中国矿业大学

计算机科学与技术学院

2017 级本科生课程报告

课程名称 硬件课程设计

设计题目 出租车计价器(C 语言)

开课学期 2020 学年第一学期

报告时间 2019 年 12 月 20 日

学生姓名 陆玺文

学　　号 03170908

班　　级 计科 2017-06 班

专　　业 计算机科学与技术

任课教师 马海波

《硬件课程设计》课程报告评分表

开课学期： 2020 学年第一学期

姓名： 陆玺文　　学号：03170908　　专业班级： 计科 2017-06

序号	毕业要求	课程教学目标	考查方式与考查点	占比	得分
1	1.3	**目标 1：**了解微机应用系统解决复杂工程问题的基本方法。掌握微机应用系统硬件电路设计及软件功能需求分析方法和模型。能够针对微机系统应用领域工程需求的系统要求，进行分析与设计。	中期检查与设计文档 掌握解决复杂工程问题的基本方法。微机应用系统软硬件设计相关的理论知识。	10%	
2	4.3	**目标 2：**能够针对硬件电路组成需求描述进行系统硬件设计，能够分析系统功能的软件需求，根据模块设计原则，综合考虑系统的算法模型和软硬件开发，进行合理的方案设计、编程实现、系统测试及对设计方案进行优化。	中期检查与设计文档 考核题目需求分析和功能分析；综合知识应用能力及设计方案的完整性； 考核软件编程及系统调试测试，设计方案进行优化。	30%	
3	9.1	**目标 3：**具备多学科背景知识，并制定项目计划，能够按照标准规范进行设计。能够在多学科背景下具备独立分析问题解决问题的能力。	中期检查与设计文档 考核独立分析问题解决问题的能力	10%	
4	10.3	**目标 4：**掌握设计报告撰写，通过成果演示、陈述发言的清晰表达、回答问题准确性等。	现场验收与答辩 考核编程实现的代码难度和复杂性、设计工作量等；考核设计成果、所涉及的问题答辩。验收设计报告的结构合理性、内容和图表的正确性。验收设计报告排版的规范性。	40%	
5	12.1	**目标 5：**对选题主动通过各种途径寻求解决方法（主动查阅资料、请教老师、同学讨论等）。通过各种资源平台的使用及教师意见的反馈，完成高质量的设计任务，有无创新意识。	现场验收与答辩 考核设计成果完整性；所涉及的设计课题的创新性。	10%	
总成绩				100%	

任课教师：马海波

2019 年 12 月 20 日

目录

1.绪论

1.1 选题说明

如今，随着城市经济的发展，市民对于快速出行有了更高的需求，城市内的出租车数量也大大提高，计价器作为里程计量与金额计算仪器得到了广泛的运用。设计与实现一个计量精准，功能完善，界面美观的计价器具有极高的实用价值经济价值。

此外，计价器主功能为里程计算与界面显示，对于硬件的要求并不苛刻，在所选用的实验平台上可以完备地模拟出完整的计价器功能，能够更好地将所设计实现的模拟计价器与实物进行比对。

1.2 设计任务及要求

利用实验仪器与平台，结合 C 语言编程，模拟出租车计价器的实现。利用 0832 控制直流电机运转，通过 8253 采集霍尔传感器的转速信号，并在 LCD 屏显示公里数及价格，由 8255 模拟车辆的启、停，白天、夜间计价及清零功能。

在此基本功能基础上，加以创新实现新功能。

2.系统设计需求分析

2.1 实验系统及软件开发平台

实验系统：TPC-ZK-II 综合开放式微机原理及接口技术实验系统

软件平台：VC6.0

2.2 系统功能需求分析

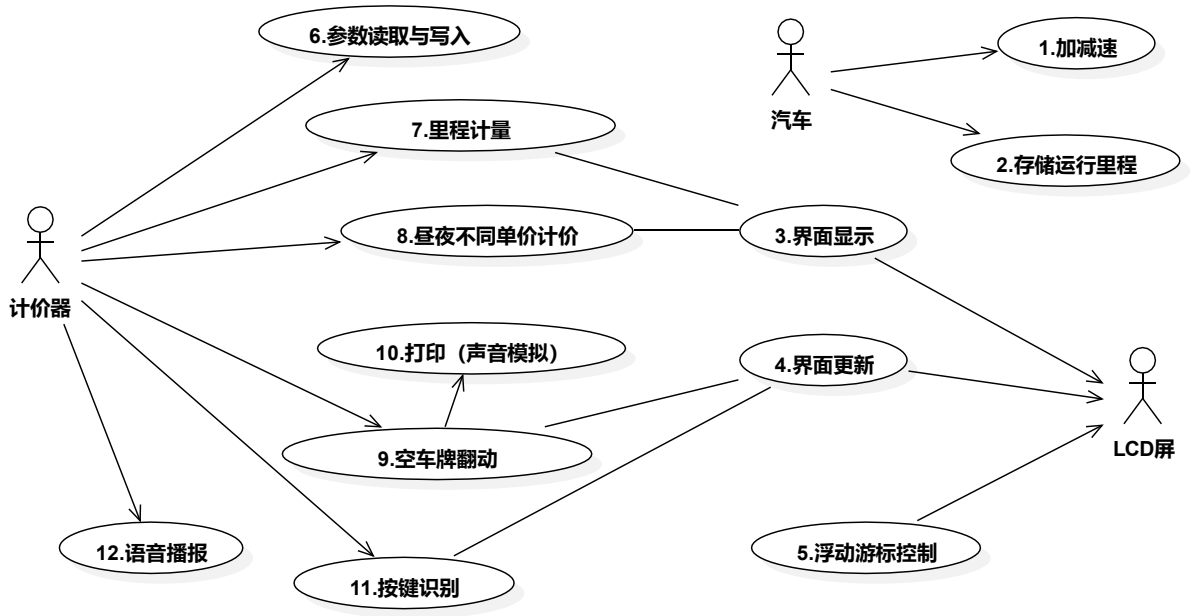


图 1　系统需求用例图

图 1 展示了系统的**功能需求**。做为演示系统，应当能够模拟汽车的加减速，进而实现计价显示的数值观察。车辆也自然应当存储行驶里程记录，便于计价器的读取。计价器应当能够实现空车牌翻动、里程计量、价格计算、昼夜单价识别切换、打印凭条、识别按键、语音播报欢迎等功能。

此外，在计价器的工作运行过程中，应当满足界面显示美观，按键响应灵敏，界面切换更新顺畅，计价准确等**性能要求**。

2.3 系统主要算法及分析

考虑到在计价器的运行过程中，按键操作是主要的交互操作，因而围绕按键识别与程序跳转设计系统主要的运行算法。

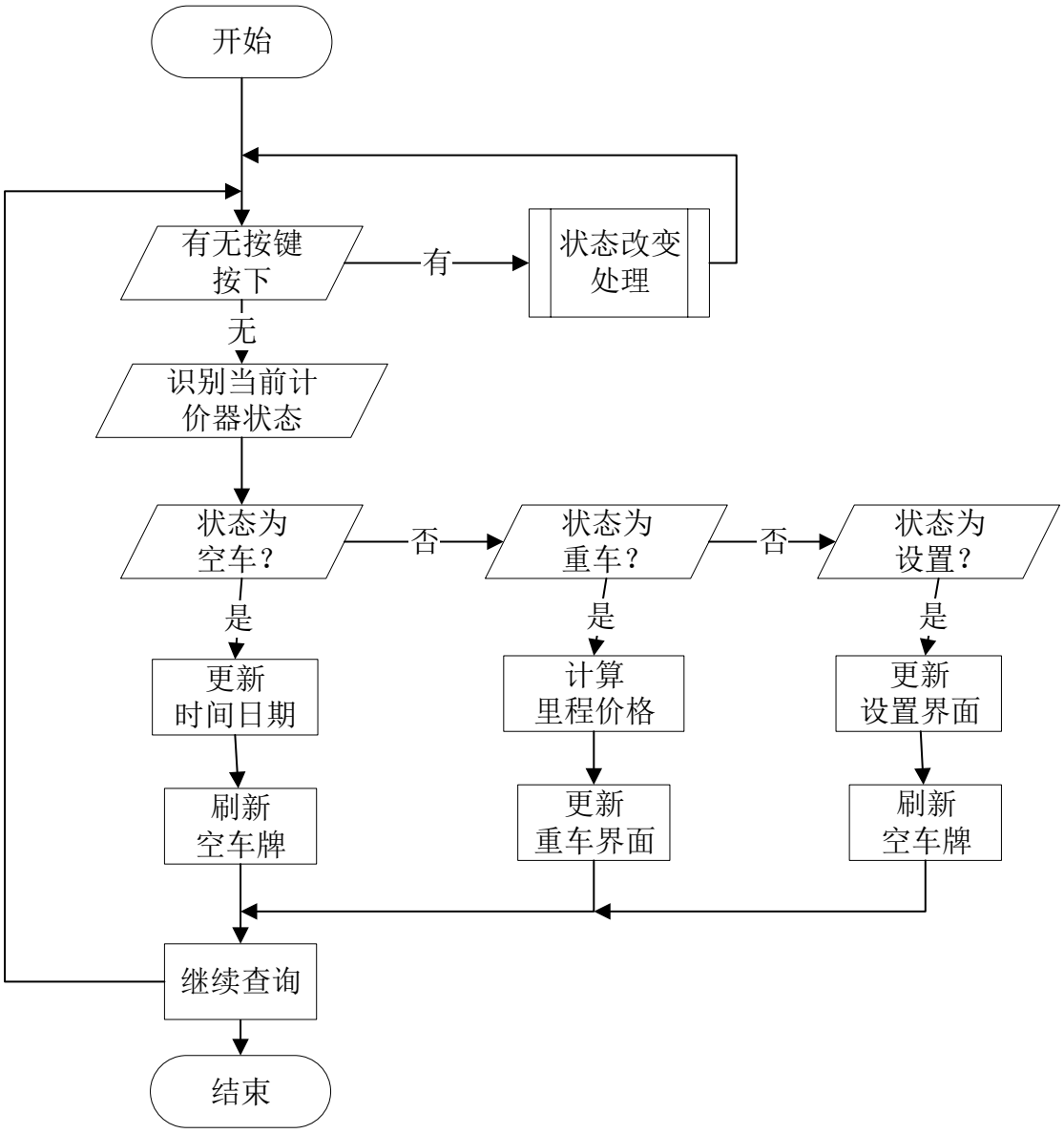


图 2　系统主算法流程图

如图 2 所示，在系统中给计价器定义若干状态，程序中循环**查询键盘**，若有按键按下，则进入状态改变程序段，进行相应处理；若无按键按下，则根据目前的状态执行相

应的界面更新。

考虑到按键动作与中断请求相似，引入**中断类接口**[1]，使系统支持运用中断识别按键。

2.4 系统的组成及工作原理

从图 1 中系统需求中可以看出，需要使用的硬件有 8255、8254、DAC0832、键盘、74LS273 简单输出、LCD12864 液晶屏、8×8 点阵屏、AD0809、喇叭。

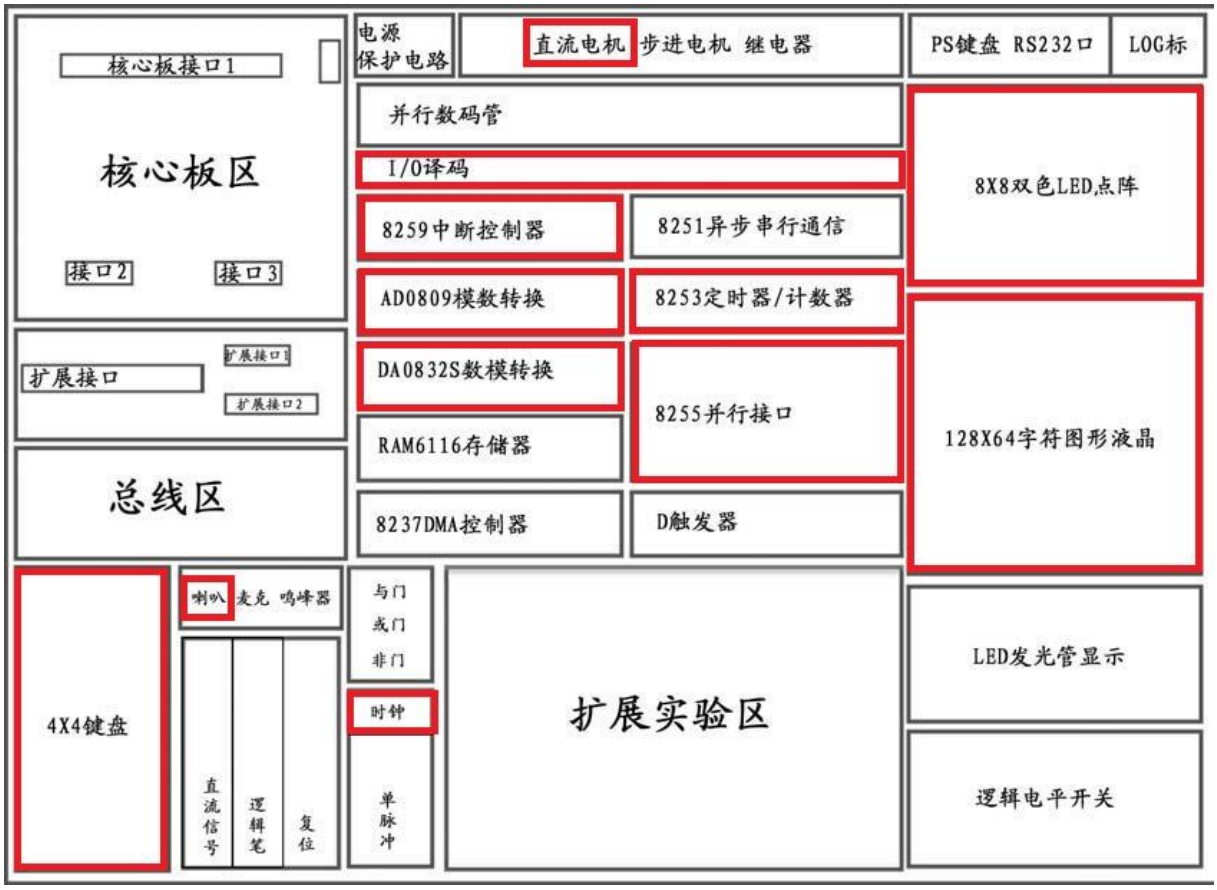


图 3　TPC-ZK 实验系统使用部分示意图

2.4.1　8255 工作任务

8255 是系统的主控芯片，选用方式字 0x81，使 A 口 B 口输出，C 口高 4 位输出第四位输入。当系统采用查询键盘的方式进行工作时，C 口高 4 位负责行线的拉低，第四位读取列线，进而识别按下的按键。B 口的低 3 位负责控制 LCD12864 液晶的控制引脚，B 口最高位 B7 通过控制 8253 的 Gate 门控信号，使系统发出“嗒嗒嗒”声模拟打印操作。

2.4.2　8254 工作任务

8254 在系统中，一是负责接收电机所产生的霍尔传感器信号，进行里程计数；二是输出特定频率波，模拟打印声“嗒嗒嗒”；三是可以当做定时器，进行界面的定时刷新。

[1] 程序中实现了 8259 中断，并提供了中断接口，但整体演示系统仍使用 8255C 端口查询键盘的方式。

2.4.3 DAC0832 工作任务

DAC0832 在系统中，一是负责模拟汽车的档位控制，调节电机转速；二是在录放音过程中，输出不同频率的信号。

2.4.4 8×8 双色点阵

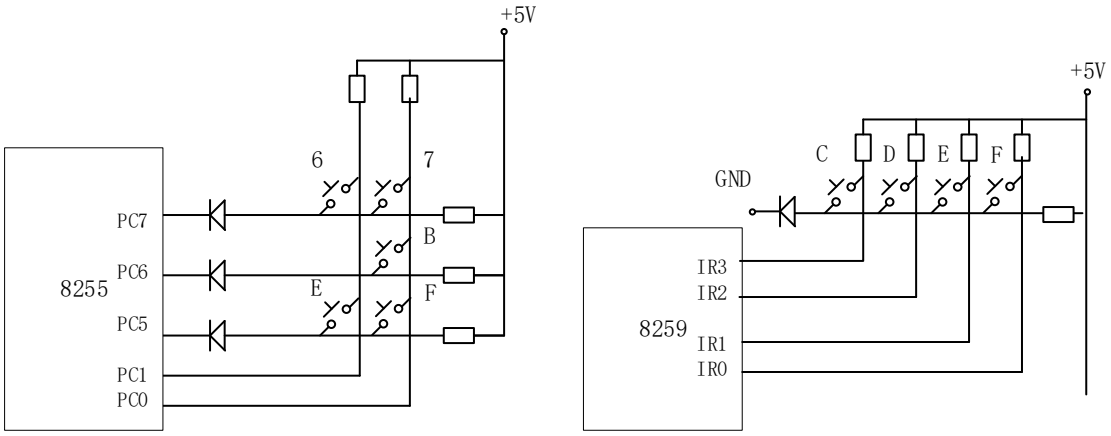
点阵屏在系统中，模拟空车牌，通过亮（灭）对应相应的空车牌抬起（翻下）。

2.4.5 LCD12864 液晶屏

液晶屏在系统中，主要负责界面的显示与更新，同时增添游标显示（关闭）使得在进一步拓展设置参数时可以有所提示。

2.4.6 8259 工作任务

TPC-ZK-II 实验平台在 USB 核心板上包含主、从两片 8259，在计价器系统中，考虑到端口数量及 C 语言不易注册中断向量问题，选用扩展 8259 芯片，结合查询中断查询字的方法，将 8255 查询键盘方式进行升级改进。同时使用一行按键，使得系统可以通过列线跳变自动得知按键键值。



(a)查询式下选用按键示意图　　(b)中断方式下选用键盘示意图

图 4　查询与中断方式下键盘示意图

2.4.7 AD0809 工作任务

AD0809 在系统中，主要负责在录放音模块中采集语音，并转换为数字量进行存储，供以后进行调用播放。

3.系统概要设计

采用面向对象的软件设计方法，使用 MVC 架构进行程序编写，每一芯片为一个模型（Model），实现对业务逻辑的处理；液晶屏与键盘代表视图（View），分别用于显示数据和接受用户请求与输入；出租车和计价器类为控制器（Controller），连接模型和视

图，调用不同的模型处理用户请求，选择不同视图显示系统的信息。

3.1 系统层次图

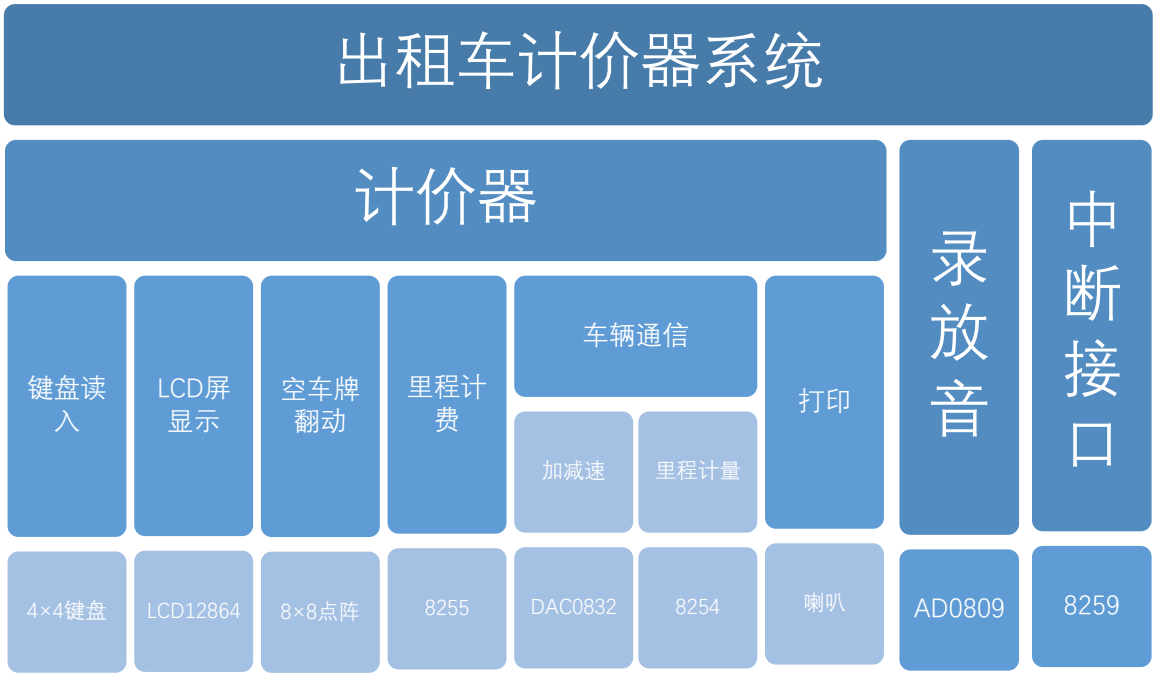


图 5　计价器系统层次图

3.2 静态建模

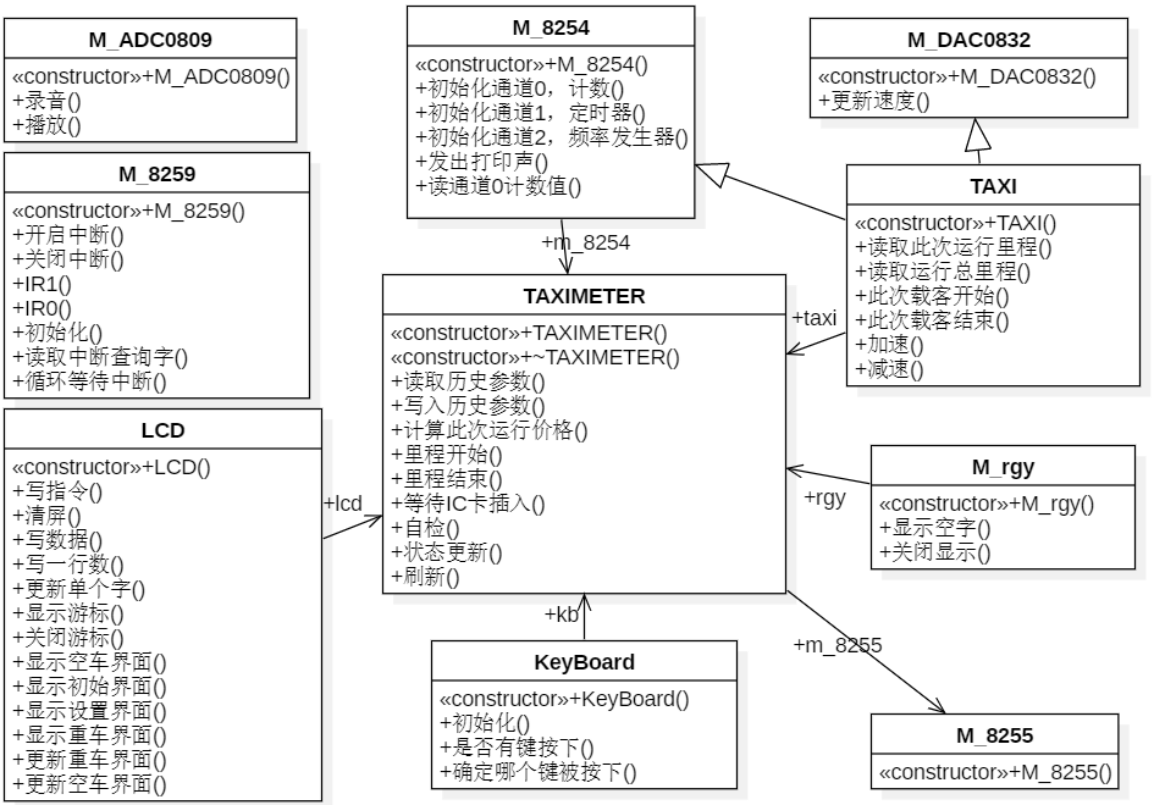


图 6　计价器系统类图（隐藏属性值）

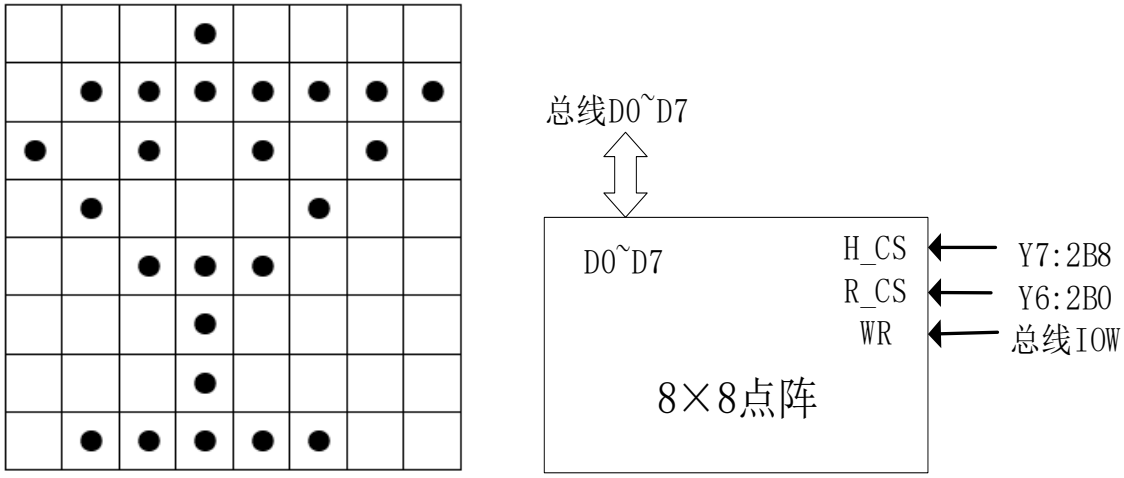
图 6 中，设计了程序的整体框架，主要以各个芯片为一类，另外设计 TAXI（车辆）类，封装加速减速操作，以及负责与 8254 交互读取当前里程的里程数。TAXIMETER（计价器）类，为程序主控制类，负责程序整体运行流程的控制，同时接收键盘键值，执行相应的状态转换，并负责界面的更新。

4.系统详细设计

在架构设计完成的基础上，针对整个系统的各个模块，进行详细的设计并实现，最终整合各模块，完成系统的整体功能。

4.1 空车牌翻动

空车牌的翻上（翻下），在现实计价器功能中代表了单次运营服务计价的开始（结束）。在微机实验箱上，选用 8×8 点阵屏显示“空”字，模拟空车牌的亮起。同时在查询方式下设定按键“B”的按下事件代表空车牌的翻动，进行点阵屏相应的亮灭操作。



(a)点阵屏空字示意图　　(b)点阵屏接线示意图

图 7　点阵屏显示与接线示意图

在具体的编程实现过程中，由图 7 可以看出，首先针对点阵屏的特点，对所要进行显示的“空”字编码，然后逐行输出特定码值，动态刷新显示各行，利用视觉暂留实现“空”字的显示。整个点阵屏工作在总线模式下，数据区与总线进行交互。同时，为了提高显示效果，在每刷新一行后，延时一小段时间（10ms），从而点阵屏更加稳定。

4.2 汽车加减速

在系统中，使用直流电机的转动模拟汽车的运行，转速即为运行时速。DAC0832 控制电机的转速，接受一个 0~255 内的数值，输出相应模拟量控制电机。在此基础上 TAXI 类继承 DAC0832，守护速度属性值，同时提供加减速操作。

DAC0832 片选端口外接 Y0:280，0～+5v 向直流电机输出。

TAXI 类代表汽车动作与状态，并维护未启动(state:0)，运行(state:1)，静止等待(state:10)，三个状态。

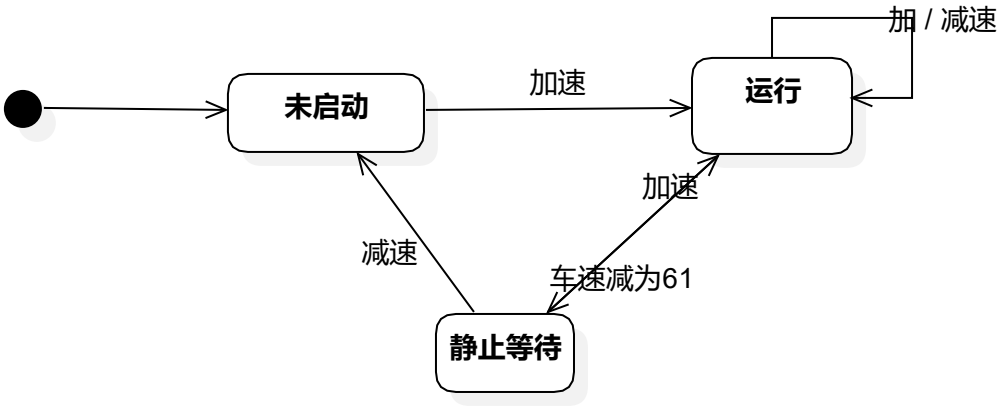


图 8　车辆（TAXI 类）状态图

如图 8 所示，当汽车速度达到 254 时，不再响应加速动作；当速度不大于 70 时，再接受减速操作时，速度固定为 61，同时转为静止等待状态；速度为 61 时，再次响应减速操作则认为汽车已经熄火，进入未启动状态。

4.3 里程计量

在系统中，TAXI 类负责维护自身的里程计量，包含单次里程与总运营里程[2]。具体里程计量由 8254 芯片的计数功能实现。编写 M_8254 类封装 8254 的功能，其中通道 0 工作在方式 0 下，采用倒计数的方式，计数初值为 16 位二进制的最大数 65536。同时提供读数方法，及时锁存端口计数值，读出数值取补并返回 TAXI，得到的即为相应时间段内通过的脉冲数量。当读出计数值大于设定阈值（60000）时，命令 8254 自行重装计数初值。

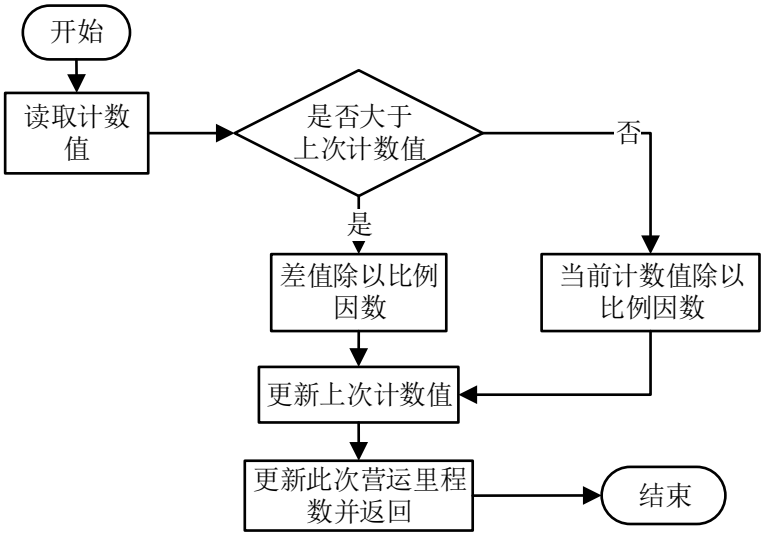


图 9　TAXI 类里程计量流程图

TAXI 类向外提供里程数读取方法，内部调用 8254 读数方法得到一个计数返回值，这里维护一个上次计数值（temp），将两者差值除以比例因子（factor）即得到里程数。

[2] 系统中实现的总运营里程仅包含载客里程总数，不包含空车状态下的行驶里程。

如图 9 所示，考虑到 8254 计数时自行重装计数初值的问题，每次读数所得值与 temp 进行比较，若小于 temp 则说明 8254 在时间间隔内完成了一次计数初值的重装。此时的里程计算不再使用差值，而直接使用所读到的计数初值进行计算[3]。

4.4 价格计算

计价器主类 TAXIMETER 从车辆类 TAXI 中，读取到当前里程数后，负责对此次服务价格的计算。为了实现昼夜不同单价计量，在每次服务开始时，系统会根据当前时间确定所要使用的单价，并维护当前运行单价（Per_Price_Now）这一属性值。计算价格时，采用分段计价法，并将结果处理成精度为小数点后一位的值返回。

$$M=\begin{cases} P_s, m\le 3 \\ P_s+(m-3)\times P_N, m>3 \end{cases}$$

上式即为计价器中价格计算函数，其中 M 为总价格 Money，P_S 为起步价，P_N 为单次运行单价，由属性 Per_Price_Now 定义。m 即为从 TAXI 类中所读取得到的里程数。

计价器的参数，在对象生成时自动从历史数据中读取，程序系统目前采用直接赋值的方式进行实现，此处可以进一步拓展从文件或存储器中读取历史参数。

4.5 状态切换

计价器工作在不同情况下时，接受操作所执行响应也应有所区别。借鉴自动机的思想，定义计价器的如下几种工作状态，并设定状态之间的转移规则。

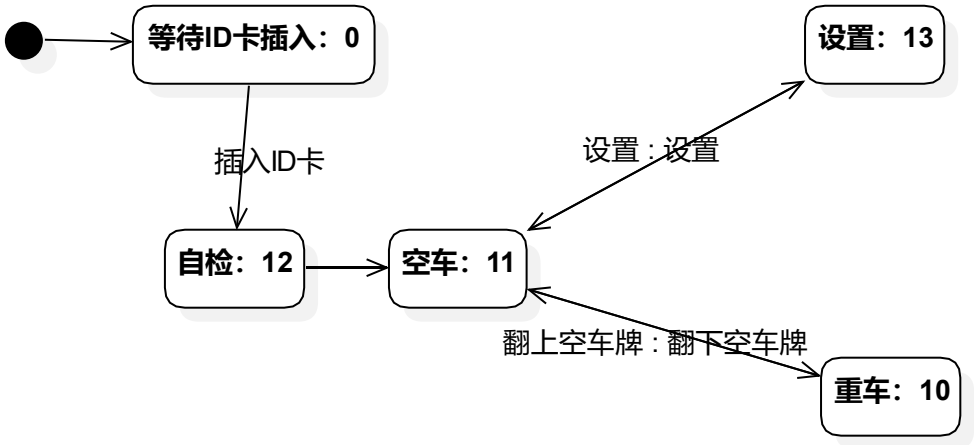


图 10　计价器状态图

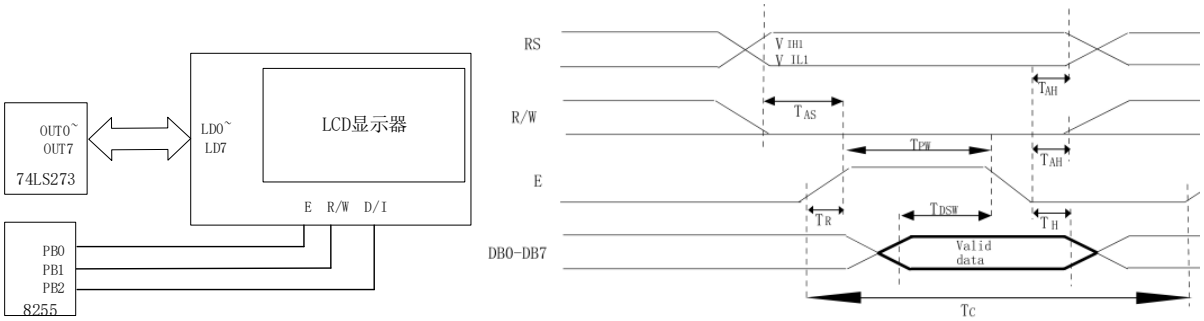
定义计价器的工作状态有，等待 ID 卡（state:0），自检（state:12），空车（state:11），设置（state:13），重车（state:10）。所有功能按键中，加速、减速按键不受影响，在任何时候都能得到响应，并执行相应的动作。其余功能键（插卡、设置、启停）在计价器的不同状态有不同的功能受限。如图 10 所示，等待插卡时，只接受插卡动作；空车时，所有按键正常响应，并转入相应状态；设置状态时，只响应设置操作回到空车界面；重

[3] 这种方式进行的循环装数再计数在理论上会有一定的计数误差。车速越快，读数间隔越长，误差越大。

车状态时，只接受停止操作，单次服务停止并回到空车界面。针对重车状态的启停，单独在计价器类中再设旅程开始与旅程结束函数，将重车状态切换时各项功能组合在一起。

4.6 界面显示与刷新

液晶屏是系统的主要显示途径，界面设计以美观稳定为主要考虑因素。



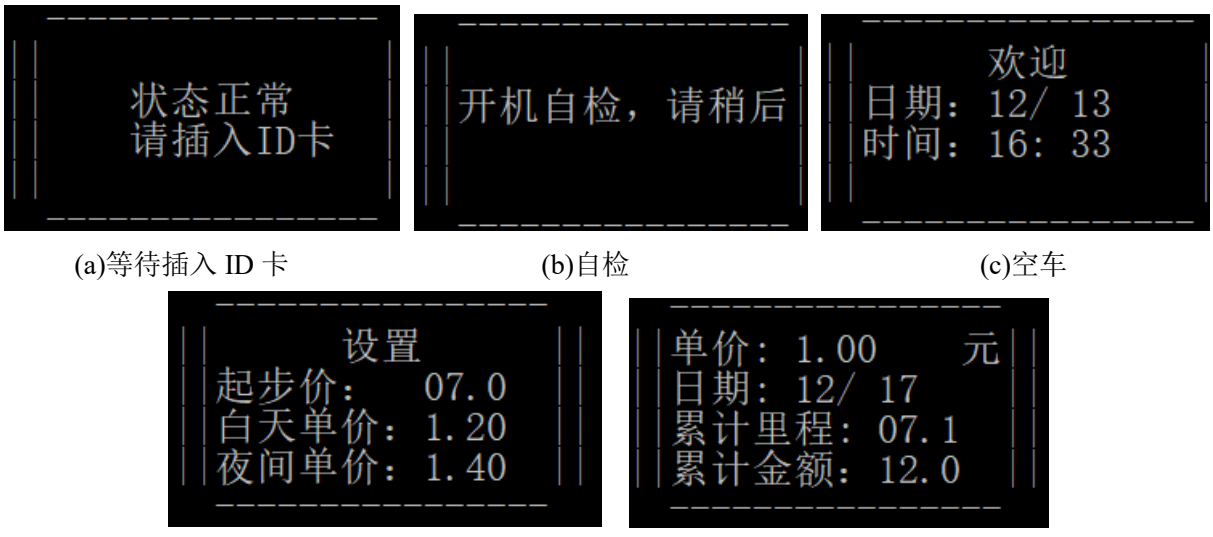
(a)显示屏接线示意图　　(b)LCD 写资料到模块时序图

图 11　显示屏接线与写资料时序示意图

图 11 中展示了液晶屏的连线示意图，以及其向写资料到模块时的时序图[6]，可以看到在 E 端口的一个高电平区间，指令数据得以写入。因而稳定时间段，E 端口保持低电平，在需要写入指令或数据时，给高电平，再回到稳定的低电平。

界面显示

为了与计价器的 5 种状态相匹配，液晶屏的主显示界面也设计了对应的 5 种界面，如图 12 所示。其中，图时钟信息的显示，由 C 语言获取系统时钟后传入参数，并将数字转为对应的显示字符编码，进行实时时钟日期的显示。



(a)等待插入 ID 卡　(b)自检　(c)空车

(d)设置　(e)重车

图 12　液晶屏界面显示设计

界面刷新

计价器的界面中，有不少信息是动态变化的，如时钟信息，里程信息，需要不断对

其进行刷新。考虑到显示效果的稳定性，故选择针对界面进行最小更新，而不是整屏更新。当系统工作在查询方式下时，每次查询的最后，若当次无操作发生，则根据此时的计价器状态，进行界面的特定更新。

在里程更新时，传入参数为保留至小数点后一位的里程数和服务费，同样需将数字转为对应的显示字符编码，在特定位置予以输出。

4.8 打印模块（喇叭发声模块）

单次运行结束时，即空车牌翻下时，发出“嗒嗒嗒”声模拟系统自动执行打印操作。其中，8254 通道 2 工作在方式 3 发出特定频率声音，8255 的 PB7 端口作为 8254 通道 2 的门控信号，使音频有所间断。其中，8254 工作在方式 2 时，当门控信号为低电平时，输出始终为高电平，为了使门控信号与输出信号一致，将输出端外接逻辑非门。整体线路连接，见图 13。

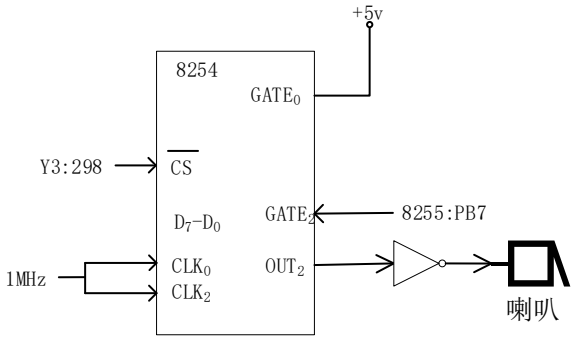


图 13　8254 连线示意图

发声函数封装在 M_8254 类中，当单次运营服务结束时，受到计价器类中旅程结束函数（Journey_End()）函数的调用，完成打印声模拟。

4.9 中断接口

程序使用查询方式扫描键盘执行任务效率较低，此外计数初值的重装与阈值设定相关有不小的误差。引入中断操作改进系统，可以省去键盘键值判断且能够实现计数初值的中断重装，大大提高系统的准确度，灵敏度。

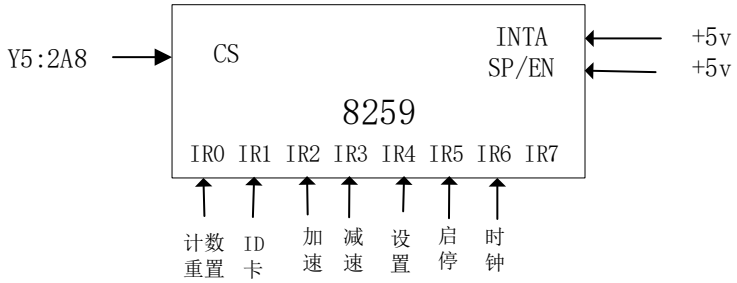


图 14　8259 改进程序接线示意图

由实验指导书[3]知，实验箱在 PC/AT 机中提供了主/从两片 8259，但开放的中断源端口仅两个，不能满足需求，故选用扩展 8259 进行中断接口的实现。在 C 语言编程环境下，使用查询中断查询字的方式来进行中断程序跳转控制。

图 14 显示了使用中断改进程序的示意图，其中最高级中断响应赋给计数初值的重置，进而当 8254 计数结束时即刻进行初值重装；第 1 级中断赋给 ID 卡插入键，在系统初始状态时接受操作；第 2、3 级中断为车辆加减速操作，不受计价器系统状态的影响；第 4 级中断赋给设置键；第 5 级中断赋给空车牌翻动操作（单脉冲实现）；第 6 级中断赋给 8254 控制下的定时器（由通道 1 实现），实现程序的定时界面刷新。

4.10 键值读取

在查询式工作方式下，系统通过拉低键盘行线，读取列线来判断所按下的键值。利用 C 语言的优势，读取列线时可以直接得知是那一列被按下（值非 1），再通过逐行拉低即可确认键值[7]。详细处理流程见图 15。

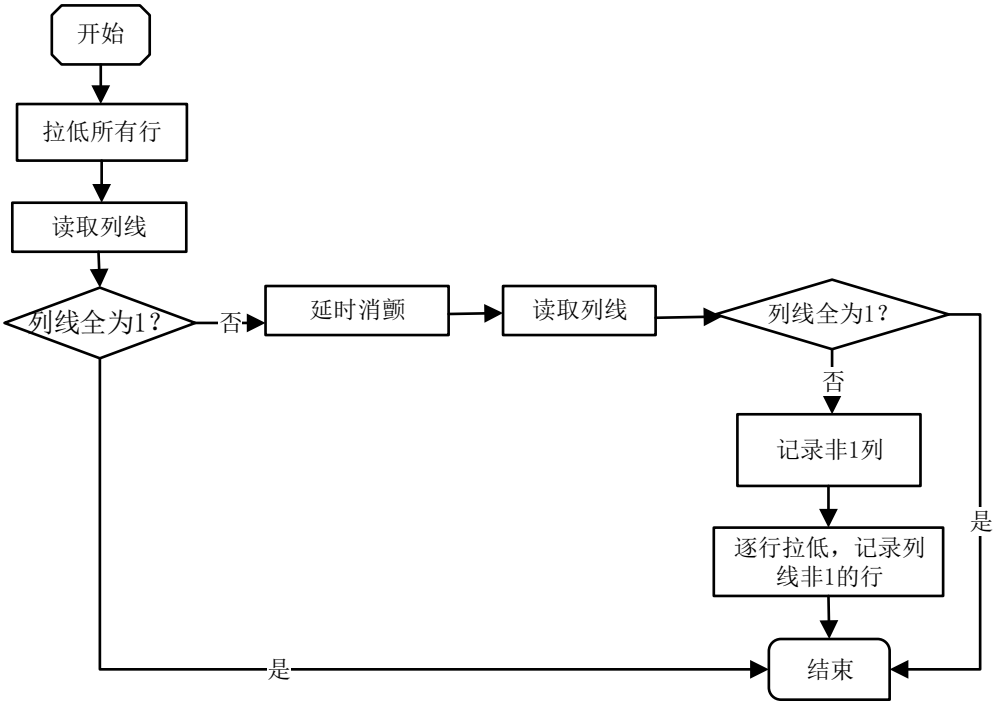


图 15　键值确认流程图

键值确认程序封装在键盘类（KeyBoard 类）中，并对外提供调用接口检查是否有按键按下，以及具体是哪一个按键。在系统工作过程中，由计价器主类调用键盘接口，并对接状态转移模块，实现控制。

4.11 录放音模块

AD0809 外接扬声器以固定时间间隔采集语音信号，并存储至文件中。当计价器执行到相应状态时，读取文件，并交 DAC0832 外接输出相应语音。录音与播放皆封装在 M_ADC0809 类中。实际设计中，通过一小段延时给 0809 足够时间转换，代替查询 EOC。

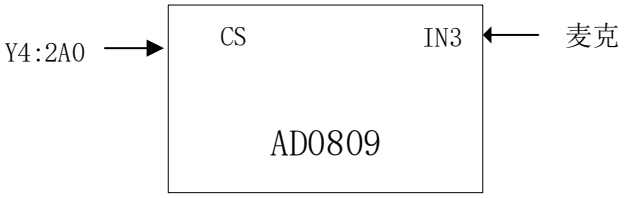


图 16　AD0809 接线示意图

AD0809 的接线情况如所示，参考资料后选用的采样频率为 6000Hz[5]。

5.系统编码测试

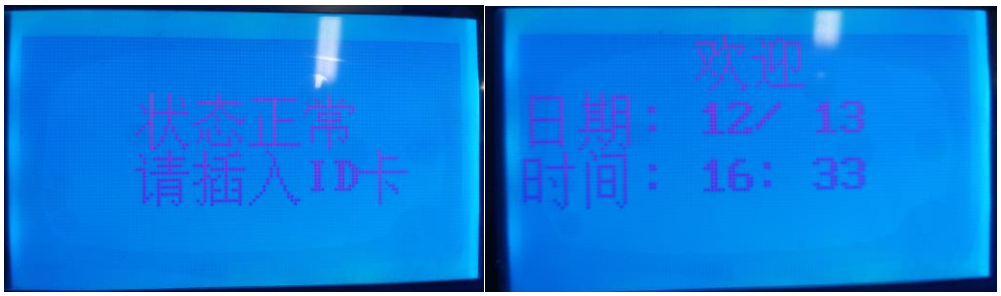
5.1 单元测试（类测试）

逐一编写各个模块类，并进行测试。在模块类的关键方法中嵌入控制台输出，进而即使没有硬件动作也能得知 C 程序的执行时序。各类模块详见附录，单元测试用例如表格 1 所示。

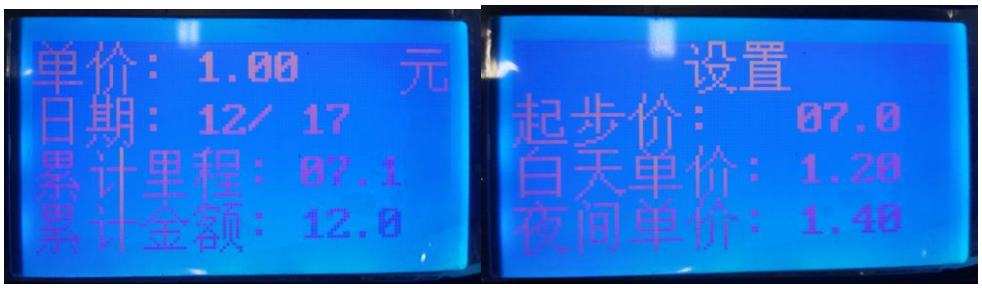
表格 1　单元测试用例

序号	测试类	测试内容	测试数据或方法	期望结果	测试结果
1	DAC0832 类	电机速度更新	0	电机停转	✓
2			200	电机高速转动	✓
3	M_rgy 类（8*8 点阵类）	空字显示	循环调用 Print_Kong()	空字稳定显示	略带闪动，总体可以✓
4		关闭显示	Print_Null()	停止显示	✓
5	M_8254 类	计数	拨动电机	可以记录每转	✓
6		喇叭声	Buzzer_Call()	发出“嗒嗒嗒”	✓
7	M_8255 类	初始化	生成对象	构造函数正常	✓
8	KeyBoard 类	检测按键按下	循环 Is_Press()	识别按键按下	✓
9		键值读取	Read()	识别键值	✓
10	LCD 类	显示	等待 ID 卡界面	正常显示	✓
11			自检界面	正常显示	✓
12			空车界面	时钟获取正常	✓
13			重车界面	正常显示	✓
14			设置界面	正常显示	✓
15		更新	更新空车界面	更新正确	✓
16			更新重车界面	更新正确	✓
17	TAXI 类（车辆类）	加速	Start_or_SpeedUp()	电机转速加快	✓
18		减速	Stop_or_SpeedDown()	电机转速减慢	✓
19		读取里程数	Read_Meter_Thistime()	读取正常	✓
20	M_8259 类	中断查询	Wait_Interrupt()	识别中断请求	✓
21	M_ADC0809 类（录放音类）	录音	Record()	录音生成文件	✓
22		放音	Play()	播放录音	✓

LCD 类为系统主要可视化界面类，在单元测试中的测试截图如图 17 所示。



(a)等待插入 ID 卡界面　　(b)空车界面



(c)重车界面　　(d)设置界面

图 17　液晶屏界面显示实物图

5.2 集成测试（系统测试）

在各模块实现的基础上，编写 TAXIMETER 类（计价器主类），整合各模块，并测试系统，整个系统测试用例见表格 2。

表格 2　系统测试用例（状态转移测试）

序号	计价器状态	测试操作	期望结果	测试结果
1	等待插入 ID	ID 键	转入自检状态	✓
2		加速、减速	正常响应	✓
3		设置、启停	不响应	✓
4	自检	任意按键	不响应	✓
5		等待 3 秒	转入空车状态	✓
6	空车	加速、减速	正常响应	✓
7		设置	转入设置界面	✓
8		ID 键	不响应	✓
9		启停	转入重车，点阵灭	✓
10		等待	自动刷新时钟、点阵	✓
11	重车	加速、减速	正常响应	✓
12		设置、ID	不响应	✓
13		启停	转入空车，点阵亮喇叭响	✓
14		等待	刷新里程与价格	✓
15	设置	加速、减速	正常响应	✓
16		设置	转入空车	✓
17		其他按键	不响应	✓
18		等待	刷新点阵	✓

整体系统演示视频如下：https://v.youku.com/v_show/id_XNDQ3NTg3ODQwMA==.html?spm=a2h3j.8428770.3416059.1

6.设计结果分析及结论

在系统的运行测试过程中，有不少与设计或实际相左的结果，其中有一些在设计阶段已经估计到了，还有一些经过分析之后，也得到了结论。

6.1 计数初值偏离

当计价器转入重车状态开始计价时，若此时计数器已经开始了工作，且电机已经开始了转动，则初值不是 0。而程序中转入重车状态后第一次计量里程时，保留的上次计数值 temp 为 0，这里会有一个由初值造成的偏移误差，范围在(0~65535)。

6.2 按键灵敏度降低

系统工作在查询式下时，设定消颤时长为 50ms，且当程序工作在其他步骤时，可能遗漏查询响应。实际测试，当手指按键的时长适当提高时，有助于识别。

7.实验体会

《硬件课程设计》**收获颇丰**：一是对于上学期所学理论课程有了非常全面的复习；二是对于完整系统软件的设计实现能力有了很大增长；三是结合同期所学《软件工程》，在设计与分析系统，撰写报告陈述思路方面得到了很好的实践；四是查阅资料，尤其是阅读硬件芯片手册的能力也有了很大的提升。

受限于课程的时间，回顾整个 48 学时，觉得仍然有一小部分的**遗憾** ，没有能够去实际的检验利用中断实现 8254 自动重装计数初值；没能够实现将录音模块嵌入主系统中，实现语音播报；时钟获取也可以考虑利用实验仪上 PC 机自动获取；设想的日志模块尚未实现……

不过总体上，仍然十分难忘整个过程。

参考文献

[1] ST7920GB 中文字型码表. (2000 年 4 月 3 日).

[2] TPC-2003A 通用 32 位微机实验系统学生用实验指导书. (2004 年 10 月).

[3] TPC-ZK 系列 USB 教师实验指导书. (2017 年 9 月 25 日). 检索来源: 原创力文档: https://max.book118.com/html/2017/0925/134943357.shtm

[4] 陈楠. (2014 年 12 月 15 日). TPC-ZK-II 实验指导书.

[5] 接口技术综合性实验报告-数字录音机. (2016 年 9 月 3 日). 检索来源: 百度文库: https://wenku.baidu.com/view/fc8137da4b73f242326c5feb

[6] 深圳亚斌显示科技有限公司. (无日期). 中文字库液晶显示模块使用手册. 深圳.

[7] 周荷琴, & 冯焕清. (2014). 微型计算机原理与接口技术 (第 5 版 版本). 合肥: 中国科学技术大学出版社.

附录

代码 1　M_rgy 类（8*8 点阵类）

```
#define port_HCS 0x2B8
#define port_RCS 0x2B0
struct M_rgy {
    int port_h;
    int port_r;
    int Sleep_time;
    M_rgy():port_h(port_HCS),port_r(port_RCS),Sleep_time(10) {}
    void Print_Kong() {
        byte Lie[8]= {0x04,0xef,0x55,0x22,0x1c,0x08,0x08,0x3e};
        byte h=0x01;
        for(int i=0; i<8; ++i) {
            PortWriteByte(port_r,Lie[i]);
            PortWriteByte(port_h,h);
            h=h<<1;
            Sleep(Sleep_time);
        }
    }
    void Print_Null() {
        PortWriteByte(port_h,0x00);
    }
};
```

代码 2　M_8254 类

```
#define port_8254 0x298
struct M_8254 {
    int port_ctr;
    int result;
    M_8254():port_ctr(port_8254+3) {
        Init_Gate0();
        Init_Gate2();
        result=0;
    }
    void Init_Gate0() { //通道0，此段计数初值可供之后中断调用
        PortWriteByte(port_ctr,0x30);//控制字：00 11 000 0通道0，二进制计数
        PortWriteByte(port_ctr-3,0x00);
        PortWriteByte(port_ctr-3,0x00);
        cout<<"8254 init, Gate 0 chosed"<<endl;
    }
    void Init_Gate1() { //通道1，此段硬延时产生一个刷新屏幕的频率
        PortWriteByte(port_ctr,0x77);//控制字：01 11 011 1通道1，10进制计数
        PortWriteByte(port_ctr-2,0x00);
```

```
19.         PortWriteByte(port_ctr-2,0x00);
20.         cout<<"8254 init, Gate 1 chosed"<<endl;
21.     }
22.     void Init_Gate2() { //通道2，用于给蜂鸣器输出打印声（未完全）
23.         PortWriteByte(port_ctr,0xB7);//控制字：10 11 011 1通道2，方式3，BCD计数
24.         PortWriteByte(port_ctr-1,0xe9);
25.         PortWriteByte(port_ctr-1,0xf1);
26.         cout<<"8254 init, Gate 2 chosed"<<endl;
27.         PortWriteByte(0x289,0x00);//默认为不发声；
28.     }
29.     void Buzzer_Call() {
30. //8255 B7连Gate2，out连非门再输出（因为8254工作在方式3时，Gate为0默认为不计数输出
    恒高）
31.         cout<<"Buzzer_Call init"<<endl;
32.         for(int i=0; i<8; ++i) {
33.             PortWriteByte(0x289,0x80);
34.             Sleep(150);//对应发声时间
35.             cout<<"end:"<<endl;
36.             PortWriteByte(0x289,0x00);
37.             Sleep(30);//对应的是沉默时间
38.         }
39.         cout<<"Buzzer_Call ended"<<endl;
40.     }
41.     int read() {
42.         PortWriteByte(port_ctr,0x00);//控制字：00 00 000 0通道0锁存
43.         byte L;
44.         byte H;
45.         PortReadByte(port_ctr-3,&L);
46.         PortReadByte(port_ctr-3,&H);
47.         result=65536-(256*H+L);
48.         cout<<"8254计数补值为："<<result<<endl;
49.         if(result>60000)Init_Gate0();
50.         return result;
51.     }
52. };
```

代码 3　M_8255 类

```
1. #define port_8255 0x288
2. struct M_8255 {
3.     int port;
4.     M_8255():port(port_8255) {
5.         PortWriteByte(port+3,0x81);
6.     }
7. };
```

代码 4　KeyBoard 类

```
#define port_8255_C 0x28A
struct KeyBoard {
    /**键盘工作流程
    *1. 当行线都拉低时，正常列线应该全为1
    *2. 若出现非1，说明改行按键被按下了，改为逐行拉低去检测是哪一行出现非1
    *3. 继续拉低该行，继续去检测是哪一列非1，在c中，哪一列可以直接计算得到
    **/
    int port_c;
    int result;
    int row;
    int col;
    int ret;//返回按键数值
    byte condition;
    KeyBoard():port_c(port_8255+2) {
        init();
        cout<<"keyboard started."<<endl;
    }
    void init() {
        PortWriteByte(port_c+1,0x81);//控制字：1 00 0 0 00 1
    }
    bool IsPRES() {
        PortWriteByte(port_c,0x00);//拉低所有行
        PortReadByte(port_c,&condition);
        if((int)condition%16==15)return false;
        else {
//先引入一步“消颤”，根据实验，
//没有这一步会在一次按下中，多次识别
            Sleep(50);
            PortReadByte(port_c,&condition);
            if((int)condition%16==15) {
                return false;
            }
            //change();
            //cout<<"PRES!"<<endl;
            //read();
            return true;
        }
    }
    void read() {
        int a=15-condition%15;
        col=0;
        while(a!=0) {
```

```
            a/=2;
            col++;
        }
//逐行拉低，读取非1时的该行，同时去计算是哪一列被按下了
        byte line=0xEF;
        row=0;
        do {
            PortWriteByte(port_c,line);
            PortReadByte(port_c,&condition);
            ++row;
            line=line<<1;
        } while((int)condition%16==15);

        if(col==1) {
            if(row==1)ret=1;
            else if(row==2)ret=2;
            else if(row==3)ret=3;
        } else if(col==2) {
            if(row==1)ret=4;
            else if(row==2)ret=5;
            else if(row==3)ret=6;
        }
        cout<<"the board is "<<row<<" "<<col<<", ret is "<<ret<<endl;
    }
    int Pressed_Button() {
        //这个函数作为查询式的一个接口函数，始终等待按键按下，一旦按下返回内置的键值
        while(!IsPRES());
        read();
        cout<<"Button "<<ret<<" is Pressed."<<endl;
        return ret;
    }
};
```

代码 5　LCD 类

```
#include"time.h"
#define port_74LS273 0x290
struct LCD:public M_8255 {
    int port_data;
    int port_ctr;
    int Addr_start;
    int Youbiao_isON;
    LCD():M_8255(),port_data(port_74LS273),port_ctr(port_8255+1),Addr_start(128) ,Youbiao_isON(0) {};
```

```
9.      void cmd_setup() {
10.         PortWriteByte(port_ctr,0x00);
11.         PortWriteByte(port_ctr,0x04);
12.         PortWriteByte(port_ctr,0x01);
13.     }
14.     void Clear_All() {
15.         PortWriteByte(port_data,0x01);
16.         cmd_setup();
17.     }
18.     void data_setup() {
19.         PortWriteByte(port_ctr,0x01);
20.         PortWriteByte(port_ctr,0x05);
21.         PortWriteByte(port_ctr,0x01);
22.     }
23.     void Display_Line(int n,int data[8]) {
24.         switch(n) {
25.             case 1:
26.                 Addr_start=128;
27.                 break;
28.             case 2:
29.                 Addr_start=144;
30.                 break;
31.             case 3:
32.                 Addr_start=136;
33.                 break;
34.             default:
35.                 Addr_start=152;
36.         }
37.         for(int i=0; i<8; i++) {
38.             PortWriteByte(port_data,Addr_start);
39.             cmd_setup();
40.             PortWriteByte(port_data,data[i]>>8);
41.             data_setup();
42.             PortWriteByte(port_data,data[i]%256);
43.             data_setup();
44.             Addr_start++;
45.         }
46.     }
47.     void Display_Update_Car_Empty() {
48.         struct tm *local;
49.         time_t t=time(NULL);
50.         local=localtime(&t);
51.         cout<<"Date: "<<local->tm_mon<<" / "<<local->tm_mday<<endl;
52.         cout<<"Time: "<<local->tm_hour<<" : "<<local->tm_min<<endl;
```

```
53.
54.         int B_mon,B_mday,B_hour,B_min;
55.         B_mon=0x3030+((local->tm_mon+1)/10)*256+(local->tm_mon+1)%10;
56.         B_mday=0x3030+(local->tm_mday/10)*256+local->tm_mday%10;
57.         B_hour=0x3030+(local->tm_hour/10)*256+local->tm_hour%10;
58.         B_min=0x3030+(local->tm_min/10)*256+local->tm_min%10;
59.         Display_Update_ch(2,3,B_mon);
60.         Display_Update_ch(2,5,B_mday);
61.         Display_Update_ch(3,3,B_hour);
62.         Display_Update_ch(3,5,B_min);
63.     }
64.     void Display_Update_ch(int n,int r,int data) {
65.         switch(n) {
66.             case 1:
67.                 Addr_start=128;
68.                 break;
69.             case 2:
70.                 Addr_start=144;
71.                 break;
72.             case 3:
73.                 Addr_start=136;
74.                 break;
75.             default:
76.                 Addr_start=152;
77.         }
78.         PortWriteByte(port_data,Addr_start+r);
79.         cmd_setup();
80.         PortWriteByte(port_data,data>>8);
81.         data_setup();
82.         PortWriteByte(port_data,data%256);
83.         data_setup();
84.     }
85.     void Display_Youbiao_ON() {//显示游标
86.         PortWriteByte(port_data,0x0F);
87.         cmd_setup();
88.     }
89.     void Display_Youbiao_OFF() {
90.         PortWriteByte(port_data,0x0C);
91.         cmd_setup();
92.     }
93.     void Display_Self_Check() {
94. //
95. //开机自检，请稍候
96. //
```

```
97.  //
98.          Clear_All();
99.          int Data_Self_Check[8]=
     {0xBFAA,0xBBFA,0xD7D4,0xBCEC,0xA3AC,0xC7EB,0xC9D4,0xBAF2};
100.         Display_Line(2,Data_Self_Check);
101.     }
102.     void Display_Wait_ICcard() {
103. //
104. //__ __ 状态正常 __ __
105. //__ __ 请插入IC 卡 __
106. //
107.         Clear_All();
108.         int Data_Condition_OK[8]=
     {0xA1A0,0xA1A0,0xD7B4,0xCCAC,0xD5FD,0xB3A3,0xA1A0,0xA1A0};
109.         int Data_Wait_ICcard[8]=
     {0xA1A0,0xA1A0,0xC7EB,0xB2E5,0xC8EB,0x4944,0xBFA8,0xA1A0};
110.         Display_Line(2,Data_Condition_OK);
111.         Display_Line(3,Data_Wait_ICcard);
112.     }
113.     void Display_Setup() {
114. //__ __ __ 设 置 __ __ __
115. //起 步 价 :_ __ 07 .0 __
116. //白 天 单 价 :_ 1. 20 __
117. //夜 间 单 价 :_ 1. 40 __
118.         int Data_Setup[8]=
     {0xA1A0,0xA1A0,0xA1A0,0xC9E8,0xD6C3,0xA1A0,0xA1A0,0xA1A0};
119.         int Data_Starting_Fare[8]=
     {0xC6F0,0xB2BD,0xBCDB,0x3A00,0xA1A0,0x3037,0x2E30,0xA1A0};
120.         int Data_PerPrice_Day[8]=
     {0xB0D7,0xCCEC,0xB5A5,0xBCDB,0x3A00,0x312E,0x3230,0xA1A0};
121.         int Data_PerPrice_Night[8]=
     {0xD2B9,0xBCE4,0xB5A5,0xBCDB,0x3A00,0x312E,0x3430,0xA1A0};
122.         Display_Line(1,Data_Setup);
123.         Display_Line(2,Data_Starting_Fare);
124.         Display_Line(3,Data_PerPrice_Day);
125.         Display_Line(4,Data_PerPrice_Night);
126.     }
127.     void Display_Car_Empty() {
128. //__ __ __ 欢 迎 __ __ __
129. //日 期 :_ 12 /_ 09 __ __
130. //时 间 :_ 15 :_ 30 __ __
131. //
132.         int Data_Welcome[8]=
     {0xA1A0,0xA1A0,0xA1A0,0xBBB6,0xD3AD,0xA1A0,0xA1A0,0xA1A0};
```

```
133.        int Data_Date[8]=
    {0xC8D5,0xC6DA,0x3A00,0x3132,0x2F00,0x3039,0xA1A0,0xA1A0};
134.        int Data_Time[8]=
    {0xCAB1,0xBCE4,0x3A00,0x3135,0x3A00,0x3330,0xA1A0,0xA1A0};
135.        int Blank[8]=
    {0xA1A0,0xA1A0,0xA1A0,0xA1A0,0xA1A0,0xA1A0,0xA1A0,0xA1A0};
136.        Display_Line(1,Data_Welcome);
137.        Display_Line(2,Data_Date);
138.        Display_Line(3,Data_Time);
139.        Display_Line(4,Blank);
140.
141.        Display_Update_Car_Empty();
142.    }
143.    void Display_Update_Car_Full(double Mileage_Thistime,double
    Price_Thistime) {
144.        int Meter_0,Meter_1;
145.        Meter_0=0x2E30+(int)(Mileage_Thistime*10)%10;
146.
    Meter_1=0x3030+(int)Mileage_Thistime/10*256+((int)Mileage_Thistime*10%1
    00/10);
147.        Display_Update_ch(3,6,Meter_0);
148.        Display_Update_ch(3,5,Meter_1);
149.
150.        int Money_0,Money_1;
151.        Money_0=0x2E30+(int)(Price_Thistime*10)%10;
152.
    Money_1=0x3030+(int)Price_Thistime/10*256+((int)Price_Thistime*10%100/1
    0);
153.        Display_Update_ch(4,6,Money_0);
154.        Display_Update_ch(4,5,Money_1);
155.    }
156.    void Display_Car_Full(double Per_Price) {
157. //单价 :_ 1. 20 __ __ 元   如: 7.0
158. //时间 :_ 00 :_ 23 __ __  如: 1:23   //行驶时间暂时去掉，优先实现完备系统，刚需
    为里程与计价
159. //累计 里 程 :_ __ ._ __
160. //累计 金 额 :_ __ ._ __
161.        int Data_Per_Price[8]=
    {0xB5A5,0xBCDB,0x3A00,0x312E,0x3230,0xA1A0,0xA1A0,0xD4AA};
162.        //int Data_Total_Time[8]=
    {0xCAB1,0xBCE4,0x3A00,0x3030,0x3A00,0x3031,0xA1A0,0xA1A0};
163.        int Data_Total_Meter[8]=
    {0xC0DB,0xBCC6,0xC0EF,0xB3CC,0x3A00,0x3030,0x2E30,0xA1A0};
164.        int Data_Total_Money[8]=
```

```
    {0xC0DB,0xBCC6,0xBDF0,0xB6EE,0x3A00,0x3037,0x2E30,0xA1A0};
165.        Display_Line(1,Data_Per_Price);
166.        //Display_Line(2,Data_Total_Time);
167.        Display_Line(3,Data_Total_Meter);
168.        Display_Line(4,Data_Total_Money);
169.        int Per_p_1=0x302E+(int)Per_Price*256;
170.        int Per_p_0=0x3030+(int)10*(Per_Price-(int)Per_Price)*256;
171.        Display_Update_ch(1,3,Per_p_1);
172.        Display_Update_ch(1,4,Per_p_0);
173.    }
174. };
```

代码 6　DAC0832 类

```
1.  #define port_DAC0832 0x280
2.  struct M_DAC0832 {
3.      int port_ctr;
4.      int speed;
5.      M_DAC0832():port_ctr(port_DAC0832),speed(0) {
6.          PortWriteByte(port_ctr,speed);
7.          cout<<"M_0832 init, Speed_0 = 0"<<endl;
8.      }
9.      void Speed_Update(int sp) {
10.         PortWriteByte(port_ctr,sp);
11.         cout<<"Speed update, Speed_now = "<<speed<<endl;
12.     }
13. };
```

代码 7　TAXI 类（车辆类）

```
1.  struct TAXI:public M_DAC0832,M_8254 {
2.  //TAXI 类继承M_DAC0832芯片,M_8254芯片,在此类中完成里程数计算
3.  //继承speed与加减速，同时自身带有数据成员state状态
4.  //state:0-未启动，1-运行，10-静止等待（运行中，车速0）
5.      int state;
6.      double Meter_Total;
7.      double Meter_Thistime;
8.      double temp;
9.      double factor;
10.     TAXI():M_DAC0832(),state(0),Meter_Thistime(0),Meter_Total(0),temp(0),fac
    tor(1440) {};
11.     double Read_Meter_Thistime() {
12.         int Now=this->read();
13.         //8254计数是倒计数，这里在装入下一个计数值时会出现较大误差
14.         //在出租车速度不快时没有问题，当速度达到五档时有很大误差。
15.         if(Now>=temp) {
```

```
16.             Meter_Thistime+=(double)(Now-temp)/factor;
17.             temp=Now;
18.         } else {
19.             Meter_Thistime+=(double)(Now)/factor;
20.             temp=Now;
21.         }
22.         return Meter_Thistime;
23.     }
24.     double Read_Meter_Total() {
25.         return Meter_Total+Meter_Thistime;
26.     }
27.     void Thistime_Start() {
28.         Meter_Thistime=0;
29.         temp=0;
30.     }
31.     int Thistime_End() {
32.         Meter_Total+=Read_Meter_Thistime();
33.         cout<<"本次运行结束，总里程为："<<Meter_Thistime<<endl;
34.         return Meter_Thistime;
35.     }
36.     void Start_or_SpeedUp() {
37.         if(state==0) {
38.             state=1;//汽车启动
39.             speed=90;
40.             this->Speed_Update(speed);
41.             return;
42.         }
43.         if(speed<245)speed+=10;
44.         else speed=254;
45.         this->Speed_Update(speed);
46.         if(state==10) {
47.             state=1;
48.             cout<<"Taxi continue runing"<<endl;
49.         }
50.     }
51.     void Stop_or_SpeedDown() {
52.         if(state==10) {
53.             state=0;
54.             cout<<"Taxi stop"<<endl;
55.             return;
56.         }
57.         if(speed>70)speed-=10;
58.         else speed=61;
59.         Speed_Update(speed);
```

```
        if(speed==61) {
            state=10;
            cout<<"Taxi wait"<<endl;
        }
    }
};
```

代码 8　TAXIMETER 类（计价器主类）

```
struct TAXIMETER {
    int state;
    /**state 代表计价器的状态
    *0 IC卡未插
    *10 重车状态
    *11 空车状态
    *12 自检状态
    *13 设置状态
    *14 计价暂停状态
    **/
    double Per_Price_Day;
    double Per_Price_Night;
    double Per_Price_Now;

    double Starting_Price;
    int Total_Times;//累积载客次数
    int Total_Price;
//单次营运中的一些参数
    int Money_Thistime;
    int Time_Start;
    int Time_End;

    LCD lcd;
    M_8255 m_8255;
    M_8254 m_8254;
    KeyBoard kb;
    TAXI taxi;
    M_rgy rgy;

    TAXIMETER():state(0) {
        History_Read();
        //Wait_IC();
    };
    ~TAXIMETER() {
//析构函数中可以加“写硬盘”
    }
```

```
37.     void History_Read() {
38.         //此步可以使用存储器进行断电存储
39.         Per_Price_Day=1.2;
40.         Per_Price_Night=1.4;
41.         taxi.Meter_Total=123;
42.         Total_Price=374;
43.         Total_Times=12;
44.         Starting_Price=7;
45.         cout<<"Read Log OK, "<<Per_Price_Day<<" "<<Per_Price_Night<<" "
46.             <<taxi.Meter_Total<<" "<<Total_Price<<" "<<Total_Times<<"
"<<endl;
47.     }
48.     void History_Write() {}//写入参数，可以使用存储器扩充
49.     double Caculate_Money_Thistime(double meter) {
50.         double M;
51.         if(meter<3)M=7;
52.         else {
53.             M=Per_Price_Now*(meter-3)+7;
54.             M*=10;
55.             M=(int)(M+0.5);
56.             M=(double)M/10;//精确到小数点后一位
57.         }
58.         cout<<"Meter:  "<<meter<<",Money:  "<<M<<endl;
59.         return M;
60.     }
61.     void Journey_Start() {
62.         state=10;
63.         struct tm *local;
64.         time_t t=time(NULL);
65.         local=localtime(&t);
66.
    if(local->tm_hour>18||local->tm_hour<6)Per_Price_Now=Per_Price_Night;
67.         else Per_Price_Now=Per_Price_Day;
68.         rgy.Print_Null();
69.         taxi.Thistime_Start();
70.         lcd.Display_Car_Full(Per_Price_Now);
71.     }
72.     void Journey_End() {
73.         state=11;
74.         rgy.Print_Kong();
75.         taxi.Thistime_End();
76.         m_8254.Buzzer_Call();
77.         Sleep(1000);
78.         lcd.Display_Car_Empty();
```

```
79.     }
80.     void State_Change(int ret) {
81.         //这段函数根据键盘操作执行状态之间的跳转代码，类似于事务中心的转移
82.         switch(state) {
83.             case 11: {
84.                 //空车
85.                 if(ret==6)taxi.Start_or_SpeedUp();
86.                 else if(ret==3)taxi.Stop_or_SpeedDown();
87.                 else if(ret==1) {
88.                     Journey_Start();
89.                 } else if(ret==2) {
90.                     state=13;
91.                     lcd.Display_Setup();
92.                 } else return;
93.                 break;
94.             }
95.             case 10: {
96.                 //重车状态，仅可以加减速
97.                 if(ret==6)taxi.Start_or_SpeedUp();
98.                 else if(ret==3)taxi.Stop_or_SpeedDown();
99.                 else if(ret==1) {
100                    Journey_End();
101                }
102                return;
103            }
104            case 13: {
105                //设置状态，仅可以设置或者加减速
106                if(ret==6)taxi.Start_or_SpeedUp();
107                else if(ret==3)taxi.Stop_or_SpeedDown();
108                else if(ret==2) {
109                    if(lcd.Youbiao_isON==0) {
110                        lcd.Display_Youbiao_ON();
111                        lcd.Youbiao_isON=1;
112                    } else if(lcd.Youbiao_isON==1) {
113                        lcd.Display_Youbiao_OFF();
114                        lcd.Youbiao_isON=2;
115                    } else if(lcd.Youbiao_isON==2) {
116                        lcd.Display_Car_Empty();
117                        lcd.Youbiao_isON=0;
118                        state=11;
119                    }
120                }
121                //else if(ret==5);
122                else return;
```

```
123             }
124             default:
125                 ;
126         }
127     }
128     void Wait_IC() {
129         lcd.Display_Wait_ICcard();
130 //接下来查询等待键盘插入IC（按键模拟）
131         while(kb.Pressed_Button()!=4);
132         state=12;
133         Self_Check();
134     }
135     void Self_Check() {
136         lcd.Display_Self_Check();
137         Sleep(1500);
138         state=11;
139         lcd.Display_Car_Empty();
140         Update();
141 //等待空车牌被翻下，或是按键被按下
142     }
143     void Update() {
144         while(1) {
145             if(kb.IsPRES()) {
146                 kb.read();
147                 State_Change(kb.ret);
148             } else {
149                 switch(state) {
150                     case 11:
151                         lcd.Display_Update_Car_Empty();
152                         rgy.Print_Kong();
153                         break;
154                     case 13://此处本为设置，可以加一些反白的操作
155                         rgy.Print_Kong();
156                         break;
157                     case 10:
158 
    lcd.Display_Update_Car_Full(taxi.Read_Meter_Thistime(),Caculate_Money_Th
    istime(taxi.Meter_Thistime));
159                         break;
160                     default:
161                         ;
162                 }
163             }
164         }
```

```
165     }
166 };
```

代码 9　M_ADC0809 类（录放音类）

```
1.  #define port_ADC0809 0x2A0
2.  #define port_DAC0832 0x280
3.  using namespace std;
4.  struct M_ADC0809 {
5.      BYTE *Data_Point;//数据区指针
6.      int port_change;
7.      M_ADC0809():port_change(port_ADC0809+3) { //以IN3口转换为例
8.          Data_Point = (BYTE *)malloc(6000); /*分配空间用于存放录音数据*/
9.          if(!Data_Point) {
10.             printf("No memory!\7");
11.             exit(0);
12.         }
13.     }
14.     void Record() {
15.         BYTE data;
16.         printf("Press any key to record!\n"); /*录音提示*/
17.         getch();
18.         printf("录音中……\n");   /*录音提示*/
19.         fstream file;
20.         file.open("SoundOut.txt",ios::out);
21.         for(int i=0; i<30000; i++) {
22.             /*启动A/D,采集6000个数据放在开辟的内存空间中*/
23.             PortWriteByte(port_change,0);//特色的软启动
24.             //delay();
25.             PortReadByte(port_change,&data);
26. //省略检测EOC，等待1ms后便直接读取转换后的数值
27.             *(Data_Point+i) = data;
28.             file<<data<<endl;
29.         }
30.         file.close();
31.         printf("录音已结束。\n");    /*录音提示*/
32.     }
33.     void Play() {
34.         printf("Press any key to playing!\n");/*放音提示*/
35.         getch();
36.         BYTE data;
37.         fstream file;
38.         file.open("SoundOut.txt",ios::in);
39.         for(int i=0; i<30000; i++) {
40.             /*将ii中的6000个从D/A输出*/
```

```
            data = *(Data_Point+i);
            PortWriteByte(port_DAC0832,data);
        }
        file.close();
        printf("Playing end!\n");
    }
    void delay() {
        byte d;
        do {
            PortReadByte(0x28a,&d);
        } while(d&1!=0);
        do {
            PortReadByte(0x28a,&d);
        } while(d&1!=1);
    }
};
```

代码 10　M_8259 类

```
#define port_8259 0x2A8
struct M_8259 {
    int port_odd;
    int port_even;
    M_8259():port_odd(port_8259+1),port_even(port_8259) {
        Init();
        InterruptMask_Open();
        Wait_Interrupt();
    }
    void InterruptMask_Open() {
        PortWriteByte(port_odd,0x00);//写入OCW1开中断
    }
    void Interrupt_End() {
        //写入OCW2为00100000来结束刚才服务过的中断
        PortWriteByte(port_even,0x20);
        Wait_Interrupt();
    }
    void IR0() {
        cout<<"中断0"<<endl;

        Interrupt_End();
    }
    void IR1() {
        cout<<"中断1"<<endl;
        Interrupt_End();
    }
```

```
27.     void Init() {
28.         PortWriteByte(port_even,0x13);//ICW1
29.         PortWriteByte(port_odd,0xB0);//ICW2
30.         PortWriteByte(port_odd,0x03);//ICW4,自动结束中断方式
31.     }
32.     int Read_ISR() {
33.         byte data;
34.         PortWriteByte(port_even,0x0B);
35.         PortReadByte(port_even,&data);
36.         return data;
37.     }
38.     void Wait_Interrupt() {
39.         cout<<"Wait  Interrupt...."<<endl;
40.         while(Read_ICheck()==-1) {
41.             Sleep(20);
42.         }
43.     }
44.     int Read_IRR() {
45.         byte data;
46.         PortWriteByte(port_even,0x0A);
47.         PortReadByte(port_even,&data);
48.         return data;
49.     }
50.     int Read_ICheck() {
51.         //读取中断查询字，最高位表示有无中断，低三位显示是哪一级中断
52.         byte data;
53.         PortWriteByte(port_even,0x0C);
54.         PortReadByte(port_even,&data);
55.         //cout<<(int)data<<endl;
56.         if((int)data/128==1) {
57.             switch((int)data%8) {
58.                 case 0:
59.                     IR0();
60.                     break;
61.                 case 1:
62.                     IR1();
63.                 default:
64.                     ;
65.             }
66.             return (int)data%8;
67.         } else return -1;
68.     }
69.     int Read_IMR() {
70.         byte data;
```

```
71.         PortReadByte(port_odd,&data);
72.         return data;
73.     }
74. };
```

代码 11　主文件代码

```
1.  #include <conio.h>
2.  #include <stdio.h>
3.  #include "ApiExUsb.h"
4.  #pragma comment(lib,"ApiExUsb.lib")
5.  #include<iostream>
6.  using namespace std;
7.
8.  #include"TAXIMETER.h"
9.  int main(){
10.     Cleanup();
11.     if(Startup()){
12.         cout<<"right"<<endl;
13.     }
14.     TAXIMETER tm;
15.     tm.Wait_IC();
16.     Cleanup();
17.     return 0;
18. }
```