**C语言课程设计报告**

**16040310015 李伟健**

**状态函数流程图：**

**if ((GetCallLight || GetPanelFloorLight) == true)**

**SetMotorPower(0);**

**SetDoor();**

DoorOpen

MovingUp

**③**

**if ((GetCallLight || GetPanelFloorLight) == true)**

**SetMotorPower(1);**

MovingDown

Idle

**②**

**①**

**if(IsDoorClosed(GetNearestFloor()))**

**if(GetCallLight())/if(GetOpenDoorLight())**

**SetDoor();**

DoorClosing

:

**④**

**if ((GetCallLight || GetPanelFloorLight) == true)**

**SetMotorPower(-1);**

**if((GetCallLight||GetPanelFloorLight)==true)**

**SetMotorPower(0);**

**SetDoor();**

**③**

1. **：if (GetCloseDoorLight() == true)**

**SetDoor(); / if (IsDoorOpen(GetNearestFloor()))**

1. **：if (GetOpenDoorLight() == true)**

**SetDoor(); / if (IsBeamBroken() == true)**

1. **：if ((GetOpenDoorLight || GetCloseDoorLight) == true)**

**SetOpenDoorLight(false);**

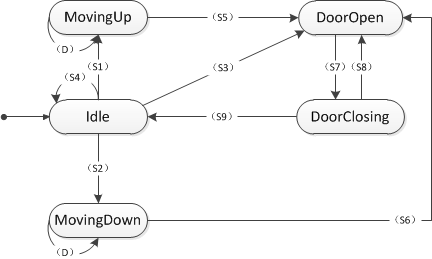
**SetCloseDoorLight(false);**

1. **：if (GetCloseDoorLight()==true)**

**SetCloseDoorLight();**

**对状态机图分解描述：**

**（字母对应过程见图）**



**S1：电梯在静止状态下检测到门外呼叫按钮或门内楼层按钮，从而开始向上移动：**

**if ((GetCallLight || GetPanelFloorLight) == true)**

**SetMotorPower(1);**

MovingUp

Idle

**S2：电梯在静止状态下检测到门外呼叫按钮或门内楼层按钮，从而开始向上移动：**

**if ((GetCallLight || GetPanelFloorLight) == true)**

**SetMotorPower(-1);**

Idle

MovingDown

**S3：电梯在静止状态下检测到门外呼叫按钮，或检测到门内开门按钮，从而开门；**

**if(GetCallLight())/if(GetOpenDoorLight())**

**SetDoor();**

DoorOpen

Idle

**S4：电梯在静止状态下检测到门内关门按钮，指令无效，仍是静止状态；**

Idle

**if(GetCloseDoorLight()==true)**

**SetCloseDoorLight();**

**S5：电梯在上行过程中检测到门外呼叫按钮或门内楼层按钮，在到达相应楼层后，电梯门开启；**

**if ((GetCallLight || GetPanelFloorLight) == true)**

**SetMotorPower(0);**

**SetDoor();**

DoorOpen

MovingUp

**S6：电梯在上行过程中检测到门外呼叫按钮或门内楼层按钮，在到达相应楼层后，电梯门开启；**

**if((GetCallLight||GetPanelFloorLight)==true)**

**SetMotorPower(0);**

**SetDoor();**

DoorOpen

MovingDown

**S7：电梯门若开启，电梯检测到门内关门指令，或开启后两秒，电梯门关闭；**

**if (GetCloseDoorLight() == true)**

**SetDoor()；**

**/if(IsDoorOpen(GetNearestFloor()))**

DoorClosing

DoorOpen

**S8：电梯门若关闭，电梯检测到门内开门指令，或在关门过程中红外检测到障碍物，电梯门开启；**

**if (GetOpenDoorLight() == true)**

**SetDoor();**

**/if (IsBeamBroken() == true)**

DoorClosing

DoorOpen

**S9：电梯门关闭后，进入静止状态；**

**if(IsDoorClosed(GetNearestFloor()))**

Idle

DoorClosing

**D：电梯上行和下行的过程中，开门与关门指令无效。**

MovingDown

MovingUp

**if((GetOpenDoorLight||GetCloseDoorLight) == true)**

**SetOpenDoorLight(false);**

**SetCloseDoorLight(false);**

**可实现功能：**

**（参考PPT中所给出的指标，下述功能均可实现）**

**反思：刚开始在运行5功能的时候，电梯到了二楼后第一次开门没有消费门外呼叫按钮，进行了两次开门/关门，经过检查，发现在**if (fabs(GetFloor() - floor) < Lib\_FloorTolerance)//判断是否到达目标楼层

{

SetMotorPower(0);//Transition

SetDoor(floor, true);//Transition

SetPanelFloorLight(floor, false);

\*state = DoorOpen;

if (floor != 3)//判断楼层是否为最高层

SetCallLight(floor, true, false);

else

SetCallLight(floor, false, false);

printf("Transition: from MovingUp to DoorOpen .\n");

**上述语句在判断不为最高层时，只消费向上的呼叫按钮，不消费向下的呼叫按钮，同理在判断不为最低层时也有同样错误，经过排查，加上消费门外向下（movingup状态中）/向上（movingdown状态中）按钮后，程序可正常运行。**

1. **电梯停于1F或2F时，按3F向下呼叫按钮；电梯上升到3F停止，开门/关门。**
2. **电梯停于2F或3F时，按1F向上呼叫按钮；电梯下降到1F停止，开门/关门。**
3. **电梯停于1F时，按2F向上呼叫按钮；电梯上升到2F停止，开门/关门。**
4. **电梯停于3F时，按2F向下呼叫按钮；电梯下降到2F停止，开门/关门。**
5. **电梯停于1F，2F和3F均有按钮呼叫；电梯先上升到2F，开门/关门，然后上升到3F停止，开门/关门。**
6. **电梯停于3F，2F和1F均有按钮呼叫；电梯先下降到2F，开门/关门，然后下降到1F停止，开门/关门。**
7. **电梯上升途中或下降途中，任何反方向按钮呼叫均无效。**
8. **电梯停于1F，按3F向下呼叫按钮，然后立即按2F向下呼叫按钮；电梯上升到3F停止，开门/关门，然后下降到2F停止，开门/关门。**
9. **电梯停于2F，按门内楼层按钮3，然后门内楼层按钮1；电梯上手到3F停止，开门/关门，然后下降到1F，开门/关门。**
10. **电梯停于1F，按门内楼层按钮3，当电梯上升在1F到2F中间以下，按2F向上呼叫按钮；电梯先上升到2F，开门/关门；然后再到3F，开门/关门。**
11. **电梯停于1F，按门内楼层按钮3，当电梯上升在1F到2F中间以上，按2F向上呼叫按钮；电梯先上升到3F，开门/关门；然后再到2F，开门/关门。**
12. **电梯停于1F，按2F向下呼叫按钮和向上呼叫按钮以及3F的向下呼叫按钮；电梯上升到2F停止，开门/关门，2F的向上呼叫按钮灯关闭，开门/关门， 2F的向下呼叫按钮灯关闭，然后上升到3F停止，开门/关门，3F的向下呼叫按钮关闭。**
13. **电梯在2F以上，10s无动作，自动降到1楼。**
14. **所有停止，开门/关门后，对应楼层的同方向门外呼叫按钮灯（最高楼向下呼叫按钮，最底层向上呼叫按钮）和门内楼层按钮灯关闭。**
15. **运动状态，开关门按钮失效。正在开门，开门按钮失效；正在关门，关门按钮失效。**
16. **空闲状态，门是关闭的，因此按关门按钮失效。**
17. **开关门结束，延时2秒用于乘客上下电梯【延时功能在库函数中实现，不用在状态函数中实现】，然后进入关门状态。**
18. **开门结束前，按关门按钮，转而进入关门状态；关门结束前，按开门按钮，转而进入开门状态。**

#include "stdafx.h"

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#include "ElevatorLib.h"

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Idle状态，电梯停止在某楼层，门是关闭的，处于静止状态，等待相关事件的发生，从而转换到下一个状态。

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void StateIdle(int \*state)

{

bool up;

int floor;

floor = IdleWhatFloorToGoTo(&up);

if(GetOpenDoorLight()==true)//Event

{

SetDoor(GetNearestFloor(), true);//Transition

SetOpenDoorLight(false);

\*state = DoorOpen;

printf("Transition: from Idle to DoorOpen.\n");

}

else if (GetCallLight(GetNearestFloor(), true) || GetCallLight(GetNearestFloor(), false))//Event，门外向下或向上呼叫按钮被按下

{

SetDoor(GetNearestFloor(), true);//Transition

SetCallLight(GetNearestFloor(), true, false);

SetCallLight(GetNearestFloor(), false, false);

\*state = DoorOpen;

printf("Transition: from Idle to DoorOpen.\n");

}

else if ((GetCallLight || GetPanelFloorLight) == true)//Event，门外呼叫按钮或门内楼层按钮被按下

{

if (floor > 0 && up)//判断楼层

{

SetMotorPower(1);//Transition

\*state = MovingUp;

printf("Transition: from Idle to MovingUp.\n");

}

else if (floor > 0 && !up)//判断楼层

{

SetMotorPower(-1);//Transition

\*state = MovingDown;

printf("Transition: from Idle to MovingDown.\n");

}

}

else if (GetCloseDoorLight()==true)

{

SetCloseDoorLight(false); return;

}

}

/\*MovingUp状态函数\*/

void StateMovingUp(int \*state)

{

bool up;

int floor;

floor = GoingUpToFloor();

if ((GetCallLight || GetPanelFloorLight) == true)//Event，门外呼叫按钮或门内楼层按钮被按下

{

if (fabs(GetFloor() - floor) < Lib\_FloorTolerance)//判断是否到达目标楼层

{

SetMotorPower(0);//Transition

SetDoor(floor, true);//Transition

SetPanelFloorLight(floor, false);

\*state = DoorOpen;

if (floor != 3)//判断楼层是否为最高层

{

SetCallLight(floor, true, false);

SetCallLight(floor, false, false);

}

else

SetCallLight(floor, false, false);

printf("Transition: from MovingUp to DoorOpen .\n");

}

}

else if ((GetOpenDoorLight || GetCloseDoorLight) == true)//Event，门内开门按钮火关门按钮被按下

{

SetOpenDoorLight(false);

SetCloseDoorLight(false);

}

}

/\*MovingDown状态函数\*/

void StateMovingDown(int \*state)

{

bool up;

int floor;

floor = GoingDownToFloor();

if((GetCallLight||GetPanelFloorLight)==true)//Event

{

if (fabs(GetFloor() - floor) < Lib\_FloorTolerance)//判断是否到达目标楼层

{

SetMotorPower(0);//Transition

SetDoor(floor, true);//Transition

SetPanelFloorLight(floor, false);

\*state = DoorOpen;

if (floor != 1)//判断楼层是否为1层

{

SetCallLight(floor, false, false);

SetCallLight(floor, true, false);

}

else

SetCallLight(floor, true, false);

printf("Transition: from MovingDown to DoorOpen .\n");

}

}

else if ((GetOpenDoorLight || GetCloseDoorLight) == true)//Event，门内开门按钮火关门按钮被按下

{

SetOpenDoorLight(false);

SetCloseDoorLight(false);

}

}

/\*DoorOpen状态函数\*/

void StateDoorOpen(int \*state)

{

bool up;

if (GetCloseDoorLight() == true)//Event

{

SetDoor(GetNearestFloor(), false);//Transition

SetCloseDoorLight(false);

\*state = DoorClosing;

printf("Transition: from DoorOpen to DoorClosing .\n");

}

else if (IsDoorOpen(GetNearestFloor()))//Event，门打开

{

SetDoor(GetNearestFloor(), false);//Transition

\*state = DoorClosing;

printf("Transition: from DoorOpen to DoorClosing .\n");

}

else if (GetOpenDoorLight() == true)

{

SetOpenDoorLight(false);

}

}

/\*DoorClosing状态函数\*/

void StateDoorClosing(int \*state)

{

bool up;

if (GetOpenDoorLight() == true)//Event

{

SetDoor(GetNearestFloor(), true);//Transition

SetOpenDoorLight(false);

\*state = DoorOpen;

printf("Transition: from DoorClosing to DoorOpen .\n");

}

else if (GetCloseDoorLight() == true)

{

SetCloseDoorLight(false);

}

else if (IsBeamBroken() == true)//Event，红外探测到物体

{

SetDoor(GetNearestFloor(), true);//Transition

\*state = DoorOpen;

printf("Transition: from DoorClosing to DoorOpen .\n");

}

else if (IsDoorClosed(GetNearestFloor()))//门关闭

\*state = Idle;

printf("Transition: from DoorClosing to Idle .\n");

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 状态机，每隔一定时间(如，100ms)被调用一次，采集系统的运行状态

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void main\_control(int \*state)

{

if(IsElevatorRunning()) // 仿真正在运行

{

switch(\*state)

{

case Idle:

// Idle状态，一定时间无动作，自动到一楼

if(GetNearestFloor() !=1 ) {

AutoTo1Floor();

}

StateIdle(state);

break;

case MovingUp:

CancelTo1Floor(); // 其它状态，取消自动到一楼

StateMovingUp(state);

break;

case MovingDown:

CancelTo1Floor();

StateMovingDown(state);

break;

case DoorOpen:

CancelTo1Floor();

StateDoorOpen(state);

break;

case DoorClosing:

CancelTo1Floor();

StateDoorClosing(state);

break;

default:

printf("没有这种状态!!!\n");

}

}

}