**基于STC89C52单片机的计数器设计**

# 任务书

## 设计任务书

1. 掌握单片机开发的一般方法。
2. 掌握计分器的一般工作原理。

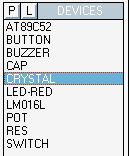
## 基本要求

本实验中设计的计时计分器，硬件电路原理图见下图1,采用 AT89C52 单片机作为控制核心,LCD1602为显示器，K0键为比赛开始/暂停键，K1~K4为4个比分刷新键，K1、K2键分别控制甲对分数的加减，K3、K4键分别控制乙对分数的加减，每按下一次均加一或减一。P2.0~ P2.7为LCD液晶显示器的数据信号输入端,LCD1602的数据或指令的选择端RS接单片机的P3.5口，读写控制端R/S接单片机的P3.6口，LCD模块的使能信号控制端E接单片机的P3.7口。由于指示灯和蜂鸣器的工作电流一般都在百毫安以上,故一定要使用功率器件驱动,如晶体三极管,小功率MOS管等，这样I/O口只担负驱动功率器件的任务，负荷电流是很小的（5mA以下）。故由图可看见,蜂鸣器一端通过三极管接在单片机的P3.0口,另一端接5V高电平。8个装饰彩灯一端分别接在电源上，另一端则分别接在P1.0~ P1.7口。当比赛开始/暂停键K0按下，LCD液晶显示器才会开始计时，彩灯会积累被点亮。

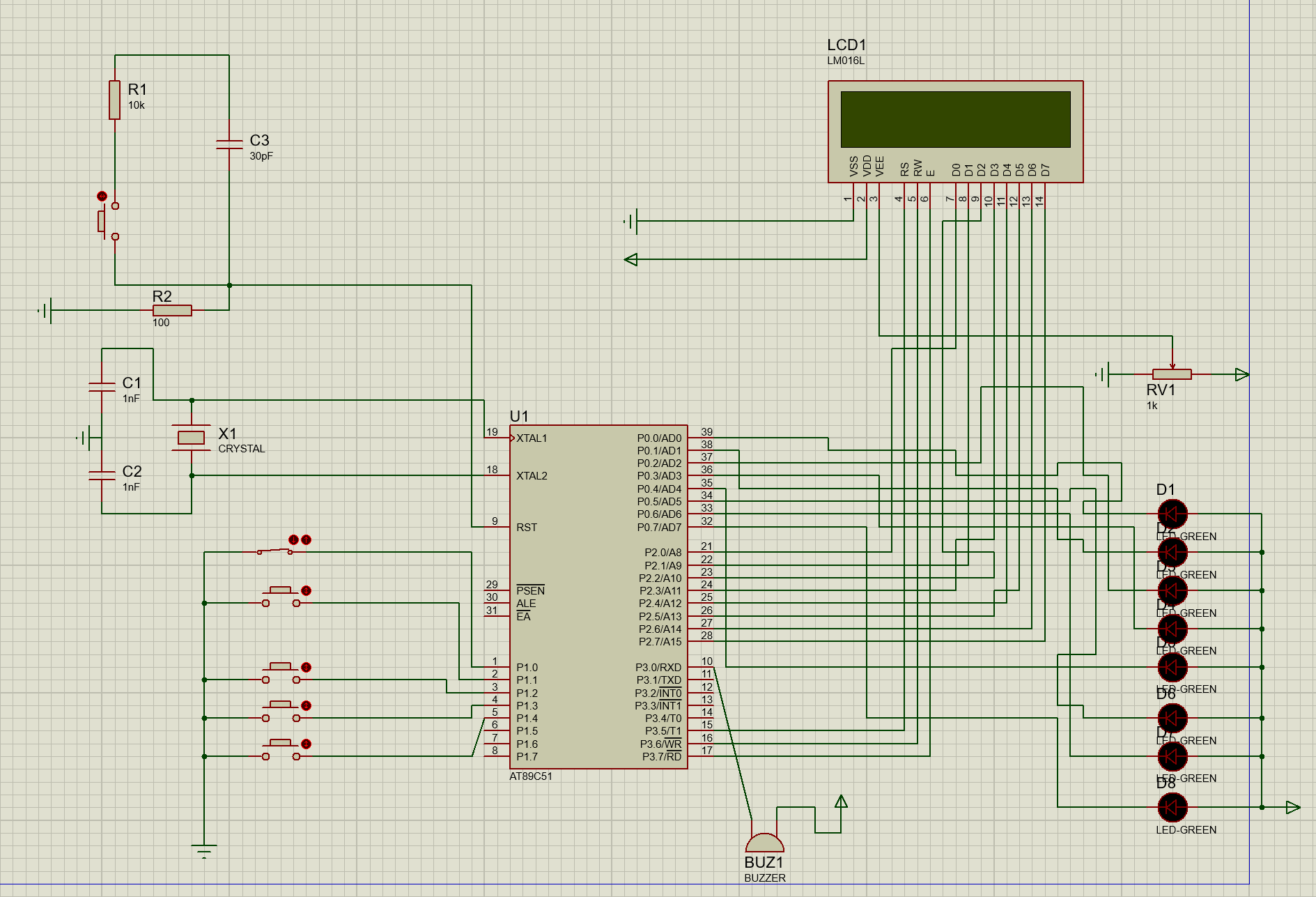
复位电路：采用按键复位电路,AT89C52 的 9 脚外接 RC电路及按键。由于单片机为高电平复位,开始时要先按下系统复位按钮K5,以清除系统信息 ,同时开放各按钮,以备各参赛选手进行抢答.每次答题结束,需要进行下一轮的抢答前都需按一下按钮K5。

时钟电路：采用内部时钟电路,18脚、19脚外接晶振和电容.晶振选用 6MHz的石英晶体。

本实验电路用到的器件如图2所示：



本实验的仿真电路图为：

****

目录

**[一． 系统需求分析 5](#_Toc20106)**

[1.1市场分析 5](#_Toc12710)

[2.2电路功能介绍 5](#_Toc26034)

**[二． 系统设计方案 6](#_Toc6811)**

[2.1 代码示例 6](#_Toc16552)

[2.2程序流程图 14](#_Toc6232)

**[三． 系统硬件设计及作用 16](#_Toc11058)**

[3.1器件概览 16](#_Toc32404)

[3.2器件作用 16](#_Toc5005)

**[四． 系统软件设计 17](#_Toc20082)**

[4.1主要函数功能 17](#_Toc31429)

[4.2遇到的问题及解决办法 20](#_Toc16251)

**[五． 系统演示结果 24](#_Toc12860)**

[5.1重要示例图 24](#_Toc32127)

[5.2心得体会及尚存在的问题 26](#_Toc16024)

# **系统需求分析**

## 1.1市场分析

计数器在目前的实际生活中有着很大的应用，例如比赛的积分系统和竞猜节目的计分系统等都用到了计数器。比赛的积分系统是对体育比赛过程中产生的时间，比分等数据进行快速记录和采集以及加工处理。但是比赛时经常采用人工计时计分的方式，由裁判进行人工计时和计分，并进行比较，因此人为因素对比赛的干扰性很强，从而设计一款能取代人工计时计分的系统是非常有必要的，能够一定程度上的提高比赛的正性和结果信服力。

本次设计的计时计分系统就是以单片机为核心的简易计时计分系统，主要由ＳＴＣ８９Ｃ５１芯片级以及ＬＣＤ１６０２彩屏进行控制。用汇编语言进行编程并写入单片机执行。

注：本次代码已上传至https://github.com/huxiao1/jishuqi/

## 2.2电路功能介绍

当P1.0单刀开关拨下则计时功能开始，此时P1.1，P1.2按键控制的第一项分数的加减功能和P1.3，P1.4控制的第二项的分数加减功能开始起作用。八个LED灯代表从0到9的二进制技术显示，显示屏的第一行可供自己编辑以适应不同的应用场景。

# **系统设计方案**

## **2.1 代码示例**

RS EQU P3.5 // \*数据H/命令L选择端 2.5

RW EQU P3.6 // 读H/\*写L选择端 2.6

E EQU P3.7 // 使能信号 2.7

LCD EQU P2 // P2口接LCD 开发板P0

BF EQU P2.7

ORG 0000H

LJMP MAIN

ORG 000BH

LJMP JISHI

ORG 0040H

/\*中断函数\*/

JISHI:

MOV DPTR,#TAB2

JNB P1.0,START //位为0则计时开始跳转到计时函数

AJMP RETURN //位为1中断返回

START:

JB P1.1,L2 //jiaoda++ 位为1则跳转 扫描下一个按键

/\*以下为检测到P1.1按键按下时的解决办法\*/

MOV 36H,#0FFH

K1: MOV 37H,#0FFH //延时

DL0:DJNZ 37H,DL0

DL1:DJNZ 36H,K1

LJMP DK1

L2: JB P1.2,L1 //jiaoda-- 位为1则跳转 扫描下一个按键

/\*以下为检测到P1.2按键按下时的解决办法\*/

MOV 36H,#0FFH

K2: MOV 37H,#0FFH //延时

DL2: DJNZ 37H,DL2

DL3: DJNZ 36H,K2

LJMP DK2

L1: JB P1.3,L //caida++ 位为1则跳转 扫描下一个按键

MOV 36H,#0FFH

/\*以下为检测到P1.3按键按下时的解决办法\*/

K3: MOV 37H,#0FFH //延时

DL4: DJNZ 37H,DL4

DL5: DJNZ 36H,K3

LJMP DK3

L: JB P1.4,START1 //caida-- 位为1则跳转 扫描下一个按键

MOV 36H,#0FFH

/\*以下为检测到P1.4按键按下时的解决办法\*/

K4: MOV 37H,#0FFH //延时

DL6: DJNZ 37H,DL6

DL7: DJNZ 36H,K4

LJMP DK4

START1: DEC P0

MOV A,P0

CJNE A,#00H,START2 //A不为00H则转移

MOV P0,#0FFH

START2:

MOV R1,#15

//秒位

MOV A,R2

MOVC A,@A+DPTR //R2=0 从0开始加

CALL LCDP2

INC R2

CJNE R2,#10,RETURN //R2没有到10就进入中断返回函数

MOV R2,#0

DEC R1

DEC R1 //R1=13

INC R3 //R3=1 从1开始加

MOV A,R3

MOVC A,@A+DPTR

CALL LCDP2

CJNE R3,#10,RETURN

MOV R3,#0

DEC R1 //R1=12

INC R4 //R4=1

MOV A,R4

MOVC A,@A+DPTR

CALL LCDP2

CJNE R4,#6,RETURN

MOV R4,#0

MOV A,R4

MOVC A,@A+DPTR

CALL LCDP2

DEC R1

DEC R1 //R1=10

INC R5

MOV A,R5

MOVC A,@A+DPTR

CALL LCDP2

CJNE R5,#10,RETURN

MOV R5,#0

MOV A,R5

MOVC A,@A+DPTR

CALL LCDP2

DEC R1 //R1=9

INC R6

MOV A,R6

MOVC A,@A+DPTR

CALL LCDP2

CJNE R6,#6,RETURN

AJMP RING //蜂鸣器

RET

//定位到所需位置并改数据

LCDP2:PUSH ACC

MOV A,R1

ADD A,#0C0H

CALL WAIT

CALL W\_CMD

POP ACC

CALL WAIT

CALL W\_DATA

RET

/\*最高位溢出蜂鸣器函数\*/

RING: CLR P3.0

MOV 38H,#0FFH

RING1: DJNZ 38H, RING1 //响的延时函数

RET

/\*P1.1按下交大++\*/

DK1:

MOV R1,#2 //交大的个位

MOV A,R7

MOVC A,@A+DPTR

CALL LCDP2

INC R7 //从 1到9再变化到 0

CJNE R7,#11,RETURN

MOV R7,#0

MOV R1,#1 //交大的十位

MOV A,30H

MOVC A,@A+DPTR

CALL LCDP2

INC 30H

MOV A,30H

CJNE A,#11,RETURN

MOV 30H,#0

MOV R1,#0 //交大的百位

MOV A,31H

MOVC A,@A+DPTR

CALL LCDP2

INC 31H

AJMP RETURN

RETURN: MOV TH0,#3CH //从新赋初值中断结束

MOV TL0,#0B0H

RETI

/\*P1.2按下交大--\*/

DK2:MOV R1,#2 //交大的个位

DEC R7

DEC R7 //此时R7是FFH

MOV A,R7

MOVC A,@A+DPTR

CALL LCDP2

INC R7

CJNE R7,#0,RETURN

MOV R7,#10

MOV A,R7

MOVC A,@A+DPTR

CALL LCDP2

DEC 30H

DEC 30H

MOV R1,#1 //交大的十位

MOV A,30H

MOVC A,@A+DPTR

CALL LCDP2

AJMP RETURN

DK3: MOV R1,#6

MOV A,32H

MOVC A,@A+DPTR

CALL LCDP2

INC 32H

MOV A,32H

CJNE A,#11,RETURN

MOV 32H,#0

MOV R1,#5

MOV A,33H

MOVC A,@A+DPTR

CALL LCDP2

INC 33H

MOV A,33H

CJNE A,#11,RETURN

MOV 33H,#0

MOV R1,#4

MOV A,34H

MOVC A,@A+DPTR

CALL LCDP2

INC 34H

AJMP RETURN

DK4: MOV R1,#6

DEC 32H

DEC 32H

MOV A,32H

MOVC A,@A+DPTR

CALL LCDP2

INC 32H

MOV A,32H

CJNE A,#0,RETURN

MOV 32H,#10

MOV A,32H

MOVC A,@A+DPTR

CALL LCDP2

DEC 33H

DEC 33H

MOV R1,#5

MOV A,33H

MOVC A,@A+DPTR

CALL LCDP2

AJMP RETURN

/\*

LCDP2:PUSH ACC

MOV A,R1

ADD A,#0C0H

CALL WAIT

CALL W\_CMD

POP ACC

CALL WAIT

CALL W\_DATA

RET

\*/

//RS=1/0 数据/命令(坐标）选择端

//R/W=1/0 读/写选择端

//E 使能

/\*初始化\*/

WAIT:MOV LCD,#0FFH

CLR RS //命令选择

SETB RW

CLR E

NOP

SETB E //读取BUSY标志

JB BF,WAIT //BF为1则转移

RET

/\*写指令\*/

W\_CMD:ACALL WAIT

MOV LCD,A //数据线赋值

CLR RS

CLR RW //写入一个字节的控制指令

SETB E

NOP

CLR E //结束关指令

RET

/\*写数据\*/

W\_DATA:ACALL WAIT

MOV LCD,A

SETB RS //写数据

CLR RW

SETB E

NOP

CLR E

RET

TAB:DB 'HAINAN VS HEBEI'

DB '000:000 00:00:0'

TAB2: DB 30H,31H,32H,33H,34H,35H,36H,37H,38H,39H,30H //数字转换成ascII值

MAIN:

MOV R0,#0

MOV R2,#0

MOV R3,#0

MOV R4,#0

MOV R5,#0

MOV R6,#0

MOV R7,#1

MOV 30H,#1

MOV 31H,#1

MOV 32H,#1

MOV 33H,#1

MOV 34H,#1

MOV P0,#0FFH

/\*1602管脚配置\*/

MOV A,#00000001B //清屏指令，光标归位,ddr内部数据清空

ACALL W\_CMD

MOV A,#00111000B //开显示 8位数据总线，显示两行 5\*7点阵每字符

ACALL W\_CMD

MOV A,#00001100B //显示功能开无光标 显示开关控制指令

ACALL W\_CMD

MOV A,#00010100B // 每输入一次指令，AC加一，整体画面不滚动

ACALL W\_CMD

MOV A,#10000000B // 设置数据指针起点，第一行第一个位置

ACALL W\_CMD

MOV DPTR,#TAB

MOV R0,0

/\*写第一行数据\*/

LOOP1:MOV A,R0

MOVC A,@A+DPTR

ACALL W\_DATA

INC R0

CJNE R0,#15,LOOP1 //第一行的数据要全写入

MOV A,#0C0H //下一步写数据的位置为第二行开头

ACALL W\_CMD

/\*写第二行数据\*/

LOOP2:MOV A,R0

MOVC A,@A+DPTR

ACALL W\_DATA

INC R0

CJNE R0,#31,LOOP2

/\*T0定时中断初始化 16位定时器定时15536\*/

MOV TMOD,#0000001B

MOV TH0,#3CH

MOV TL0,#0B0H

SETB EA

SETB ET0

SETB TR0

LOOP3:AJMP LOOP3 //等待中断

END

## 2.2程序流程图

### 2.2.1主函数流程图

程序初始化

MAIN

命令，数据写入LCD1602

W\_CMD写命令

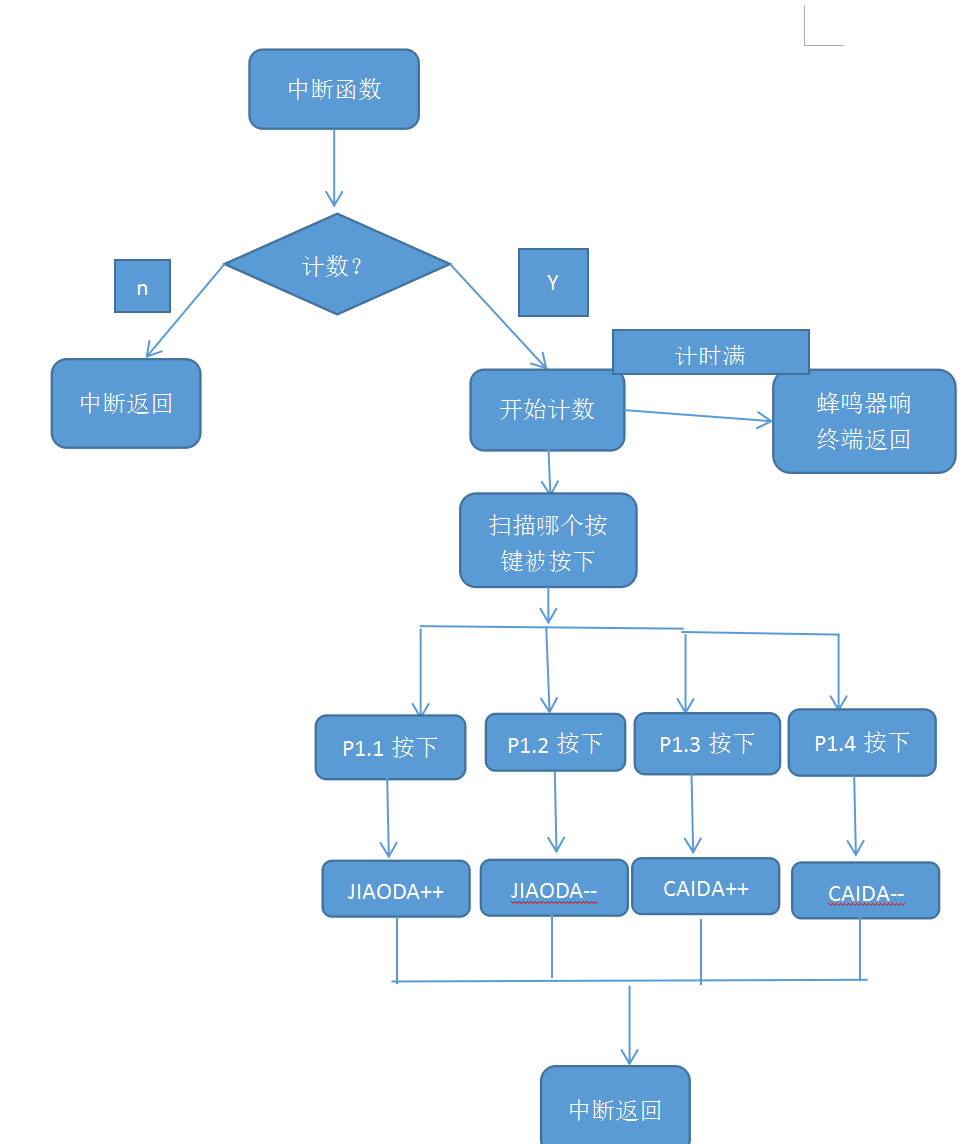
W\_DATA写数据

WAIT查看BUSY函数

定时器T1初始化

中断函数

### 2.2.2中断函数流程图



# **系统硬件设计及作用**

## 3.1器件概览

|  |  |
| --- | --- |
| 器件 | 作用电路 |
| 22PF电容两个 | 晶振电路 |
| 30PF电容一个 | 复位电路 |
| 51单片机 |  |
| LCD1602 | 显示部分 |
| 滑动变阻器 |
| 蜂鸣器 |  |
| Led灯8个 | 对应二进制显示 |
| 按键开关和单刀开关 | 控制电路 |

## 3.2器件作用

按键开关：控制记分板上分数的加减。

单刀开关：控制计数的开始与按键开关的作用是否启用。

LED灯：显示目前秒数的二进制。

蜂鸣器：计数器溢出提示。

LCD1602：分数和计时时间的显示功能。

其他：晶振电路保证单片机正常起振和复位电路保证单片机的容错和跑飞。

# 系统软件设计

## 4.1主要函数功能

W\_CMD:ACALL WAIT

MOV LCD,A //数据线赋值

CLR RS

CLR RW //写入一个字节的控制指令

SETB E

NOP

CLR E //结束关指令

RET

W\_DATA:ACALL WAIT

MOV LCD,A

SETB RS //写数据

CLR RW

SETB E

NOP

CLR E

RET

**功能：向LCD写入命令和数据**

/\*写第一行数据\*/

LOOP1:MOV A,R0

MOVC A,@A+DPTR

ACALL W\_DATA

INC R0

CJNE R0,#15,LOOP1 //第一行的数据要全写入

MOV A,#0C0H //下一步写数据的位置为第二行开头

ACALL W\_CMD

/\*写第二行数据\*/

LOOP2:MOV A,R0

MOVC A,@A+DPTR

ACALL W\_DATA

INC R0

CJNE R0,#31,LOOP2

**功能：利用查表的方式向LCD写入两行数据**

LCDP2:PUSH ACC

MOV A,R1

ADD A,#0C0H

CALL WAIT

CALL W\_CMD

POP ACC

CALL WAIT

CALL W\_DATA

RET

**功能：因为LCD需要一位一位的写入数据，所以此子函数用来当LCD屏上的数据需要发生变化时定位到所需位置并将变化后的数据重新写入。并提拱了对累加器的内容保护。**

L2: JB P1.2,L1 //jiaoda-- 位为1则跳转 扫描下一个按键

/\*以下为检测到P1.2按键按下时的解决办法\*/

MOV 36H,#0FFH

K2: MOV 37H,#0FFH //延时

DL2: DJNZ 37H,DL2

DL3: DJNZ 36H,K2

LJMP DK2

**功能：检测此按键是否按下**

DK2:MOV R1,#2 //交大的个位

DEC R7

DEC R7 //此时R7是FFH

MOV A,R7

MOVC A,@A+DPTR

CALL LCDP2

INC R7

CJNE R7,#0,RETURN

MOV R7,#10

MOV A,R7

MOVC A,@A+DPTR

CALL LCDP2

DEC 30H

DEC 30H

MOV R1,#1 //交大的十位

MOV A,30H

MOVC A,@A+DPTR

CALL LCDP2

AJMP RETURN

**功能：对LCD上显示的分数部分进行改写，分为个位和十位的两个数据的改写**

MOV A,R2

MOVC A,@A+DPTR //R2=0 从0开始加

CALL LCDP2

INC R2

CJNE R2,#10,RETURN //R2没有到10就进入中断返回函数

MOV R2,#0

DEC R1

DEC R1 //R1=13

INC R3 //R3=1 从1开始加

MOV A,R3

MOVC A,@A+DPTR

CALL LCDP2

CJNE R3,#10,RETURN

MOV R3,#0

DEC R1 //R1=12

INC R4 //R4=1

MOV A,R4

MOVC A,@A+DPTR

CALL LCDP2

CJNE R4,#6,RETURN

MOV R4,#0

MOV A,R4

MOVC A,@A+DPTR

CALL LCDP2

DEC R1

DEC R1 //R1=10

**功能：计数显示部分的秒位，分位和时位的显示方式，当秒位计数到60指针自动前移一位到分位当分位达到60则前移到十位以此类推。**

## 4.2遇到的问题及解决办法

1. 按键消抖问题：

因为一开始没有加软件消抖程序所以当加计数或者减计数被按下后数字都自动加二或者减二，因此在每个扫描按键子程序的后面都加入了软件消抖程序

例如：

JB P1.4,START

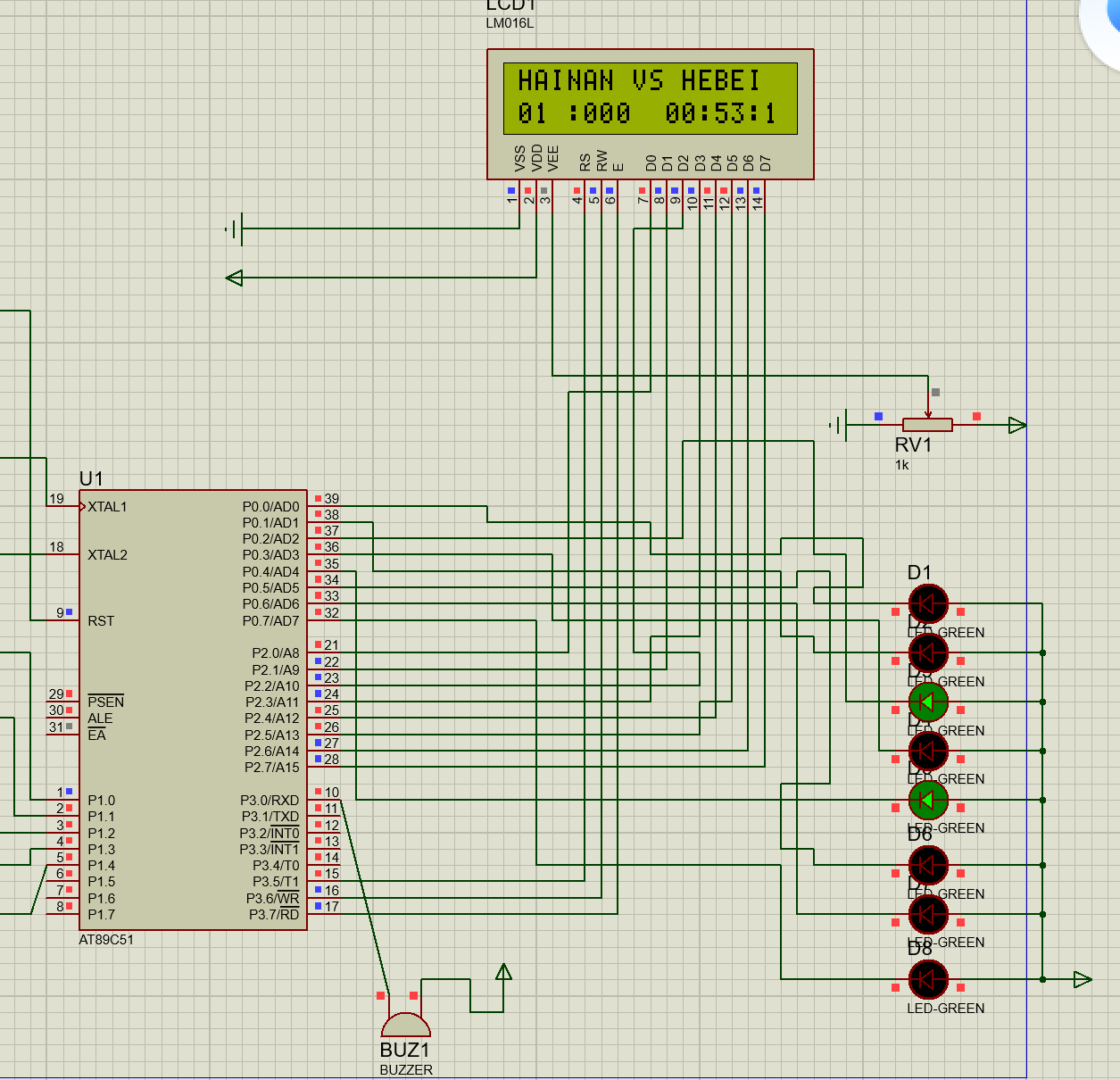
SETB P1.4

LJMP DK4

按键1.4松开则回到START函数，否则按键清零。

1. 十位数减到个尾市出现的问题

当分数加到10时此时要减一就会出现如下界面



说明加减技术的衔接寄存器可能出现了冲突混用。

解决办法：

一开始我想当判断到交大的个位为0且要执行减计数时则进入我设计的子程序：

GAI: MOV R7,#0BH

MOV R1,#1 //交大的十位

DEC 30H

DEC 30H

MOV A,30H

MOVC A,@A+DPTR

CALL LCDP2

MOV R1,#2

AJMP DO

让个位变为9十位减一，可以成功的从10过渡到9

但是这样当减到9之后再加想1进位时便会出现数字乱掉的问题。后来发现时十位的寄存器没有改回来，发现还是寄存器出现混用冲突的问题因此采用多个子程序来进行问题数据的独立显示，改进如下：

DK1:

MOV R1,#2 //交大的个位

MOV A,R7

MOVC A,@A+DPTR

CALL LCDP2

INC R7 //从 1到9再变化到 0

CJNE R7,#11,RETURN

MOV R7,#0

MOV R1,#1 //交大的十位

MOV A,30H

MOVC A,@A+DPTR

CALL LCDP2

INC 30H

MOV A,30H

CJNE A,#11,RETURN

MOV 30H,#0

MOV R1,#0 //交大的百位

MOV A,31H

MOVC A,@A+DPTR

CALL LCDP2

INC 31H

AJMP RETURN

RETURN: MOV TH0,#3CH //从新赋初值中断结束

MOV TL0,#0B0H

RETI

DEC R7

/\*P1.2按下交大--\*/

DK2: MOV R1,#2 //交大的个位

CJNE R7,#00H,DO

AJMP GAI

DO:

INC R7

CJNE R7,#0AH,GAI2

DO2:

DEC R7

CJNE R7,#00H,DO3

MOV R7,#0AH

MOV R1,#1 //交大的十位

DEC 30H

DEC 30H

MOV A,30H

MOVC A,@A+DPTR

CALL LCDP2

MOV R1,#2

INC 30H

AJMP DO3

DO3:

CJNE R7,#0DH,DO5

MOV R7,#07H

MOV A,R7

MOVC A,@A+DPTR

CALL LCDP2

INC R7

CJNE R7,#0,RETURN

DO5:

CJNE R7,#09H,DO4

MOV R7,#08H

MOV A,R7

MOVC A,@A+DPTR

CALL LCDP2

MOV R7,#0EH

CJNE R7,#0,RETURN

DO4: DEC R7

MOV A,R7

MOVC A,@A+DPTR

CALL LCDP2

INC R7

CJNE R7,#0,RETURN

MOV R7,#10

MOV A,R7

MOVC A,@A+DPTR

CALL LCDP2

DEC 30H

DEC 30H

MOV R1,#1 //交大的十位

MOV A,30H

MOVC A,@A+DPTR

CALL LCDP2

AJMP RETURN

GAI: MOV R7,#0BH

MOV R1,#1 //交大的十位

DEC 30H

DEC 30H

MOV A,30H

MOVC A,@A+DPTR

CALL LCDP2

MOV R1,#2

INC 30H

AJMP DO

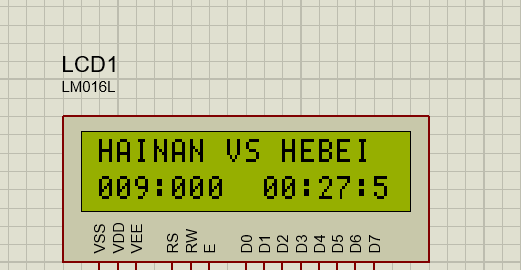
GAI2:

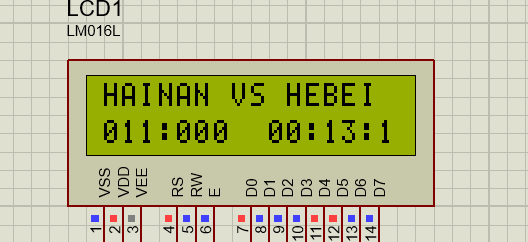
DEC R7

//INC R7

AJMP DO2

此时可以正常实现十位数向个位数的减法运算。



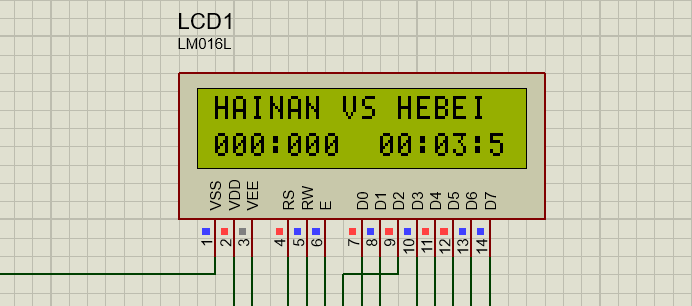


从11减到9和从9减到0均正常。

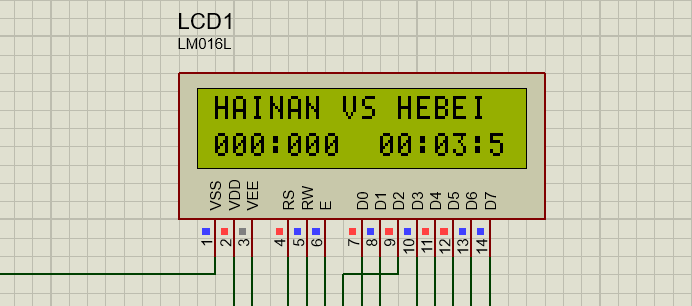
# 系统演示结果

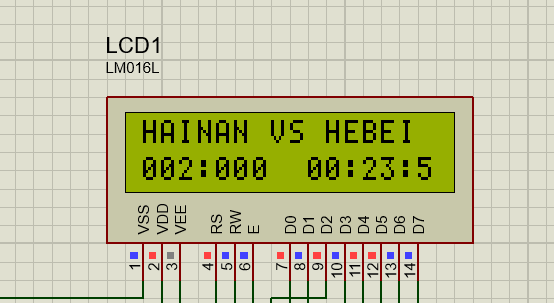
## 5.1重要示例图

### 5.1.1读秒：

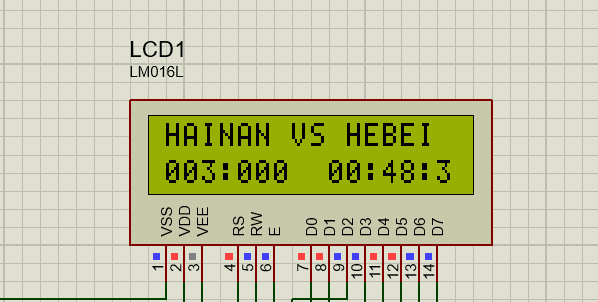
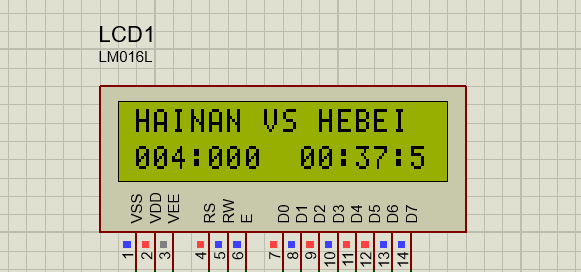


### 5.1.2加计数：



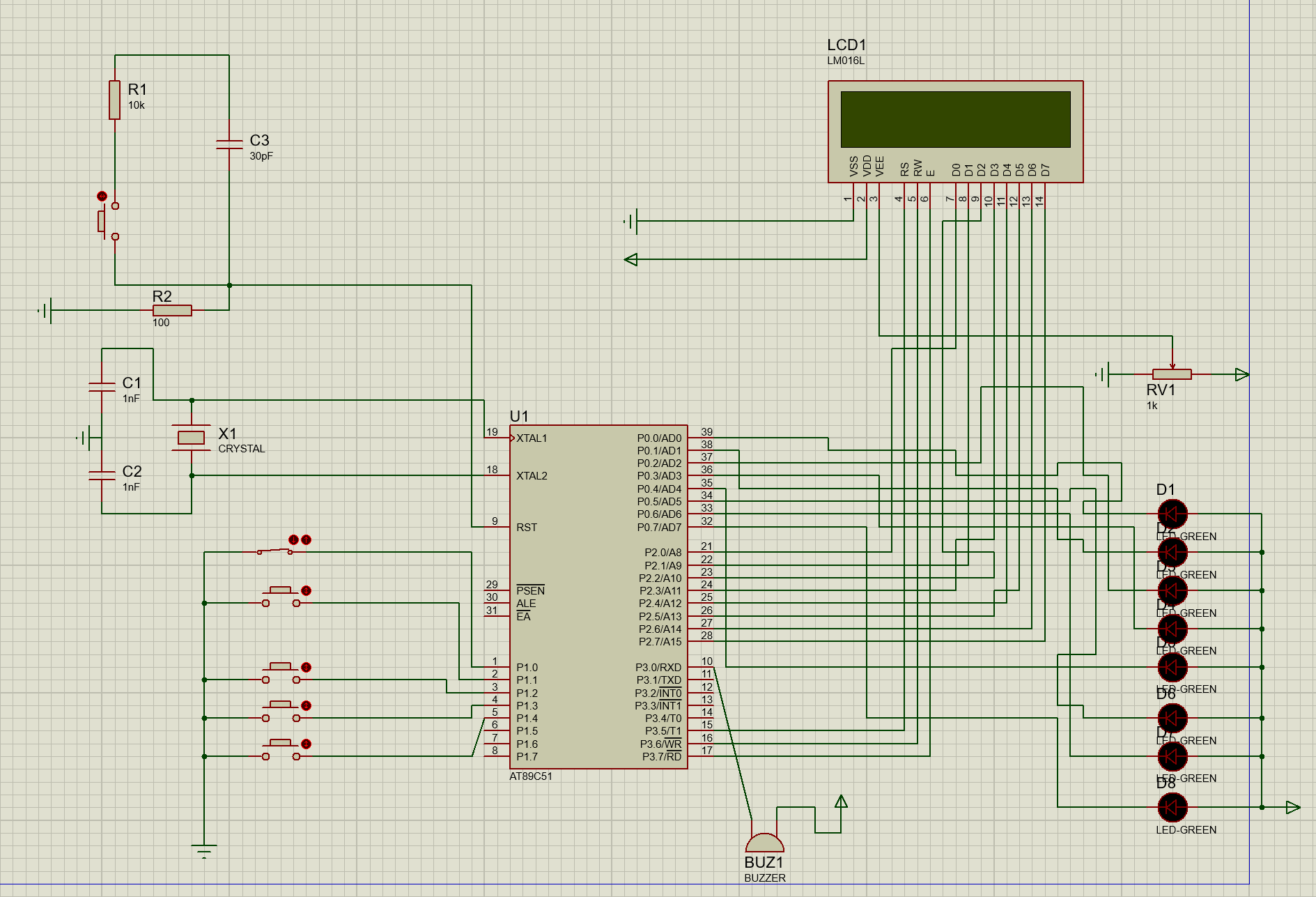


### 5.1.3减计数：



### 5.1.4十位加减计数及十位向个位加减计数（见4.2遇到的问题及解决办法一章）

### 5.1.5电路总图：

****

## 5.2心得体会及尚存在的问题

**心得体会**：

用汇编语言写程序需要时刻注意子程序的位置以及数量，子程序越多所用到的跳转函数就越多程序的可读性就越差，而且我发现用汇编语言写程序时最矛盾的一点就是前面寄存器用的多了会影响后续程序的引用但是用的少了会经常出现寄存器数值在各个子程序中的冲突，例如这次的十位数减法程序就出现了寄存器冲突问题，我不得不使用四个子程序来避免冲突使得越写越乱。而且汇编语言没有两数相等则转移的条件转移指令所以我不得不使用多个子函数将主程序拆开来配合CJNE指令使用感觉非常不方便。

尚存在的问题：

当加法计数到10再加一到11时有时会出现一次按键不起作用的现象，尚未发现解决办法。