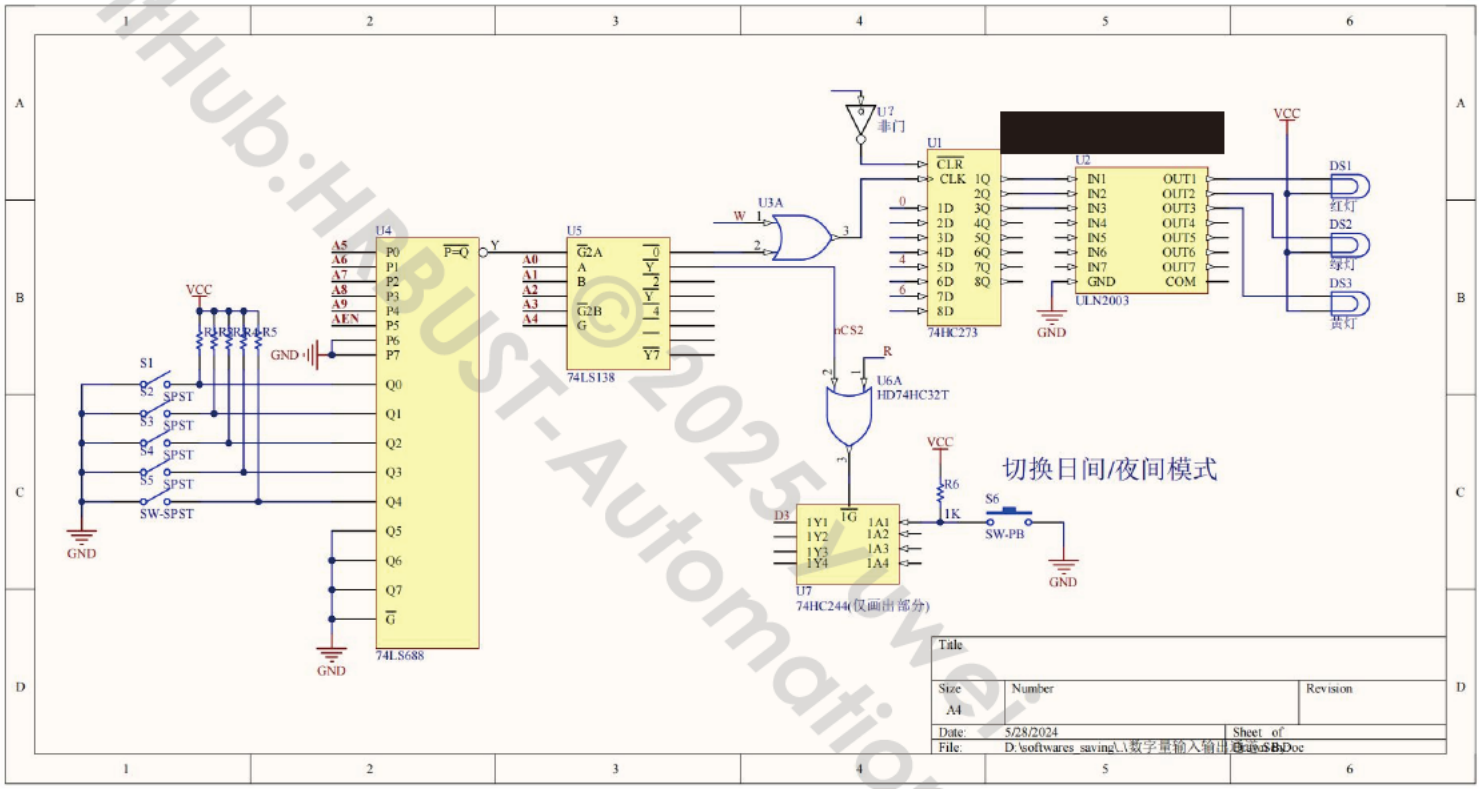
设计交通灯控制系统作业

电路图



电路设计说明

①驱动放大电路的设计

驱动放大电路具有驱动放大、信号隔离、电平转换的作用，我们在课堂上学过用晶体管、固态继电器、达林顿驱动器来构成驱动电路，在本次设计的交通灯控制系统中，我选择使用达林顿驱动器来驱动三个交通灯。达林顿驱动器具有高电流放大能力，这对需要较大电流的交通灯来说尤为重要，能够确保交通灯稳定亮起。此外，达林顿驱动器的内部电路结构简单，便于设计和实现，且其高稳定性能够保证交通灯在各种工作条件下可靠运行。

相比之下，晶体管驱动器虽然可以实现类似功能，但单个晶体管的电流放大能力有限，可能无法满足交通灯的电流需求，且需要额外的电路设计来增强其驱动能力。固态继电器虽然具有电气隔离和零功率开关等优点，但它们的开关速度较慢，成本较高，不太适合需要频繁切换和成本敏感的交通灯系统。因此，综合考虑驱动能力、设计简便性和系统稳定性，达林顿驱动器是更适合交通灯控制系统的选择。

②输出锁存器的选择

在本次设计的交通灯控制系统中，我选择使用 74HC273 锁存器来保留数据，这是因为 74HC273 锁存器具有高稳定性，能够可靠地保持输入数据，不受外部干扰的影响。此外，74HC273 锁存器的数据保持能力强，一旦输入数据被接收并锁存，即使输入信号消失，锁存器仍然能够保持数据不变，这对于需要长时间保持状态的交通灯控制系统来说至关重要。而且，74HC273 锁存器的控制信号简单明了，通常由时钟信号和控制使能信号控制，使得锁存器易于集成到系统中，并方便进行调试和维护。因此，基于稳定性高、数据保持能力强和易于控制等优点，我选择使用 74HC273 锁存器来设计交通灯控制系统，以确保交通灯状态的稳定性和可靠性。

③地址译码+控制逻辑的设计

在设计交通灯控制系统的地址译码和控制逻辑电路部分时，我首先考虑了使用 5 位拨码开关来实现 10 位地址总线中的 A5 到 A9 的可变地址。然后，我通过数字比较器来比较拨码开关的输出和地址总线的结果是否一致，将比较器的输出接到了 74LS138 译码器的门控端 G2A，以选择通路。接着，我将 A0 到 A3 分别与 74LS138 的 ABC 码相连，将 A3 连接到门控端 G2B，A4 连接到门控端 G，以便进行地址译码。为了让 74HC273 的初始地址为 0x230，我设置 A3A2A1A0 为 0000，A7A6A5A4 为 0010，A9A8 为 10，这样只需调整拨码开关的第 2、3、4 个合上，第 1、5 个打开即可满足条件。通过这种方式，我完成了对 74HC273 的地址分配。

此外，为了方便切换日间和夜间两种模式，我额外增加了一个数字量输入通道——按键。我在按键的输入口放置了一个三态缓冲器，并将缓冲器的片选接到了 138 译码器的 Y1 输出，使得按键的地址为 0x232。这样设计既保证了系统的灵活性和易操作性，又满足了交通灯控制系统的各项要求。

电路工作原理解释

在整个电路的工作过程中，首先，拨码开关被设置为指定的位置，并且将 74HC273 锁存器的地址写入 PC 总线。接着，数字比较器比较拨码开关输出和地址总线上的结果是否一致，若一致，则使能 74LS138 译码器。当 CPU 发送地址时，经过译码后，nCS1 信号有效为 0，CPU 将数据送到数据总线上，并使能写信号，此时 nIOW 信号为 0，同时将 CLK 引脚置为 0。在数据写入完成后，CPU 取消写信号，nIOW 信号变为 1，此时 74HC273 的 CLK 引脚产生上升沿，将数据总线上的数据锁存到达林顿驱动器中。达林顿驱动器放大数据并驱动交通灯的亮灭。此外，可以通过选通缓冲器 74HC244 获取按键的键值，按键用于切换交通灯系统的夜间模式和日间模式。

在整个过程中，拨码开关和数字比较器起到了地址译码的作用，确保了正确的地址被选通。CPU 通过发送地址和数据到数据总线上实现对 74HC273 锁存器的写入操作，而 CLK 引脚的上升沿则触发了数据的锁存操作。锁存器锁存的数据被达林顿驱动器放大并用于驱动交通灯的控制。同时，缓冲器的选通使得按键的状态可以被读取，从而实现了交通灯系统模式的切换。

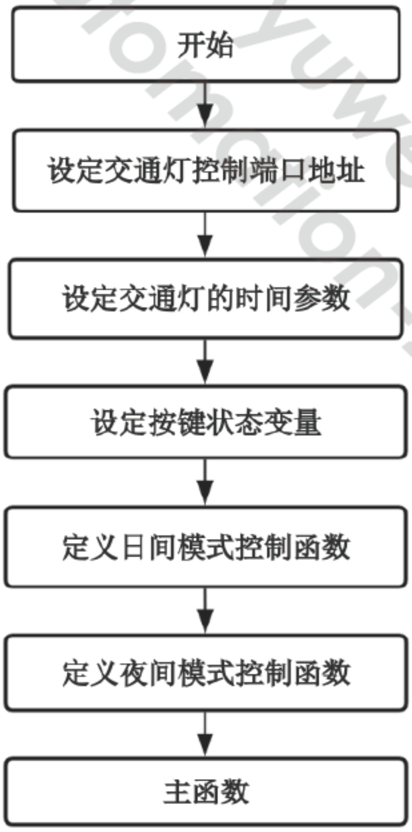
程序设计思路

首先，程序定义了交通灯控制端口地址和按键读取端口地址，并设置了日间模式和夜间模式的常量。接着，定义了绿灯、黄灯、红灯以及夜间模式黄灯闪烁的时间参数，以及一个用于读取按键状态的变量。时间参数用了 X，Y，Z 来替代，具体使用时可以替换成期望的时间。

程序中定义了两个主要函数，分别用于日间模式和夜间模式的控制。在日间模式下，程序循环执行绿灯、黄灯、红灯的亮灭控制，并通过延时函数实现时间间隔的控制。在夜间模式下，程序循环执行黄灯的闪烁控制，同样通过延时函数实现闪烁间隔的控制。

在主函数中，程序通过不断读取按键的状态来判断当前是处于日间模式还是夜间模式。如果按键状态为 1，表示切换到夜间模式；否则，保持日间模式。然后根据当前模式执行对应的控制函数。

整个程序的设计思路比较简单，通过不断地读取按键状态来实现日间模式和夜间模式的切换，并在每种模式下调用相应的控制函数来实现交通灯的亮灭控制。同时，程序中使用了延时函数来控制各种灯的亮灭时间间隔，使得交通灯的控制更加精确和稳定。下图是以流程图的形式来说明。




```c
#include <dos.h>
#include <stdio.h>

#define TRAFFIC_LIGHT_ADD 0x230       // 交通灯控制端口地址
#define KEY_ADD 0x232            // 按键读取端口地址
```

```
6.
7.  #define DAY_MODE 0     // 日间模式
8.  #define NIGHT_MODE 1     // 夜间模式
9.
10. // 亮灯时间设置
11. uint32_t green_time = x;                    // 绿灯时间，单位秒
12. uint32_t yellow_time = y;                   // 黄灯时间，单位秒
13. uint32_t red_time = z;                      // 红灯时间，单位秒
14. uint32_t blink_time = 1;                    // 夜间模式黄灯闪烁间隔时
    间，单位秒
15.
16. // 日间模式控制
17. void day_mode() {
18. while (true) {
19. IOW(TRAFFIC_LIGHT  0x02); // 绿灯亮
20.         delay(green_time * 1000);
21.         IOW(TRAFFIC_ IGHT, 0x03);  // 黄灯亮
22.         delay(yellow_time * 1000);
23.         IOW(TRAFFIC_LIGHT  0x01);   // 红灯亮
24.         delay(red_time * 1000);
25.     }
26. }
27.
28. // 夜间模式控制
29. void night_mode() {
30.     while (true) {
31.         IOW(TRAFFIC_LIGHT, 0x03);  // 黄灯亮
32.         delay(blink_time * 1000);
33.         IOW(TRAFFIC_LIGHT, 0x00);  // 黄灯亮
34.         delay(blink_time * 1000);
35.     }
36. }
37.
38.
39.
40. int main() {
41.     int mode = DAY_MODE;          // 初始化为日间模式
42. uint8_t key_value;                 //定义键值
43.
44. While(1)
45. {
46. key_value = IOR(KEY_ADD);          //读取键值
47.  if (key_value = = 1)                    //如果按键被按下就切换成夜间模
    式
```

```
48.   {
49.       mode = NIGHT_MODE;
50.   }
51.   else                                                    //否则继续维持日间
      模式
52.   {
53.   mode = DAY_MODE;
54.   }
55.
56.   if (mode == DAY_MODE)                 //如果是日间模式，执行日间模式下的函数
57.    {
58.          day_mode();
59.   }
60.   else if (mode =  NIGHT_MODE)                  //如果是夜间模式，执行夜间模式
      下的函数
61.   {
62.          night_mode();
63.      }
64.
65.   }
66.      return 0;
67.   }
```