

武汉大学计算机学院

本科生实验报告

基于 AT89C51 的电梯控制系统

专 业 名 称 : 计算机科学与技术

课 程 名 称 : 嵌入式系统

二〇二〇年四月

郑 重 声 明

本人呈交的设计报告，是在指导老师的指导下，独立进行实验工作所取得的成果，所有数据、图片资料真实可靠。尽我所知，除文中已经注明引用的内容外，本设计报告不包含他人享有著作权的内容。对本设计报告做出贡献的其他个人和集体，均已在文中以明确的方式标明。本设计报告的知识产权归属于培养单位。

本人签名：_____ 日期：_____

摘 要

采用 AT89C51 单片机实现双电梯调度系统，共有 8 层楼，每层楼都有单独的上行和下行按钮，每个按钮都有对应的 LED 灯标识按钮是否按下，有较好的显示效果。

关键词：AT89C51、双电梯调度

1 实验目的和意义

1.实验目的

通过设计一个基于单片机的电梯控制系统，增加学生对单片机工作原理的理解，掌握嵌入式系统设计的基本步骤。学习并掌握 Proteus 和 Keil 软件的使用方法，借助 C 语言进行编程实现多部电梯调度，掌握嵌入式系统开发的基本流程，激发学生对嵌入式系统设计的兴趣。

2.实验意义

本实验通过让学生基于单片机设计一个电梯调度程序，具有以下意义：

1. 增强学生对 51 系列单片机工作原理的理解
2. 掌握使用 Proteus 和 Keil 软件进行设计和仿真的流程
3. 了解嵌入式系统开发的基本流程
4. 激发学生对嵌入式系统的兴趣

2 实验设计

1.原理图

本实验基于单片机实现两部电梯调度系统，其中利用 Proteus 设计的硬件电路图如下图所示：

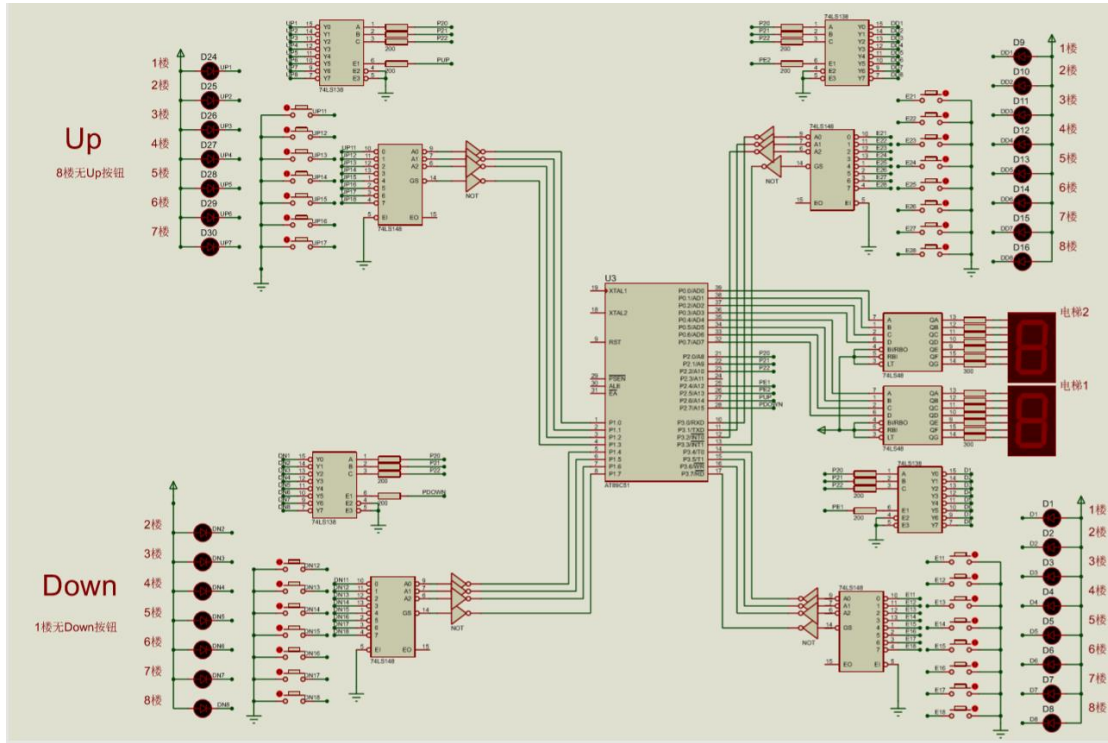


图 1 硬件原理图

其中我设计的电梯调度系统中有两部电梯，每部电梯中有 8 层楼（1 楼~8 楼），每层楼有上升和下降两个按钮（最底层无向下按钮，最高层无向上按钮）。由于系统设计较为复杂，我们在下面一一讲解。

1.1 数码管显示电路

由于电梯调度系统中有两部电梯，所以我们需要两个数码管显示每部电梯现在所处的楼层，使用 74LS48 进行作为数码管的驱动器，可以得到电路连接如下：

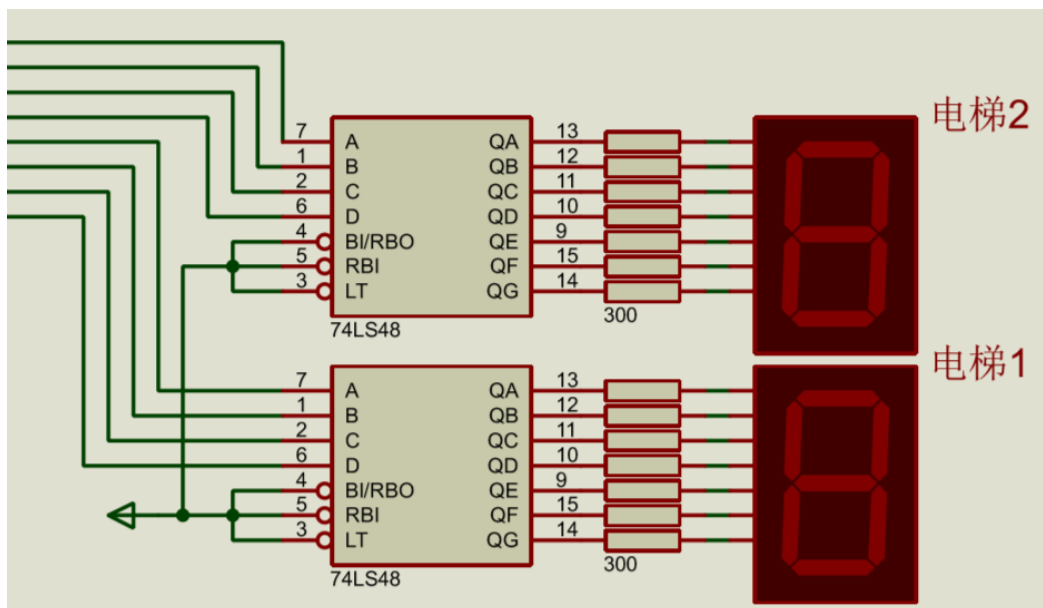


图 2 数码管连接电路

根据 74LS48 的真值表可知，BI/RBP、RBI、LT 置高电平时才可以正常工作，所以我们讲这三个端口置高电平，其余的 A、B、C、D 端口做输入端与 AT89C51 相连，QA~QG 端口做输出端，与数码管相连。

1.2 电梯按钮电路

每个电梯有 8 层楼可供选择，每层电梯都有一个按钮和一个指示灯，指示灯用来指示当前楼层是否被按下，电路图设计如下：

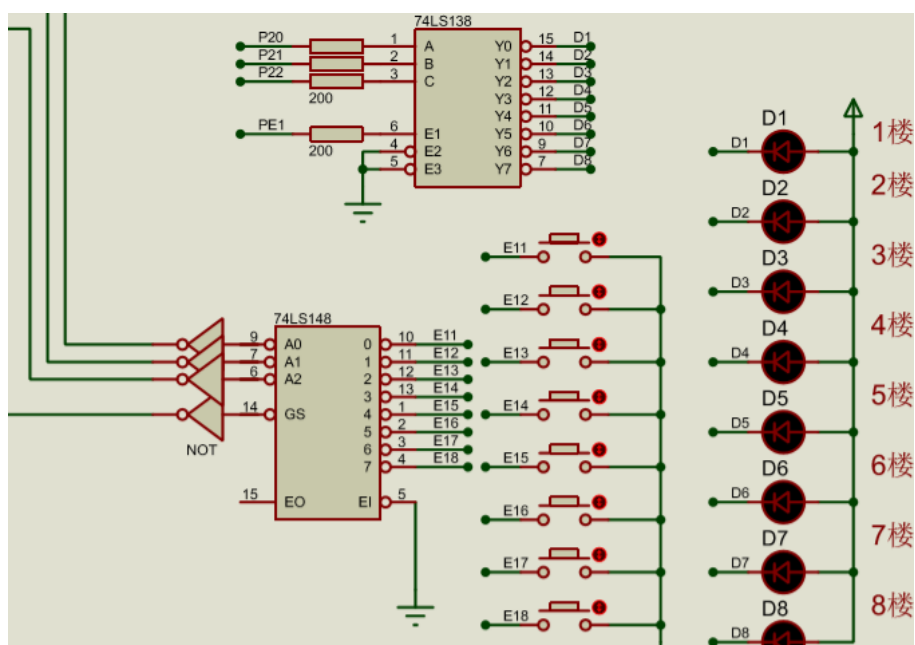


图 3 电梯按钮电路

由于一共有两部电梯，每部电梯有 8 层楼，每层楼有按钮和指示灯两个 I/O 端口，所以 AT89C51 仅有的 32 个 I/O 端口明显不够使用，所以我们使用 3/8 译码器和 8/3 译码器对信号进行编码和解码，以此提高可用的 I/O 端口数量。

由于 AT89C51 单片机能够提供的电流十分有限，所以我们的 LED 灯需要使用共阳极的方式进行连接，为了避免单片机被烧在“LED 灯泡-74LS138-AT89C51 单片机”电路上增加 200 欧限流电阻。

其余电梯和上行按钮、下行按钮的电路图设计和该电路图设计除了与单片机连接的 I/O 端口不一致外其余完全相同，再次不在赘述。

2.功能设计分析

2.1 按钮与 LED 灯功能设计

按钮和 LED 电路的设计实现的功能是当某层楼的按钮按下后，对应的 LED 灯亮起，首先我们观看 74LS138 译码器和楼层 LED 灯的连接电路：

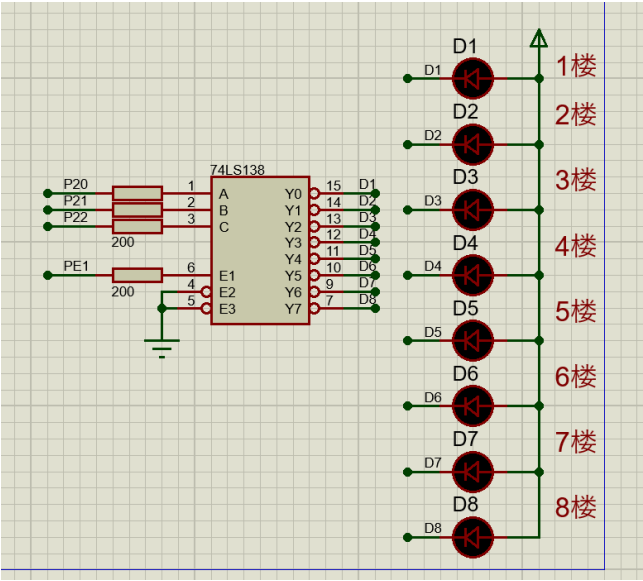


图 4 74LS138 与 LED 灯连接电路

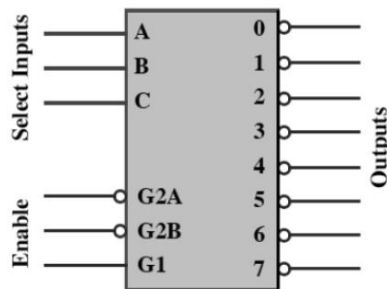
由于 LED 全部按照共阳极的解法接入电路，所以当 LED 灯左侧输出为高电平时对应的 LED 灯不亮，输出为低电平的时候对应的 LED 灯亮。如下表所示：

表格 1 LED 灯与左侧电平对应状态表

LED 灯左侧电平状况	LED 状况
高电平	不亮
低电平	亮

PS: 由于我们使用 3/8 译码器和 LED 灯连接，这就意味着我们不可能同时让两个灯都处于亮的状态，所以我们必须使用类似流水灯的处理方式，让需要亮的灯依次亮并增长每个灯亮的时间来达到同时亮的效果。

74LS138 是 3/8 译码器，它的真值表如下图所示：



Inputs						Output							
Enable			Select										
G2A	G2B	G1	C	B	A	0	1	2	3	4	5	6	7
1	X	X	X	X	X	1	1	1	1	1	1	1	1
X	1	X	X	X	X	1	1	1	1	1	1	1	1
X	X	0	X	X	X	1	1	1	1	1	1	1	1
0	0	1	0	0	0	1	1	1	1	1	1	1	1
0	0	1	0	0	1	1	0	1	1	1	1	1	1
0	0	1	0	1	0	1	1	0	1	1	1	1	1
0	0	1	0	1	1	1	1	1	0	1	1	1	1
0	0	1	1	0	0	1	1	1	1	0	1	1	1
0	0	1	1	0	1	1	1	1	1	1	0	1	1
0	0	1	1	1	0	1	1	1	1	1	1	0	1
0	0	1	1	1	1	1	1	1	1	1	1	1	0

图 5 74LS138 真值表

由真值表我们可以知道 G2A、G2B、G1 即原图中 (E1、E2、E3) 是使能位，当使能位满足的时候，由 CBA 对应电平状态构成的二进制数就是输出为低电平的端口，如：CBA 为 001 的时候，端口 1 输出为低电平，所以我们讲端口 0~7 分别与楼层 1~8 的 LED 灯相连，当 CBA 为 001 时，端口 1 输出低电平，即楼层 2 的 LED 灯亮，可以满足我们的需求。

同时我们可以注意到，CBA 输入就是 000~111 即数字 0~7，对应的 LED 灯是否亮可以取决于使能位，我们可以取 E1 作为按钮对应使能位，由于有 8 个按钮，所以我们可以使用一个 int 类型的数据来表示，每一位表示一个按钮，在 Keil 中使用类似下面的代码实现对应的灯亮：

```

1  for(int i=0;i<8;i++){
2      CBA=i;
3      if(a的第i位为1)
4          E1=1;
5      else
6          E1=0;
7  }
```

所以我们如果要想实现通过按钮控制对应楼层的 LED 灯亮，只需要检测在按下某个按钮的时候置对应的 int 类型数据(上述代码中的 a)的对应位为 1 即可，

如：第 1 层按钮按下，则 a 的二进制形式为 00000001，在上述代码执行的过程中就会将第 1 层(但是与 74LS138 第 0 个端口连接)的 LED 灯点亮。

所以按钮电路需要做的就是当某个按钮按下后给单片机传入一个对应的信息表示该按钮被按下，所以我们看一下按钮电路的设计：

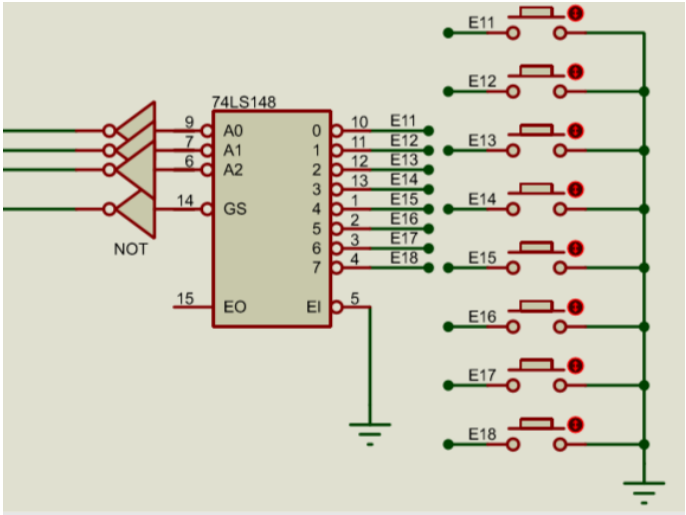


图 6 按钮电路设计

74LS148 对应的真值表如下所示：

INPUTS									OUTPUTS				
EI	0	1	2	3	4	5	6	7	A2	A1	A0	GS	EO
H	X	X	X	X	X	X	X	X	H	H	H	H	H
L	H	H	H	H	H	H	H	H	H	H	H	H	L
L	X	X	X	X	X	X	X	L	L	L	L	L	H
L	X	X	X	X	X	X	L	H	L	L	H	L	H
L	X	X	X	X	L	H	H	H	L	H	L	L	H
L	X	X	X	L	H	H	H	H	H	L	L	L	H
L	X	X	L	H	H	H	H	H	H	L	H	L	H
L	X	L	H	H	H	H	H	H	H	H	L	L	H
L	L	H	H	H	H	H	H	H	H	H	H	L	H

H = high logic level, L = low logic level, X = irrelevant

图 7 74LS148 的真值表

按照上图的接法，当某个按钮按下后，74LS148 对应的输入端置为低电平，对应一个输出状态，如：E11 被按下后，A2A1A0 为 111，取反后为 000，即 0；E12 被按下后，A2A1A0 为 110，取反后为 001，即 1。所以我们将 74LS148 的 A2A1A0 输出取反后得到的就是对应的被按下的按钮的位置，将这个位置输送给单片机，

单片机内记录这个位置并在数码管显示处理的时候使用这个数据即可实现通过按钮点亮 LED 灯。

最后我们看一下总的电路图：

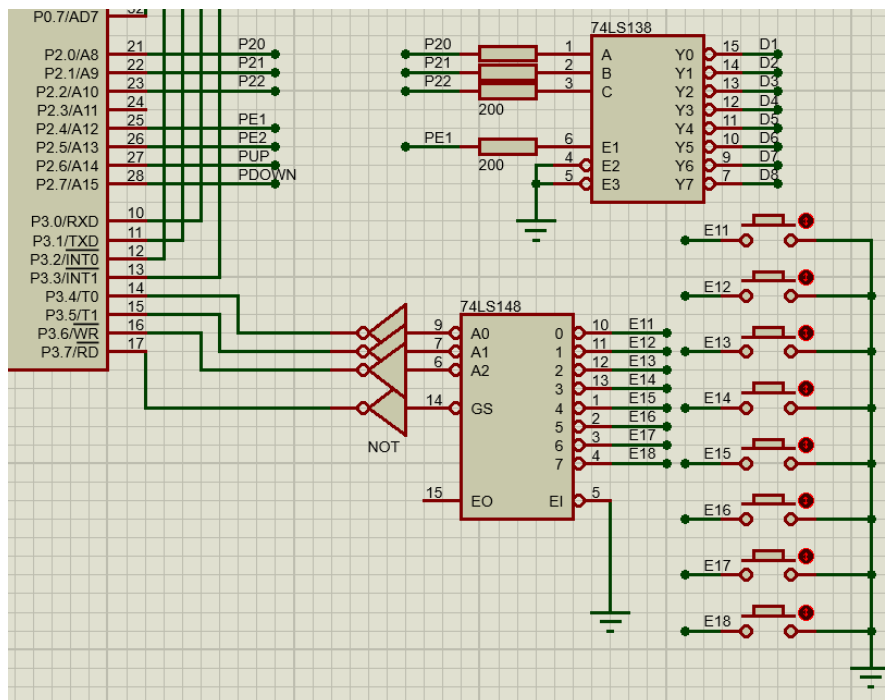


图 8 按钮与 LED 灯电路图设计

对应的 Keil 代码如下：

```

1 void display_LED(){
2     int i=0;
3     //依次点亮八个灯
4     while(i<8){
5         P2=i;
6         pe1=(e1>>i)&1==1;//如果第i位为1，则对应的灯亮
7         i++;
8         delay(50);
9     }
10 }
11 //捕获按钮
12 void update_click(){
13     //e1电梯按钮
14     if(p37){
15         e1|=(1<<((p36*4)|(p35*2)|(p34)));
16     }
17 }
18 }

```

用 e1 记录该电梯按下的楼层，然后再显示 LED 灯的时候检测 e1 的对应位是否为 1，如果是 1 就将 E1 置为 1 对应的 LED 灯亮；否则置 0，对应的 LED 灯不亮。

其余三个按钮和 LED 灯的电路图设计与该电路图完全相同，而且由于在显示 LED 灯的时候这 4 个 74LS148 的 CBA 端口输入都是 000~111，所以我们可以将这 4 个 74LS138 的 C 端口接到同一个端口中，B 和 A 端口也是，这样做可以减少使用的端口数。

2.2 数码管显示电路

数码管电路比按钮和 LED 灯电路简单许多，数码管电路的设计如下：

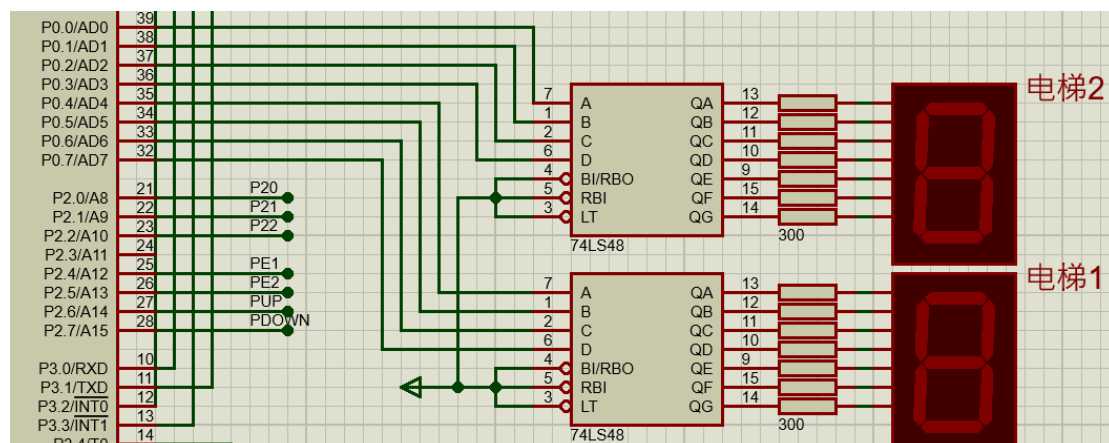


图 9 数码管电路设计

我在 Keil 的代码设计中使用两个变量表示两个电梯所处的位置，因为两个数码管通过 74LS48 与单片机的 P0 引脚相连，所以我只需要将 P0.0~P0.3 置为第二个电梯的楼层数，P0.4~P0.7 置为第一个电梯的楼层数，数码管就会显示对应的楼层数，在 Keil 中实现方式类似如下：

```
1 void display_Elevator(){
2
3     P0=E1eTwo+E1eOne*16;
4 }
```

将第一个电梯的楼层数左移 4 位加上第二个电梯的楼层数，最后赋给 P0 端口即可。

2.程序实现流程

2.1 点亮 LED 灯程序

根据上面的解释可知：

1. 四个 74LS138 的 CBA 三个端口共用 P2 的低三位

2. 每个灯是否亮由使能位 E1 表示，这个值由每个按钮序列的 int 数表示
所以我们可以得到点亮 LED 灯程序的代码如下：

```
1 void display_LED(){
2
3     int i=0;
4     //依次点亮八个灯
5     while(i<8){
6         P2=i;
7         pe1=(e1>>i)&1==1;//如果第i位为1，则对应的灯亮
8         pdown=(down>>i)&1==1;
9         pup=(up>>i)&1==1;
10        pe2=(e2>>i)&1==1;
11        i++;
12        delay(50);
13    }
14
15 }
```

其中 e1、e2、up、down 分别为电梯 1、电梯 2、上升按钮序列、下降按钮序列的用来表示状态的 int 数。

2.2 按键检测程序

由 74LS148 的真值表可知当无按键按下和按键 0 按下的时候输出端口都是 000，这样就无法区分无按键按下和按键 0 被按下；再观察后发现当有按键按下到
时候 GS 处于低电平，当无按键按下的时候 GS 处于高电平，因为我们对 GS 输出
做了取反处理，所以我们可以根据 GS 的状态区分有按键按下和无按键按下，然
后根据 A2A1A0 的输出确定是哪个按键被按下，以此来区分无按键按下和按键 0
被按下。所以 Keil 代码如下：

```
1 //捕获按键
2 void update_click(){
3     //e1电梯按键
4     if(p37){
5         e1=(1<<((p36*4)|(p35*2)|(p34)))&1;
6     }
7     //down电梯按钮
8     if(p17){
9         down=(1<<((p16*4)|(p15*2)|(p14)))&1;
10    }
11    //up的按钮
12    if(p13){
13        up=(1<<((p12*4)|(p11*2)|(p10)))&1;
14    }
15    //e2按钮
16    if(p33){
17        e2=(1<<((p32*4)|(p31*2)|(p30)))&1;
18    }
19 }
```

其中 p37、p17、p13、p33 分别为四个 74LS138 的 GS 所连接的单片机接口，即用来判断是否有按键按下，只有有按键按下时才更新 int 数。

为了保证能够记录所有被按下的按钮，所以我们的 int 数需要或运算而不是与运算。

2.3 更新 LED 灯状态

我使用一个 int 类型的数据表示 8 个按钮的按下状况，所以我们在电梯到达某一层的时候更新 LED 灯就是对这个 int 类型数据进行更新，相关的 Keil 代码如下：

```
1 void update_LED(){
2
3     if(EleOneState==1){
4         //处于上升状态
5         up&=~(1<<(EleOne-1));
6         e1&=~(1<<(EleOne-1));//第EleOne层置为0
7         //是为了接到最高楼层的向下人员而处于的向上状态,而且当前楼层是向下人员中的最高层
8         if((e1>>EleOne)<=0&&(up>>EleOne)<=0&&(down>>EleOne<=0)){
9             down&=~(1<<(EleOne-1));
10        }
11
12    }else if(EleOneState==2){
13
14        down&=~(1<<(EleOne-1));
15        e1&=~(1<<(EleOne-1));
16        //为了接到最低楼层的向上人员,而且当前楼层是向上人员中的最底层
17        if(downFloor(&e1,&EleOne)<=0&&downFloor(&down,&EleOne)
18        <=0&&downFloor(&up,&EleOne)<=0){
19            up&=~(1<<(EleOne-1));
20        }
21    }
22 }
```

我使用 EleOneState 标识电梯 1 所处的状态，其中 0 表示静止，1 表示上升状态，2 表示下降状态。针对电梯处于上升状态来说需要以下三种更新：
(其中 e1 是电梯 1 的按钮 int 数，up 和 down 分别是上升按钮和下降按钮的 int 数，且假设当前电梯 1 处于第 i 层)

1. e1 的第 i 位置 0：无论 e1 的第 i 位之前为什么，现在电梯已经到达第 i 层所以 e1 的第 i 位都应该置 0，表示请求已经完成。

2. up 的第 i 位置 0：无论之前 up 的第 i 位为什么，现在电梯已经到达第 i 层，原来在第 i 层的人进入电梯，第 i 层的向上请求已经完成，所以应该将 up 的第 i 位置 0。

3. down 的第 i 位置 0: 这种更新是有条件的, 即 e1、up、down 的比 i 高的位都为 0, 即比 i 高的楼层没有请求, 而且当前楼层有向下请求, 所以此时应该转换方向并响应该请求。

当电梯处于向下状态时对应的也有三种更新情况, 但是情况和向上状态类似, 这样就可以做到当电梯到达某楼层时响应相应的请求, 灭掉相应的请求灯, 模拟实现电梯运动。

2.4 电梯调度

2.4.1 单电梯调度算法

电梯调度是 Keil 代码的核心, 首先我从单个电梯调度入手, 按照以下规则进行实现:

1. 处于上升状态时只处理向上的请求, 处于向下状态时只处理向下的请求
2. 当有多个向上请求时, 先下降到请求调度的最底层再响应向上请求; 当有多个向下请求时, 先上升到请求调度的最高层再响应向下请求

由于只考虑单电梯调度算法, 所以电梯 1 和电梯 2 的调度算法完全相同, 下面我们以电梯 1 的调度算法为例进行解释:

当电梯处于静止状态时处理的流程图如下:

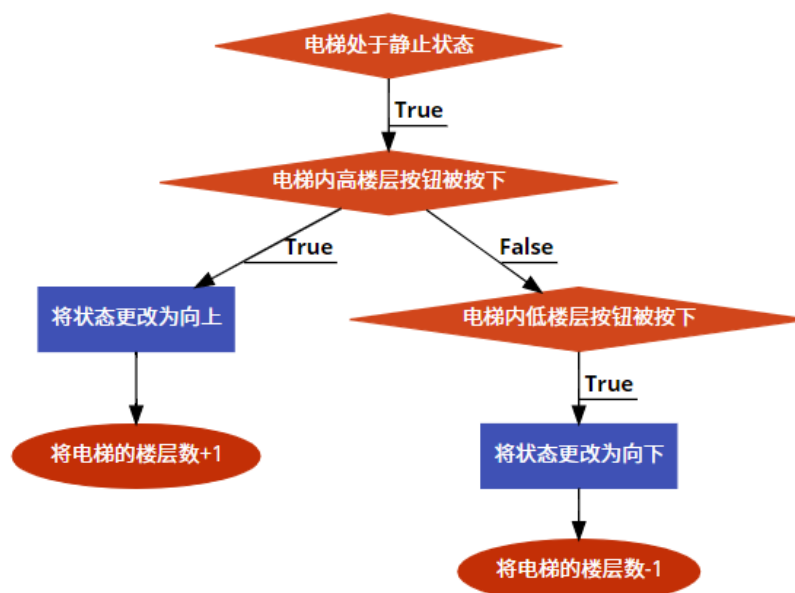


图 10 电梯处于静止状态时的处理流程

当电梯处于向上运动状态时，处理的流程图如下：

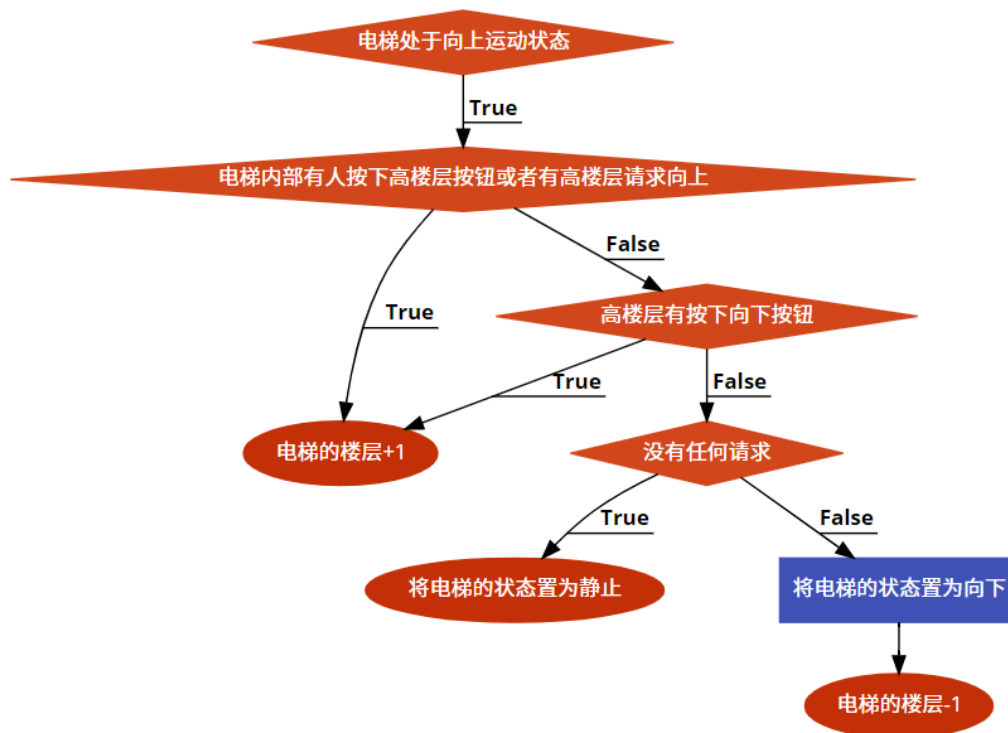


图 11 电梯处于向上状态时的处理流程

其中第一个分支遵循“向上状态只响应向上请求”原则，这种向上请求分为：

1. 电梯内部有人向上；
2. 高楼层有人按下向上按钮。

在这两种状态下电梯仍然需要保持向上状态；如果这两种状态不存在也就意味着电梯不需要再向上，可以转换方向了；第二个分支遵循“如果有多个楼层请求向下则先上升到最高楼层再向下”，如果高楼层有人请求向下我们仍然需要保持向上状态，但是这时的向上是为了到达请求向下楼层中的最高层，上升的原因改变了。如果高楼层没有向下请求，我们就查看是否低楼层有相关请求，如果有就让电梯向下，如果没有就将电梯置为静止状态。

当电梯处于向下状态的时，处理的流程图如下：

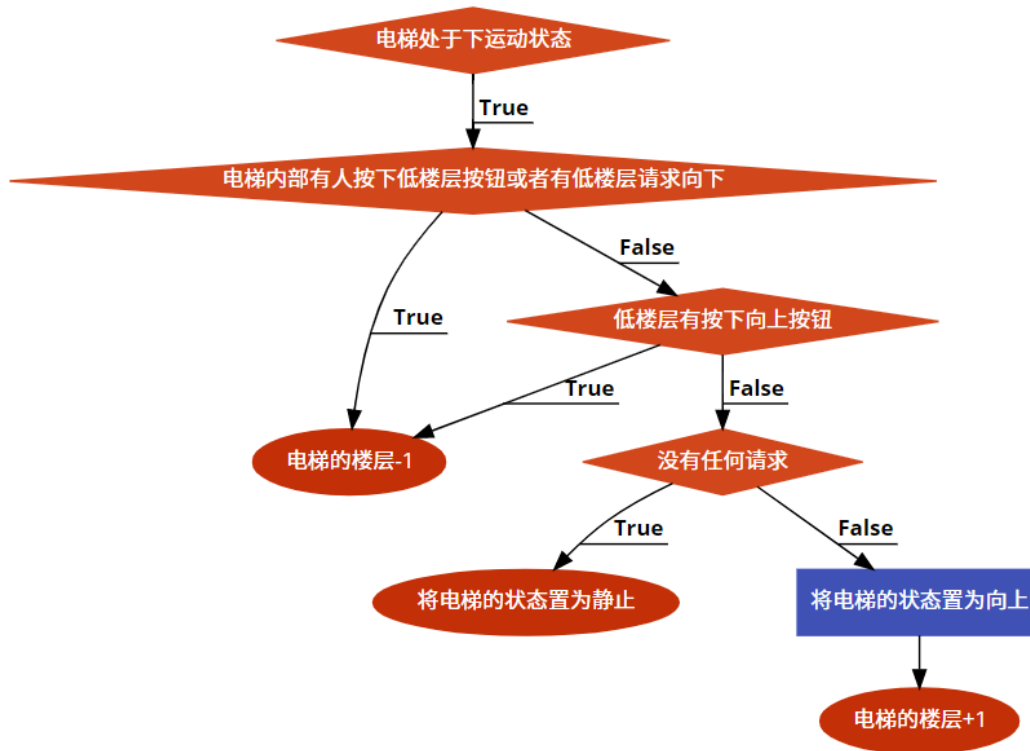


图 12 电梯处于向下状态时的处理流程

电梯处于向下状态时处理流程和向上处理流程类似，遵循的原则也相似，在这里不在赘述。

2.4.2 双电梯调度

双电梯调度以单电梯调度为基础，然后在单电梯调度的基础上进行完善。

①当两个电梯都处于静止状态时，电梯调度算法如下：

1. 处于高楼层的电梯处理比自己更高楼层的请求
2. 处于低楼层的电梯处理比自己更低楼层的请求
3. 如果 1 和 2 经过之后仍然静止，查看两个电梯之间是否有请求，如果有就使电梯同时往请求靠近；如果没有就仍然处于静止状态

②当一个电梯静止，一个电梯不静止时，电梯调度的算法如下：

1. 运动的电梯处于向上状态，则静止电梯处理比自己所处楼层低的楼层发出的向上请求
2. 如果运动电梯处于向下状态，则静止电梯处理比自己所处楼层高的楼层发出的向下请求

3 遇到的问题和感受

1.遇到的问题

在设计的过程中遇到了很多的问题，其中比较有代表性的有下面几个：

①LED 灯闪烁速度过快，无法常亮。因为需要模拟多层按钮被按下的情况，所以我们需要将多个 LED 灯同时亮，但是由于使用了译码器所以只能通过流水灯的效果达到几个 LED 灯同时亮的效果。但是在设计的时候我发现由于每个灯泡亮的时间太短，所以还是 LED 灯亮灭还是有明显的先后，在查阅资料后发现 LED 灯有一个属性：Minimun on time to light，只要修改这个属性即可。

②不知道怎么表示每个按键的状态。由于 I/O 端口的数目限制，我不知道该如何设计表示每个按键的状态，后来在咨询赵天浩之后我才知道用一个 int 来表示每个按键的状态，使用使能位进行控制。感谢赵天浩的无私帮助，同时我也明白了在遇到问题时要懂得寻求他人的帮助，合作交流也许会产生不一样的效果。

③电梯调度程序难下下手。在电梯调度设计的过程中，刚开始的时候我在网上查阅资料之后考虑了很多规则，但是考虑的都是两个电梯对每个响应都根据时间代价进行设计，后来发现好像进入死胡同了，怎么都设计不出来。后来我改变了思路，首先设计每个电梯的单独调度算法然后设计两个电梯同时进行调度的算法，虽然我设计的算法仍然有很多的缺陷，但是我相信这种分解问题的思路在以后解决问题的过程中仍然会很有用。

2.感受

这次嵌入式课程设计可以算是第一次真正意义上的自己设计并仿真一个嵌入式系统，从软件如何使用、硬件原理图如何实现、调度算法如何编写，这之中遇到了很多的困难和问题，但是通过同学的帮助和网上查询最终也解决了这些问题，而且顺利实现了两部电梯的调度，增加了不少的成就感。感谢老师给我们这次机会让我们自主设计一个电梯调度程序，让我发现了原来嵌入式系统原来是这么的有趣。