**201421130601小组 <模拟电梯程序>概要设计**

版本号：201421130601小组-Design-001

编制时间：2015年5月7日

编制人员：谢非 史柠玮 旦增次旦

1. 用户界面设计

1.1 文件方式（定义输入输出的格式）

**（一）输入：**

输入采用两种方法：

a．从键盘输入呼叫和目标。我们规定：当敲击键1、2、3、4、5、6、7、8、9时表示电梯内有乘客按目标按钮，指定相应目标楼层；当敲击键Q、W、E、R、T、Y、U、I时表示1层到8层有上行呼叫请求；当敲击键A、S、D、F、G、H、J、K时表示2层到9层有下行呼叫请求。

b．从图形界面输入呼叫和目标。当鼠标点击窗口中绘制的呼叫按钮和目标楼层时，表示有新的请求到达。

**（二）输出：**

a．命令行输出：

当电梯运行状态发生改变时，才在命令行窗口打印出电梯当前新的状态、运行方向、当前所在楼层、目标楼层、未响应请求、停靠时间等。

b．动画显示：在窗口中显示各按钮指示灯的亮灭情况、电梯方向指示灯和电梯位置及运行情况。

c．电梯运行情况的记录文件：

记录文件是文本文件，每一行表示一个时刻的电梯运行状态，包括以下内容：

* 当前时间：程序开始运行的系统时钟时间，单位秒。
* 电梯状态：向上运行、向下运行、停靠、空闲。
* 电梯当前楼层：1-9。
* 电梯目标楼层：1-9。
* 停靠时间：记录电梯已经停靠的时间。只有在停靠状态下，才需要输出该信息。
* 未响应的向上呼叫所在楼层：数字（1～8）组成的序列，数字之间用一个空格分割。

如1 5 6 表示1、5、6三层有向上呼叫。

* 未响应的向下呼叫所在楼层：数字（2～9）组成的序列，数字之间用一个空格分割。

如2 5 6 表示2、5、6三层有向下呼叫。

* 未响应的目标：数字（1～9）组成的序列，数字之间用一个空格分割。如1 5 6 表示将1、5、6三层设为目标。

[文件使用/产生说明]：以表格形式打印出电梯状态

1.2 动画方式（画出图形界面）（将在作动画版本时更新和补充）

[图片]

[界面操作说明]：输入操作功能点，输出设计

2 有限状态自动机状态转换图

2.1 [一级状态图]

[状态转换说明]

2.2 [二级状态图]

[二级状态说明]

3 高层数据结构设计

3.1常量定义

#define STATE int8\_t //电梯当前状态的类型

#define VACANT 1 //电梯处于空闲状态

#define UP 2 //电梯处于上行状态

#define DOWN 3 //电梯处于下行状态

#define PAUSE 4 //电梯处于停靠状态

#define MAXFLOOR 9 //电梯最大楼层数

3.2 全局变量定义

BOOL upCmd[8];//上行的指示灯命令

BOOL downCmd[8];//下行的指示灯命令

BOOL innerCmd[9];//内部指示灯命令

struct cmdList head;//头指针

struct cmdList \*ptr;//移动指针

struct cmdList \*outPtr;//遍历指针

STATE state;//当前状态

STATE prestate;//上一个状态

int8\_t curFloor;//当前楼层

int8\_t aimFloor;//目标楼层

int8\_t isPrtVct;

int8\_t flag\_1;

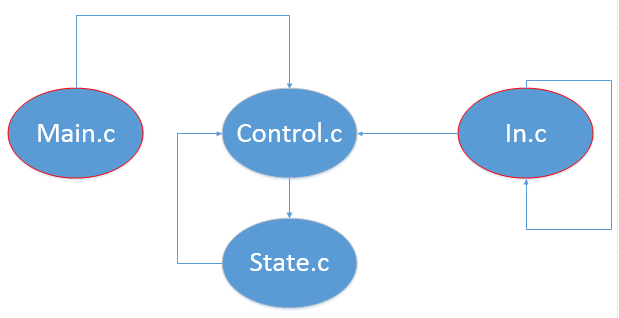
int8\_t flag\_2;

4 系统模块划分

4.1 系统模块划分

模块划分思路说明:

在main.c中循环判断电梯状态和控制运行，in.c分为一个单独线程进行读取数据并传入控制函数中控制运行



**1. 模块名称In.c**

模块功能简要描述：完成对用户输入的线程的实现,根据输入更新目标楼层

**2. 模块名称 State.c**

模块功能简要描述：运行电梯并转换电梯状态

3.模块名称 Control.c

模块功能简要描述：根据用户输入实时确定当前目标楼层

4.模块名称 Main.c

模块功能简要描述：循环判断当前状态并运行控制函数控制电梯运行

4.2各模块接口说明

**1. 模块接口elevator.h**

**该头文件包含了程序运行所需的所有全局变量定义和函数原型**

**#include <windows.h>**

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <stdint.h>**

**#define STATE int8\_t //电梯当前状态的类型**

**#define VACANT 1 //电梯处于空闲状态**

**#define UP 2 //电梯处于上行状态**

**#define DOWN 3 //电梯处于下行状态**

**#define PAUSE 4 //电梯处于停靠状态**

**#define MAXFLOOR 9 //电梯最大楼层数**

**struct cmdList{**

**int floor;**

**struct cmdList \*next;**

**};**

**/\*顺便服务策略的指示灯\*/**

**BOOL upCmd[8];//上行的指示灯命令**

**BOOL downCmd[8];//下行的指示灯命令**

**BOOL innerCmd[9];//内部指示灯命令**

**struct cmdList head;//头指针**

**struct cmdList \*ptr;//移动指针**

**struct cmdList \*outPtr;//遍历指针**

**STATE state;//当前状态**

**STATE prestate;//上一个状态**

**int8\_t curFloor;//当前楼层**

**int8\_t aimFloor;//目标楼层**

**int8\_t isPrtVct;**

**int8\_t flag\_1;**

**int8\_t flag\_2;**

**DWORD WINAPI input\_1(LPVOID parameter);//策略一输入进程**

**DWORD WINAPI input\_2(LPVOID parameter);//策略二输入进程**

**//状态转换函数**

**void state\_trans(void);**

**//输出函数**

**void print\_message(void);**

**//判断目标楼层的控制函数**

**void control\_1(void); //第一策略**

**void control\_2(void);//第二策略**

**void control\_3(void);//第三策略**

4.3各模块函数说明

|  |  |  |  |
| --- | --- | --- | --- |
| 模块文件 | 模块说明 | 模块包含的函数名 | 函数功能概要描述 |
| State.c | 控制电梯的状态的转换 | **void state\_trans(void)** | 判断当前状态并运行 |
| Control.c | 确定电梯的目标楼层 | **void control\_1(void)** | **第一策略** |
| **void control\_2(void)** | **第二策略** |
| **void control\_3(void)** | **第三策略** |

4.4 函数说明

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 函数原型 | 功能详细描述 | 参数说明 | 返回值说明 |
| 1 |  | 何时何条件被何模块调用？  完成什么功能？ |  |  |
| 2 |  |  |  |  |

5 核心算法设计

电梯系统的控制策略算法：

策略一:

输入时将数据存入链表中，main进程由链表头向后读取，即可实现先来先服务

策略，当读取到某个节点时先控制电梯运行到该节点命令所指向的目标楼层，

如果读取到链表尾部时仍无新的输入，则开始Sleep直到链表末尾不为空

策略二：将用户指令存储到

。要求对任务书中的策略进一步细化分解，结合上面的数据结构和模块设计，描述出算法的实现思路。