# 交通控制系统使用说明

目录

[交通控制系统使用说明 1](#_Toc121787537)

[一、 正常运行流程 1](#_Toc121787538)

[1.正常状态时间 1](#_Toc121787539)

[2.转变说明 1](#_Toc121787540)

[二、 技术说明 1](#_Toc121787541)

[1. 秒数时间的判断 1](#_Toc121787542)

[2. 显示时间函数 3](#_Toc121787543)

[3. 按键处理函数 4](#_Toc121787544)

[4. 主函数 5](#_Toc121787545)

[三、 演示视频 6](#_Toc121787546)

[四、 流程图 7](#_Toc121787547)

## 正常运行流程

### 1.正常状态时间

在正常使用的时候，直行绿灯10秒，左转绿灯10秒，人行横道绿灯10秒，黄灯3秒，红灯20秒，机动车道数码管显示绿灯和红灯的时间。

### 2.转变说明

直行和左转的绿灯的秒数是合在一起为20秒，当到达13秒钟的时候会熄灭直行绿灯，转变为黄灯，过了三秒后会转变为左转绿灯，到达3秒时左转绿灯消灭变成黄灯，结束后会切换90度方向的灯重复此过程。

人行横道的灯只有10秒，10秒过后自动切换成红灯。

## 技术说明

### 秒数时间的判断

为了精确到秒数，这里使用的是定时器0，TH0和TL0的初始值分别为-50000 / 256和-50000 % 256。中断函数为 intt1 ，使用的中断号是1，计数变量名为：count，初始值为0。切换条件为 count==sleep\*20，sleep为自定义的时间，为后面做增加时间和减少时间做铺垫。

为了实现时间的匹配，添加一个全局变量time，time负责倒计时，通过计算得知，1秒=20次count++，所以这里使用count%20，查看余数是不是等于0，如果余数等于0，证明过了一秒。

为了实现黄灯的显示，这里使用了一个变量yellowLED，当时间到达最后三秒时会让yellowLED获得一个状态值。

切换是通过一个标识位flag来就行切换的，其实flag是一个变量，只不过它有不同的内容，所以会表达出不同的内容，这里使用了一个巧妙的方法，用flag异或上1，如果flag为1的时候异或为0，如果是0异或为1。

代码如下：

1. **void** intt1() interrupt 1{
2. TH0 = 15535 / 256;
3. TL0 = 15535 % 256;
4. count++;
5. **if** (count == sleep \* 20){
6. count = 0;
7. yellowLED = 0;
8. leftLED = 0;
9. sleep = button;
10. time = sleep;
11. flag = flag ^ 1;
12. walking\_time = sleep - 10;
13. }
14. **else** **if** (count % 20 == 0){
15. time--;
16. **if** (walking\_time > 0){
17. walking\_time--;
18. }
19. }
20. **if** (count == sleep \* 20 - 60 || count == (sleep / 2) \* 20 - 60){
21. yellowLED = 1;
22. }
23. **if** (count == (sleep / 2) \* 20){
24. leftLED = 1;
25. yellowLED = 0;
26. }
27. }

### 显示时间函数

在这套系统中需要在不同的小系统中调用时间倒计时，所以我把显示封装了。

首先先讲机动车的显示显示出来，之后再显示人行横道的，这里使用全端口处理。

代码如下：

1. **void** showTime(){
2. P0 = number[time / 10];
3. P2 = 0x05;
4. Delay1ms(5);
5. P2 = 0x00;
6. P0 = number[time % 10];
7. P2 = 0x0A;
8. Delay1ms(5);
9. P2 = 0x00;
10. P0 = number[walking\_time / 10];
11. P2 = 0x20;
12. Delay1ms(5);
13. P2 = 0x00;
14. P0 = number[walking\_time % 10];
15. P2 = 0x10;
16. Delay1ms(5);
17. P2 = 0x00;
18. }

### 按键处理函数

三个按键的处理我全部封装到函数中了。

这里有一个问题，需要给P33、P34、P36给高电平，然后才能拿变量去接收结果才行。

忙碌方法我采用的是一个死循环并关闭计算器的方法。

增加时间其实是增加我定义的全局变量button，然后原有时间结束后才能生效。

减少时间和增加时间的逻辑一样。

1. **void** event(){
2. unsigned **char** busy;
3. unsigned **char** add;
4. unsigned **char** subtract;
5. P3\_2 = 1;
6. P3\_6 = 1;
7. P3\_3 = 1;
8. P3\_4 = 1;
9. busy = P3\_6;
10. add = P3\_3;
11. subtract = P3\_4;
12. Delay1ms(20);
13. **if** (busy == 0){
14. ET0 = 0;
15. EA = 1;
16. **do**{
17. Delay1ms(20);
18. P3\_2 = 1;
19. P3\_6 = 1;
20. P2\_7 = 1;
21. busy = P3\_6;
22. Delay1ms(20);
23. }**while**(busy == 1);
24. P2\_7 = 0;
25. ET0 = 1;
26. }
27. **if** (add == 0){
28. Delay1ms(20);
29. button += 1;
30. add = 1;
31. P2\_7 = 1;
32. Delay1ms(200);
33. P2\_7 = 0;
34. }
35. **if** (subtract == 0){
36. Delay1ms(20);
37. button -= 1;
38. subtract = 1;
39. P2\_7 = 1;
40. Delay1ms(200);
41. P2\_7 = 0;
42. }
43. }

### 主函数

为了实现全部功能，我划分为多个子系统。

两个子主系统，两个辅助子系统

第一个子主系统是上下机动车车道红灯左右人行横道绿灯

第二个子主系统是左右机动车车道红灯上下人行横道绿灯

第一个辅助子系统是上下机动车车道绿灯左右人行横道红灯

第二个辅助子系统是左右机动车车道绿灯上下人行横道红灯

在每个子主系统中还有两个辅助系统

这些辅助系统是控制左转车灯的。

1. **void** main(){
2. init();
3. P0 = number[time];
4. P2\_3 = 1;
5. P0 = number[time];
6. **while**(1){
7. **if** (flag == 1 && yellowLED == 0){
8. **if** (leftLED == 0){
9. P1 = 0x82;
10. P3 = 0x81;
11. }
12. **else** **if** (leftLED == 1){
13. P1 = 0x89;
14. P3 = 0x80;
15. }
17. showTime();
18. event();
19. }
20. **else** **if** (flag == 0 && yellowLED == 0){
21. **if** (leftLED == 0){
22. P1 = 0x28;
23. P3 = 0x02;
24. }
25. **else** **if** (leftLED == 1){
26. P1 = 0x98;
27. P3 = 0x00;
28. }
29. showTime();
30. event();
31. }
32. **if** (flag == 1 && yellowLED == 1){
33. P3 = 0x81;
34. P1 = 0x84;
35. showTime();
36. event();
37. }
38. **else** **if** (flag == 0 && yellowLED == 1){
39. P3 = 0x02;
40. P1 = 0x48;
41. showTime();
42. event();
43. }
44. }
45. }

## 演示视频

运行视频



按键视频



## 流程图

Button--

是否按P34按键

Button++

是否按P33按键

是

是否按P36按键

关闭中断

是

是否为P36按键

按键处理

中断执行

红灯

左转绿灯

黄灯

直行绿灯