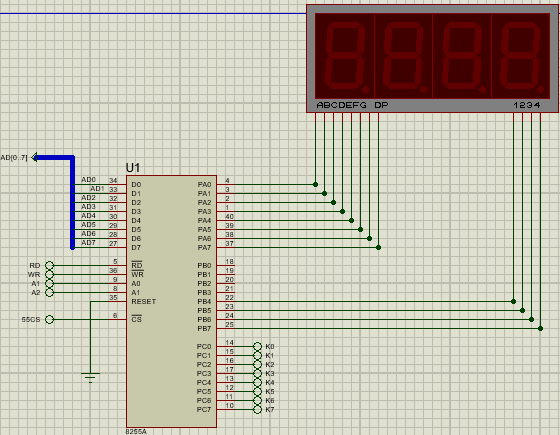
**3.2.2 自发自收**



**3.2.3 地址锁存模块**



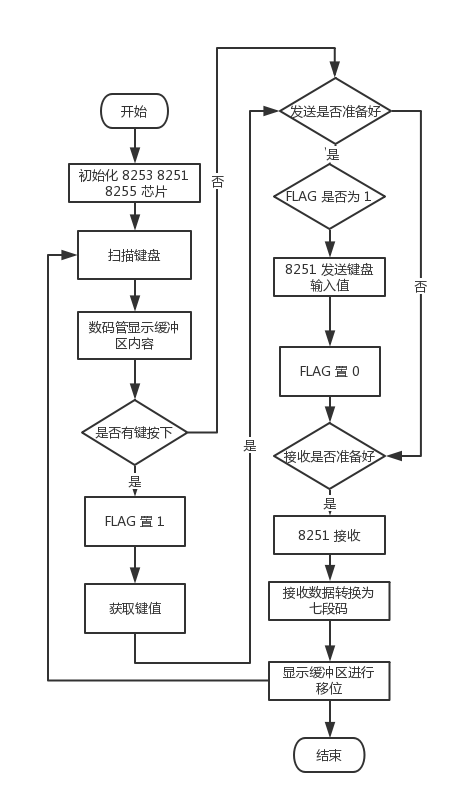
**3.2.4 显示模块**



**3.3 流程图**

此次课程设计中主要功能模块有三个：发送和接收模块、输入模块及显示模块。每个模块的逻辑功能流程图如下所示：

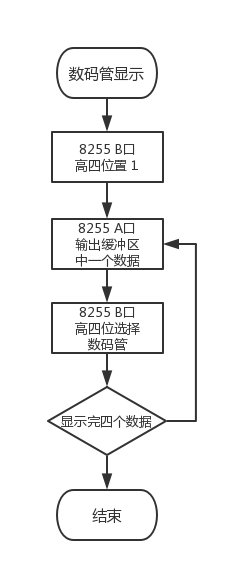
**3.3.1 主流程图**



**说明**

1. 在程序执行的过程中，通过 FLAG 标志是否从键盘中得到了输入值。从键盘中得到输入值后 FLAG置为 1，并在发送数据后将 FLAG 重置为 0。
2. 每次循环中都要显示一次数码管缓冲区的内容，否则数码管中看不见任何显示的内容。

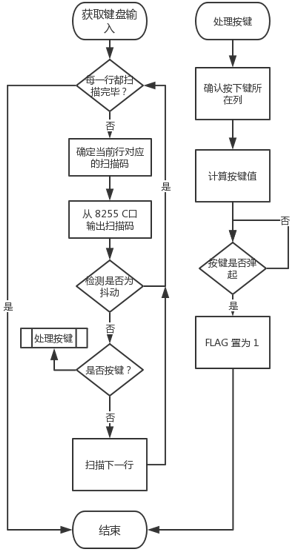
**3.3.2 显示模块流程图**



**说明**

1. 四个数码管中只可以显示一个，每个数码管轮流显示。
2. 通过视觉暂留效应，每个数码管高速的轮流显示，使得看起来是每次四个数码管同时显示

**3.3.3 键盘扫描流程图**



**说明**

1. 需要增加按键抬起识别的代码，否则每当检测到键盘输入就将输入值发送出去会使得多次重复输出同一值，导致错误。

**4 实验程序**

;====================================================================

; Main.asm file generated by New Project wizard

;

; Created: 周六 7月 8 2017

; Processor: 8086

; Compiler: MASM32

;

; Before starting simulation set Internal Memory Size

; in the 8086 model properties to 0x10000

;====================================================================

IO8251 EQU 310H

IO8251CTRL EQU 316H

IO8255A EQU 300H

IO8255B EQU 302H

IO8255C EQU 304H

IO8255CTRL EQU 306H

IO82530 EQU 308H

IO8253CTRL EQU 30EH

CODE SEGMENT PUBLIC 'CODE'

ASSUME CS:CODE,DS:DATA,SS:STACK

;延时子程序

DELAY PROC NEAR

PUSH CX

MOV CX,50H

WAIT1 : LOOP WAIT1

POP CX

RET

DELAY ENDP

;LED移位函数

LEDSHIFT PROC NEAR

PUSH DX

PUSH BX

PUSH AX

MOV BX,OFFSET BUF

;BUF中四个数依次左移，右补NUM

MOV DL,[BX+2]

MOV [BX+3],DL

MOV DL,[BX+1]

MOV [BX+2],DL

MOV DL,[BX]

MOV [BX+1],DL

MOV DL,NUM

MOV [BX],DL

POP AX

POP BX

POP DX

RET

LEDSHIFT ENDP

;LED显示函数，用于显示BUF中的四个数

SHOWLED PROC NEAR

PUSH DX

PUSH BX

PUSH AX

MOV BX,OFFSET BUF

;点亮数码管的最高位

MOV DX,IO8255B ;选择8255B口，控制位码

IN AL,DX

OR AL,0F0H ;位码置1

OUT DX,AL

MOV DX,IO8255A ;选择8255A口，输出最高位

MOV AL,[BX+3]

OUT DX,AL

MOV DX,IO8255B ;位码控制选数码管4

IN AL,DX

AND AL,0FH

OR AL,0E0H ;S3置0，其余置1

OUT DX,AL

;点亮数码管的第三最高位

MOV DX,IO8255B

IN AL,DX

OR AL,0F0H ;位码清0

OUT DX,AL

MOV DX,IO8255A ;选择8255A口，输出第二最高位

MOV AL,[BX+2]

OUT DX,AL

MOV DX,IO8255B ;位码控制选数码管2

IN AL,DX

AND AL,0FH

OR AL,0D0H ;S1置0，其余置1

OUT DX,AL

;点亮数码管的第二最高位

MOV DX,IO8255B

IN AL,DX

OR AL,0F0H ;位码清0

OUT DX,AL

MOV DX,IO8255A ;选择8255A口，输出第二最高位

MOV AL,[BX+1]

OUT DX,AL

MOV DX,IO8255B ;位码控制选数码管1

IN AL,DX

AND AL,0FH

OR AL,0B0H ;S1置0，其余置1

OUT DX,AL

;点亮数码管的最低位

MOV DX,IO8255B

IN AL,DX

OR AL,0F0H ;位码清0

OUT DX,AL

MOV DX,IO8255A ;选择8255A口，输出最低位

MOV AL,[BX]

OUT DX,AL

MOV DX,IO8255B ;位码控制选数码管0

IN AL,DX

AND AL,0FH

OR AL,070H ;S0置0，其余置1

OUT DX,AL

MOV DX,IO8255B

IN AL,DX

OR AL,0F0H ;位码清0

OUT DX,AL

POP AX

POP BX

POP DX

RET

SHOWLED ENDP

;获取键盘输入

GETKEY PROC NEAR

PUSH AX ;保护现场

PUSH CX

PUSH DX

MOV CX,00H ;从第一行开始扫描

CHECK:

MOV DX,IO8255C ;选择C口地址

MOV BX,OFFSET SCAN ;得到扫描码

ADD BX,CX

MOV AL,[BX]

OUT DX,AL ;把要扫描的那一行置0，其余置1

;防止抖动

IN AL,DX ;判断是否有键盘按下

MOV AH,AL

CALL DELAY ;延迟一段时间，避过抖动区间

IN AL,DX ;延时后再次读入

CMP AL,AH ;比较两次读入的值

JNZ CHECK ;不相等说明为抖动，重新检测

;判断按下的是哪一列

AND AL,0FH

CMP AL,0FH ;没有键按下

JZ NEXT

CMP AL,0EH ;第四列有键按下

JZ NEXT1

CMP AL,0DH ;第三列有键按下

JZ NEXT2

CMP AL,0BH ;第二列有键按下

JZ NEXT3

MOV BX,00H ;第一列有键按下

JMP GETNUM

NEXT: INC CX ;修改变量扫描下一行

CMP CX,04H

JNZ CHECK ;没有扫描完四行跳转

JMP DONE ;已经扫描完四行跳转

;记录是哪一列有键按下

NEXT1: MOV BX,03H

JMP GETNUM

NEXT2: MOV BX,02H

JMP GETNUM

NEXT3: MOV BX,01H

;计算按下的键的数值

GETNUM: MOV AL,CL

MOV DL,04H

MUL DL ;行数乘以4，再加上列数

ADD BL,AL ;此时BL中所存即为按下的键

;检测键盘是否弹起

MOV DX,IO8255C ;C口输入

IN AL,DX

MOV AH,AL

NOTUP: CALL DELAY

IN AL,DX

CMP AL,AH

JZ NOTUP ;未弹起，延时重复检测

MOV FLAG,01H ;有键按下，把flag置1，

DONE: POP DX ;恢复现场

POP CX

POP AX

RET

GETKEY ENDP

;找到键值对应的七段译码值，存到NUM中

FINDLED PROC NEAR

XOR CX,CX

MOV CL,BL ;BL中是键盘按下的值

MOV BX,OFFSET LEDTABLE

ADD BX,CX ;此时BX即为LEDTABLE中对应的地址

MOV CL,[BX] ;CL内存着此次按下键盘值的七段译码值

MOV NUM,CL

RET

FINDLED ENDP

START:

MOV AX,DATA

MOV DS,AX

;初始化8253

MOV DX,IO8253CTRL ;8253控制端口地址

MOV AL,16H ;8253选择计数器0低八位，方式3，二进制计数,00010110

OUT DX,AL

MOV DX,IO82530 ;计数器0地址

MOV AL,52 ;写入计数初值52（时钟1M，波特率1200baud，波特率因子16）

OUT DX,AL

;初始化8251

MOV DX,IO8251CTRL ;8251控制端口地址

XOR AL,AL

OUT DX,AL ;输出三次 0 进行预初始化。

OUT DX,AL

OUT DX,AL

MOV AL,40H ;工作命令字软复位，01000000

OUT DX,AL

NOP

MOV AL,4EH ;方式命令字，一个停止位，无校验，8个数据位，波特率因子16，异步01001110

OUT DX,AL

NOP

MOV AL,27H ;工作命令字，允许发送接收，00100111

OUT DX,AL

NOP

;初始化8255

MOV DX,IO8255CTRL ;8255控制端口地址

MOV AL,081H ;方式控制字：C口低四位输入，B口输出，方式0 ，A口输出，方式0

OUT DX,AL

MOV FLAG,00H

;扫描键盘

LOOP1:

CALL GETKEY

CALL SHOWLED ;每次循环时显示 BUFFER

MOV DX,IO8251CTRL ;检验8251发送是否准备好TxRDY

IN AL,DX

AND AL,01H

JNZ SEND ;8251发送准备好,转去发送

;发送未准备好,检验接收是否准备好

LOOP2:

MOV DX,IO8251CTRL

IN AL,DX

AND AL,02H ;检验8251发送是否准备好RxRDY

JNZ RECEIVE ;接收准备好，转去接收

JMP LOOP1 ;接收未准备好，转去扫描键盘

;发送准备好，判断键盘是否按下

SEND:

CMP FLAG,01H

JE SSEND

JMP LOOP2 ;发送准备好，但是没有键盘按下，转去检验接收

;键盘按下，要发送数据

SSEND:

MOV DX,IO8251 ;8251数据端口地址

MOV AL,BL

OUT DX,AL ;把发送的数据（ASC值）输出到8251

CALL DELAY

MOV FLAG,00H ;显示后把flag恢复为0

JMP LOOP2

;开始接收

RECEIVE: MOV DX,IO8251

IN AL,DX ;8251接收到数据，存到CU中

MOV BL,AL

CALL FINDLED ;将数据转为七段译码值

CALL LEDSHIFT ;移位后显示

JMP LOOP1

;退出

EXIT: MOV AL,00H ;结束输出0

MOV DX,IO8255A ;A口输出

OUT DX,AL

;MOV BL,01H ;清除显示

CODE ENDS

DATA SEGMENT

;0~F七段译码值，用于共阴极LED显示

LEDTABLE DB 3FH,06H,5BH,4FH,66H,6DH,7DH,07H,7FH,6FH,77H,7CH,39H,5EH,79H,71H

;扫描码，每一行置为0，其余行置为1

SCAN DB 70H,0B0H,0D0H,0E0H

BUF DB 00H,00H,00H,00H ;LED显示的四位数

FLAG DB 00H ;是否有数据传输的标志位

NUM DB 00H ;输入的数

DATA ENDS

STACK SEGMENT STACK 'STACK'

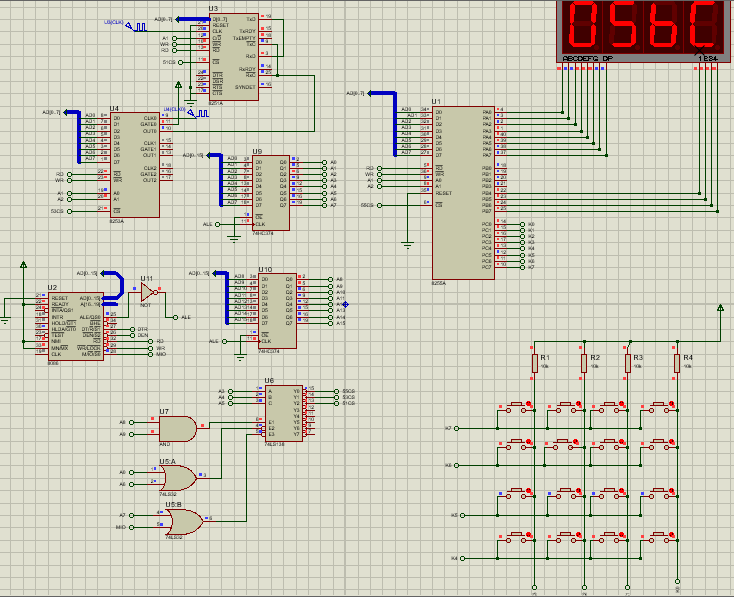
DB 100 DUP(?)

STACK ENDS

END START

**5 实验结果**

系统为自发自收。按下4×4键盘上任一按键后，可以看到对应键值在数码管上移位显示，即通过串口通信程序发送出数据。并且每按一次，数据左移一位。如下图所示：



**6 问题解决及收获体会**

**6.1 问题解决**

1.问题：LED显示无效，调试时每次只显示一位。

解决：在程序主循环中每次都要执行一下 LED 显示的函数，因为 LED 的显示是高电平使能。并且 LED 的显示原理是利用了视觉暂留效应，每次只能显示一位数字，通过高速的循环显示使得多为数字同时显示。

2.问题：8251 状态端口读出的 TxRDY 信号位总为 0，无法发送数据。

解决：在初始化话 8251 前预先向其控制字端口输出三次 0 对其进行预初始化即可。

3.问题：8251 状态端口读出的 RxRDY 信号位总为 0，无法接收数据。

解决：在发送完数据之后让程序延迟等待一下，否则无法接收到数据。

4.问题：LED 显示乱码

解决：一开始对键盘键值的编码使用了共阳极七段码，而实际使用的是共阴极的七段码，重新编写相应的七段码转换表即可解决问题。

5.问题：proteus 中调试时显示代码与执行代码位置不同步。

解决：这是由于 proteus 自身的 BUG 引起的，将数据段和堆栈段放在代码段之后执行与显示的代码位置即可同步。

**6.2** **收获体会**

在本次课程设计中，我一个人一组，使用 proteus 软件在电脑上完成了课程设计。在 proteus 软件中搭建电路进行硬件仿真，并完成汇编代码的编写，最终经过不断的调试，成功完成了课设。

在完成课程设计的过程中，出现了许多的问题。我们需要清楚地定位问题的来源是硬件还是软件，从而解决它们。这个过程是繁琐并且枯燥的。因此，耐下心来，慢慢调试，发现错误并修正错误是我在此次课设中最大的收获。

另外，我使用了模块化的设计完成课设。因此通过单步调试的方法，我们可以隔离各个模块，逐步确定每一个模块的输入输出是否正确，从而判断出问题究竟出现在哪一个模块中，问题是在硬件还是在软件。

当问题出现在硬件上时，我们需要查询出现问题的硬件的相关文档资料，弄明白是硬件的连线有问题还是汇编代码的编写有问题。

总之，通过此次课程设计，我掌握了 proteus 软件的基本使用方法，锻炼了发现以及解决问题的能力，学习到了如何定位问题出现在硬件或是软件上的能力，使我的动手能力及思维方式都有了进一步的提高。