东南大学学位论文独创性声明

本人声明所呈交的学位论文是我个人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得东南大学或其它教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

研究生签名：李成萌 日　期：07年12月20日

东南大学学位论文使用授权声明

东南大学、中国科学技术信息研究所、国家图书馆有权保留本人所送交学位论文的复印件和电子文档，可以采用影印、缩印或其他复制手段保存论文。本人电子文档的内容和纸质论文的内容相一致。除在保密期内的保密论文外，允许论文被查阅和借阅，可以公布（包括刊登）论文的全部或部分内容。论文的公布（包括刊登）授权东南大学研究生院办理。

研究生签名：李成萌 导师签名：[illegible] 日　期：07.12.20

摘 要

税控收款机是一种能够记录有关税务数据、开具和打印发票以及其它税收报表的收款机，是国家金税工程带动的新兴产业。但是，目前市场上的主流税控机多采用 8 位或 16 位单片机，并且不带有操作系统，功能简单，操作不便。针对此现状，我们设计和开发了一种基于 32 位 ARM7 处理器并且带有 μCLinux 操作系统的嵌入式税控收款机，以满足目前市场的需要。本文详细介绍了这种嵌入式收款机的软件部分的设计与实现。

文章首先描述了目前市场上流行的税控收款机的类型和技术水平，结合本次设计的地税版的税控收款机分析了系统的功能需求，并设计了系统软件和应用软件的总体方案。

接着，全文根据总体的设计方案逐步展开，分别介绍了系统软件和应用软件的具体设计和实现。系统软件设计充分考虑了嵌入式硬件资源平台和系统的功能需求，采用了开源软件 MiniGUI 和 SQLite 作为本次的开发工具，文中介绍了两种软件的体系结构、详细移植过程以及界面和数据库的详细设计方案。应用软件部分采用了模块化和层次化的设计方法，根据软件工程的模型，重点从软件的模块划分及各部分的功能、税控流程的具体实现以及应用程序的移植过程这三个方面进行详细描述。这种模块化和分层结构的设计方式使各层次和各功能模块之间相对独立，有利于系统的维护与改进，并使其具有良好的可扩展性。

然后，本文对 Linux 和 Windows 下的调试方法分别做了介绍，并总结了在调试过程中出现的问题及解决方法。

最后，对本文的工作做了一个总结，并对后续的开发工作指出了方向，对税控收款机的研发做了相关展望。

关键词：嵌入式系统，税控收款机，μCLinux，软件设计，MiniGUI，SQLite，模块化

Abstract

Fiscal cash register is the cash register which is used for recording tax data, making out and printing invoice and other report forms of tax revenue. It represents a novel industrial branch promoted by the National Gold Tax Project. While the MCUs of fiscal cash register available on today's market are mainly 8 or 16-bit and don't possess operating system, which results in the poor function and inconvenient manipulation,etc. Against the status quo, we design and develop an embedded fiscal cash register which is based on 32-bit ARM CPU and μCLinux operation system to meet the needs of current market. The paper introduces the design and realization of embedded fiscal cash register software in detail.

Firstly, the paper describes the types and technical levels of the fiscal cash register which is popular in modern market. After that, combined with the government version, the functional requirement is analyzed and integration design of the system software and application software is raised.

Secondly, the paper unfolds gradually according to the overall project plan, including the design and implement of the system software and application software. The scheme of the system software takes full account of the hardware platform and the functional requirements of the whole system, in which open-source software MiniGUI and SQLite are adopted as the development tools.In this part, the software architecture, transplant process and the interface and database design are discussed in detail. Application software is partly based on the modular and hierarchical design method. According to the model of software engineering, the paper describes the application software in three aspects, that is, software modules and corresponding functions, the realization of the tax control processes and the transplant of the applications. The modular and hierarchical structure has good independency and easy-porting capability and brilliant extended performance.

Thirdly, the debugging methods under Linux and Windows are introduced respectively. In addition, the article summarizes the problems in debugging and solutions.

Finally, a brief summarize is described and the following research work is discussed.

Keywords: Embedded systems, Fiscal cash register, μCLinux, Software design, MiniGUI, SQLite, Modular

目 录

第1章 绪论

1.1 课题的研究背景

我国是个发展中国家，社会主义市场经济和社会主义法制正处在不断建立和完善阶段，公民的依法纳税意识还有待提高。近年来，随着我国市场经济的快速发展和科学技术水平的不断提高，商业、服务业、娱乐业等许多行业开始普遍使用计算机软件和收款机等技术手段，加强财务管理和监督，大大提高了企业现代化管理水平，同时也为我国推行使用税控收款机创造了必要的条件。为进一步加强税收征管和财务监督，保障国家财政收入，维护正常的社会经济秩序，提高纳税人财务管理水平，国家各有关部门广泛推行使用税控收款机及相关的一系列税控管理软件[1]。

税控收款机是综合了税务机关的管理和纳税户的使用两方面的需求而开发出来的产品。因此，它一方面具有使用场所所需的各项功能，便于用户经营、使用和业务管理，同时又能满足税务机关对经营用户的监控、开票和税收征管方面的要求。作为一种监控手段，税控收款机可以实时监控纳税人的经营状况，为政府提供真实可靠的核税资料，从而制定公平合理的税收政策[2]。

早在2003年，由国家税务总局和信息产业部联合起草的《税控收款机国家标准》获得审批，2005年，税控收款机市场已全面启动，据国家商业局不完全统计，全国零售业为三千万，餐饮娱乐、服务行业企业更是数量众多[3]。为了更好的控制税收，减少偷税漏税，国家必然会采取强制措施推广税控收款机的使用，因此，税控市场是非常巨大的。按照一台税控收款机的单价为2000元计算，税控收款机的市场将可以达到600亿元以上，市场对税控设备的需求量呈现出爆炸式的增长。如果再加上与之配套的相关的软硬件、设备、服务、培训等，在3年~5年内，税控机领域将会形成一个至少2000亿元人民币规模的市场[4]。

1.2 相关技术研究现状

1.2.1 税控收款机的类型

一、按行业分

目前我国各省市自治区税务机关分成国税与地税两个系统，餐饮、娱乐、服务业属于“地税”系统管辖，而零售、批发业属于“国税”系统[3]。税控收款机行业内常说的“国税”收款机和“地税”收款机实际上就是指零售业使用的税控收款机和服务业使用的税控收款机。由于不同行业有不同的经营和管理方式，因此，对税控收款机也有不同的功能需求。

二、按档次分

我国目前公布的税控收款机标准中包含了两种产品——税控收款机(GB18240.1)和税控器(GB18240.3)。其实，即使同样是符合GB18240.1国家标准的税控收款机也还是有不同档次、不同用途之分的。在这里先对税控收款机做一个简单的归类。

1. 高端税控收款机（金融税控收款机）

一般是指具有PC硬件平台或至少32位处理器，配置“重频度”打印机单元，具有丰富的外设接口和实时联网能力以及银行支付终端的税控收款机。这类税控收款机的应用软件通常都运行于独立的操作系统之上。除满足税控功能外，还具有非常强大的用户经营管理功能。这种嵌入式的税控机比较多，如航天信息股份有限公司自主研发的AAH-1000税控收款机[5]，如图1-1：

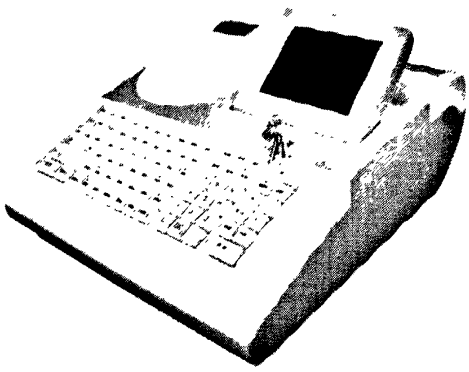

图 1-1 航天信息股份有限公司自主研发的 AAH-1000 税控收款机

2. 标准型税控收款机

采用8或16位微处理器的整合型硬件平台，配置“重频度”打印机单元，有一定的外设支持能力和通讯功能，除满足税控功能外还具有专业化和完善的用户经营管理功能。如图1-2（江苏紫金万成公司生产的WSK-3102税控收款机）：

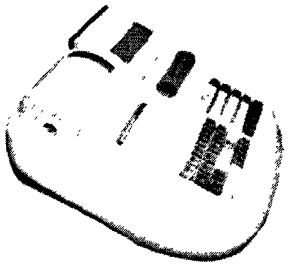

图 1-2 江苏紫金万成公司生产的标准型税控机

3. 经济型税控收款机

采用8或16位微处理器的整合型硬件平台，配置“轻频度”打印机单元，有简单的外设支持能力和通讯功能，在满足全部税控功能的基础上有一定的经营管理功能[6]。如 HX-857微型税控收款机是适应小型餐饮娱乐业特点的税控收款机，它采用专用的CPU处理模块，能够满足不同用户的需求，而且由于其功能不是很强大，所以价格不是很贵，适合一些小型餐饮娱乐业使用，图1-3是HX-857微型税控收款机实物图：

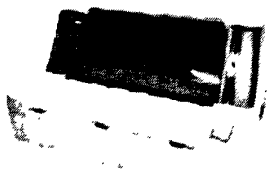

图 1-3 HX-857 微型税控收款机

4. 专用型税控收款机

为某一特殊行业或某一类特殊用户设计，能满足全部税控功能，但经营管理功能较单一[6]。如专门为加油站计费收税而设计的一款专用打印机，如“大自然SK-II加油税控机”，这种税控机不但外围接口比较少，而且功能也比较单一，如下面这个税控机是专为加油站而设计的，不是现在主流税控机的发展方向。

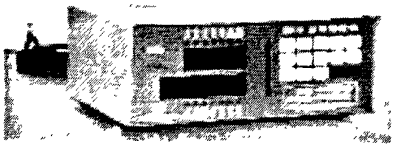

图 1-4 大自然 SK-II 加油税控机

1.2.2 税控收款机的技术水平

从以上分类情况看，目前市场上的税控收款机多使用 8 位或 16 位单片机作为微控制器，并且不采用操作系统，因此出现功能简单、存储能力弱，操作不方便等缺点[7]。基于以下考虑，目前各省份的税控收款机招标都要求使用 32 位处理器：

首先，不论从硬件实现角度还是从软件实现角度来看，税控 POS 系统都是非常复杂的体系，不同于通常的 POS 终端，税控 POS 系统不但要满足如今社会信息化的各种要求，还要能满足我国现阶段对税源监控的严格要求[8]。

其次，嵌入式 ARM 处理器集成度等方面的独特优越性和相对低廉的价位，也是我们选择用它来实现税控 POS 系统的重要依据。

再者，嵌入式税控 POS 系统是一套智能且复杂的税控体系，很难再用单片机那样功能单一的处理器来实现。即使用单片机可以做到，那么不论在系统实现的成本和系统设计的复杂度上都要远远超过 32 位架构的嵌入式处理器实现的模式[9]。针对此现状，我们提出并设计了一种基于 32ARM 微处理器、拥有掉电保护装置、并采用嵌入式操作系统和图形化税控软件的税控收款机，极大提高了税控收款机在外设支持、存储容量、掉电保护和用户操作等方面的性能。本文采取了全新的设计理念，即以高安全性的 32 位架构的嵌入式处理器为核心，结合嵌入式 μCLinux 操作系统来完成。

1.3 本文研究内容及章节安排

本文主要阐述了基于 W90P710 处理器芯片及 μCLinux 操作系统的地税版税控收款机的软件部分的设计与实现。文章首先从软件的整体架构着手设计，再进一步探讨了系统软件和应用软件两部分的设计与实现。具体的章节内容安排情况如下所示：

第二章实现对税控机软件的总体设计，主要在功能需求分析的基础上确定了软件的设计方案。具体包括 GUI 图形界面、嵌入式数据库和应用软件的模块化设计。

第三章介绍了图形界面的解决方案，文章从图形界面的设计思想入手，介绍了图形界面的选择及详细移植过程，然后又举例讨论了图形界面的具体实现。

第四章介绍了嵌入式数据库的解决方案，文章从数据库的设计思想入手，介绍了嵌入式数据库的选择及详细移植过程，最后讨论了数据库的具体实现。

第五章主要讨论了整个软件的应用程序部分。本章首先介绍了应用软件的模块化设计和主要流程的具体实现，然后介绍了应用程序的移植过程。

第六章介绍了在两种不同的平台下调试程序的方法，以及在调试中遇到的问题及解决办法。

结束语作为论文的最后一部分对系统的软件整体进行了评述和总结，并对系统尚未完善的方面提出了若干建议，以及对税控收款机的发展方向作出了展望。

第 2 章 税控收款机软件整体设计

2.1 嵌入式软件开发流程

嵌入式系统是面向用户、面向产品、面向特定应用的专用计算机系统。它是软件与硬件的结合体，以应用为中心，对功能、可靠性、成本、体积、功耗等都具有严格要求[10]。在本次设计中，我与实验室同学一起完成了整个系统，我负责的工作主要是基于 μCLinux 操作系统之上的软件部分的设计。软件开发流程如图 2-1 所示：

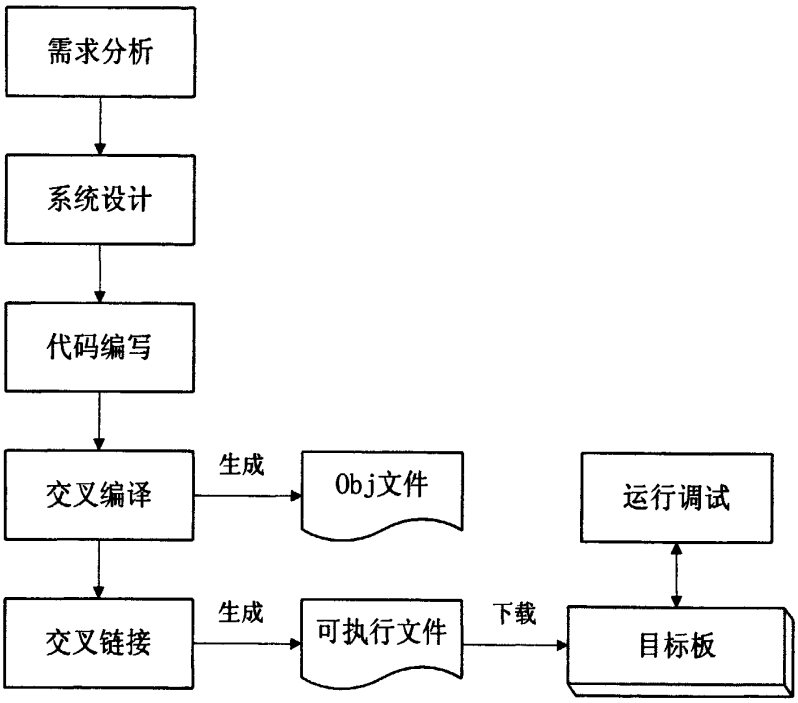

图 2-1 嵌入式软件开发流程

2.2 功能需求分析

本文设计的是一款地税版税控收款机，应用于服务业和餐饮业。主要功能如下：

✓ 商业收款机功能与税控功能的有机结合。能够进行权限管理、单品管理、报表管理等。与零售业税控机相比，地税版只有少量的单品管理；

✓ 友好的人机界面。税控收款机是面向对象设计的人机交互产品，图形化的人机界面是非常必要的，需要支持中文字体；

✓ 安全可靠的数据管理系统。数据是税控收款机的核心，一台合格的税控机必须能够安全可靠地管理数据。其中包括数据的安全生成、可靠存储 5-10 年的税控日交易数据，申报稽查数据的安全传输等；

✓ 方便快捷的信息查询。包括机器、用户、服务商及员工的所有信息；

✓ 税控功能。税控功能主要是通过 IC 卡实现的，带有微处理器的 IC 卡具有安全功能并增加了税控专用命令，IC 卡包括税控卡、用户卡、管理卡，通过税控卡正确生成税控数据，税控数据分别可靠存储在税控卡和税控存储器中；通过用户卡将税控数据安全传递到税务机关的税控收款机管理系统，税控机关通过管理卡检查税控收款机的交易情况[11]。具体功能包括税控机初始化、分发发票、安装发票、开票、申报、完税、稽查等；

✓ 异常处理功能。 掉电保护和税控 IC 卡操作异常处理。

除了上面提到的功能需求之外，对税控收款机还有一些共同的要求，如安全认证、防破坏性、平均无故障时间、使用寿命等技术标准。税控收款机既要满足这些公共的技术标准，又要满足用户使用的功能要求，才能同时被用户和税务机关所接受。

2.3 软件整体架构及设计方案

明确了税控收款机的需求后，就需要进行方案的论证和系统的基本架构。做为一个典型的嵌入式系统，系统的整体结构如图 2-2 所示[12]：

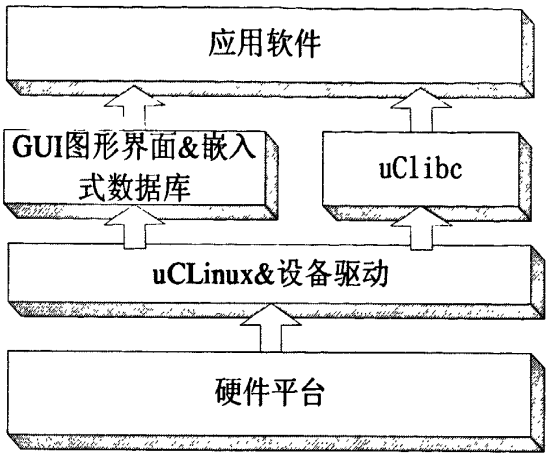

图 2-2 税控机体系结构图

本文所设计的就是基于操作系统和设备驱动之上的软件部分，包括 GUI 图形界面、嵌入式数据库和应用软件部分。

2.3.1 GUI 界面的设计方案

图形用户界面是当今计算机技术的重大成就之一，它极大地方便了非专业用户的使用，人们不再需要死记硬背大量的命令，而可以通过窗口、菜单方便地操作。[10]税控机界面是直接面向用户的，它的好坏直接影响到用户对它的感受。因此在设计中，我采用图形化界面（嵌入式 GUI）来实现。嵌入式 GUI 就是在嵌入式系统中为特定硬件设备或环境而设计的图形用户界面系统。所以这种 GUI 不但要具有普通图形界面的特征，而且在实际应用中，嵌入式系统对它还有如下的基本要求：

✓ 轻型、占用资源少；

✓ 高性能；

✓ 高可靠性；

✓ 可配置。

在嵌入式产品的开发过程中，软件开发人员通常采取如下几种方案来解决产品的图形需求[13]：

1. 编写针对特定图形输出设备的接口，自行开发图形相关的功能函数。比如一些图形功能简单的低端嵌入式产品就经常使用这种方案解决图形问题。然而，利用这种手段编写的程序，无法将显示逻辑和数据处理逻辑划分开来，从而导致程序结构不好，不便于调试，并导致大量的代码重复。这种方案的缺点很明显，即可移植性差，维护成本高。

2. 购买针对特定嵌入式操作系统的图形中间件软件包。一些嵌入式操作系统厂商，也为自己的操作系统专门开发了对应的图形用户界面（GUI）中间件产品。比如 μC/OS-II、Nucleus 上的 GRAFIX 包，VxWorks 上的 WindML 包等等。这种方案为嵌入式产品开发提供了直接可用的方案，并且能够和原有操作系统良好配合；但缺点是这类软件包的功能通常比较简单，且价格昂贵。另外，基于这些软件包开发的 GUI 应用软件不具备跨操作系统的可移植性。

3. 采用开放源码的嵌入式 GUI 支持系统。随着嵌入式 Linux 操作系统的应用，开源社区也在不断为嵌入式系统提供不同的开放源码嵌入式图形解决方案，比如 MicroWindows，OpenGUI 等开源软件。

这些开放源码的嵌入式 GUI 软件，为我们提供可行的解决方案。

4. 使用由独立软件开发商提供的嵌入式 GUI 产品，比如由北京飞漫软件技术有限公司开发的 MiniGUI 等。这种产品是开源（遵循 GNU 的 GPL 条款发布）的嵌入式 GUI 软件产品，但均采用双授权模式，即针对商业应用收取软件的许可费。Qt/Embeded 属于高端产品，只支持嵌入式 Linux 操作系统，需要 16MB 以上的静态存储空间及 64MB 以上的动态存储空间。MiniGUI 则可支持从中低端到高端的大多数嵌入式产品，其跨操作系统特性，以及适合嵌入式产品的小巧、高效的特点，使它受到了更多嵌入式产品开发商的青睐。

分析以上四种方案，第四种方案成本比较低，而且非常适合嵌入式系统的应用。因此，设计中采用第四种方案。

2.3.2 嵌入式数据库的设计方案

自几十年前出现的商业应用程序以来，数据库就成为软件应用程序的主要组成部分。正与数据库管理系统非常关键一样，它们也变得非常庞大，并占用了相当多的系统资源，增加了管理的复杂性。随着软件应用程序逐渐模块化，一种新型数据库会比大型复杂的传统数据库管理系统更适应。嵌入式数据库直接在应用程序进程中运行，提供了零配置（zero-configuration）运行模式，并且资源占用非常少[14]。

税控收款机的主要功能是实现对税务数据的管理和对商业数据的管理，其中税务数据的管理有比较成熟的国标规定，而商业数据的管理却存在着定义不清晰，需求个性化的复杂状况，解决数据管理的最合适方法是使用数据库管理系统，这样将有效地提高数据管理部分的开发、设计、个性化及可靠性。从多次市场反馈信息分析，用户对数据管理的需求是使用商业收款机的最主要原因，而税控数据管理是国家主管部门的强制性要求，并且各地税务管理部门对税控数据的管理除了按国家要求以外，同样存在对税控数据管理的附加要求，因此，移植一个小型嵌入式数据库管理系统子系统非常迫切也十分必要，该部分工作将着重于以下几个要求：

1. 能够将各种数据有序管理起来，并对其他应用程序提供统一的接口和服务。管理数据包含以下几个部分：
   - ✓税务数据集；
   - ✓商业数据；
   - ✓银行卡接口数据集（预留）；
2. 能够保存 5-10 年的税务日交易数据和至少 20000 条发票明细数据；
3. 能够完成税控机管理、商品管理、员工管理及报表管理等一系列功能。

2.3.3 应用软件的设计方案

应用软件的设计是以一定的方法为基础的，对于税控收款机这样一种相对复杂的软件开发任务，设计中根据软件设计的模型，从用户需求和系统要实现的任务功能出发，主要遵循了以下原则：

1. 易用性。提供的应用编程接口要尽可能简洁而又满足需要，函数要有良好的容错性，便于将来升级为银税机开发时，其他开发者能很容易的掌握和运用；
2. 高性能。要在保证终端设备能够正常运行的情况下，考虑到对存储器(包括 SDRAM 和 FLASH)的严格要求，要尽量减少可执行代码所需的空间，提高程序的运行速度。这与应用程序有关，更与 API 函数代码的优化有关；
3. 模块化。把整个软件划分为较小的模块。为了减少模块与模块之间的关联性，设计中各个模块之间的逻辑结构相对独立，无函数的交叉调用，数据传递由全局变量完成。这种模块化设计使得各个子系统之间相对独立，更加便于系统的调试，提高了系统的稳定性，同时也为软件移植和系统升级大大提供了方便；
4. 协同开发。软件以及软硬件之间采用协同开发模式；
5. 可移植性。嵌入式操作系统和开发出来的应用程序要具有良好跨平台性，要能支持主流的微处理器硬件平台。便于将来的升级优化；
6. 安全性高。考虑到各种异常情况，具有完备的掉电保护功能，确保数据正确。

整个软件共分为三大模块，系统主控模块、商业管理模块和税控管理模块。图 2-3 是整个软件的模块框：

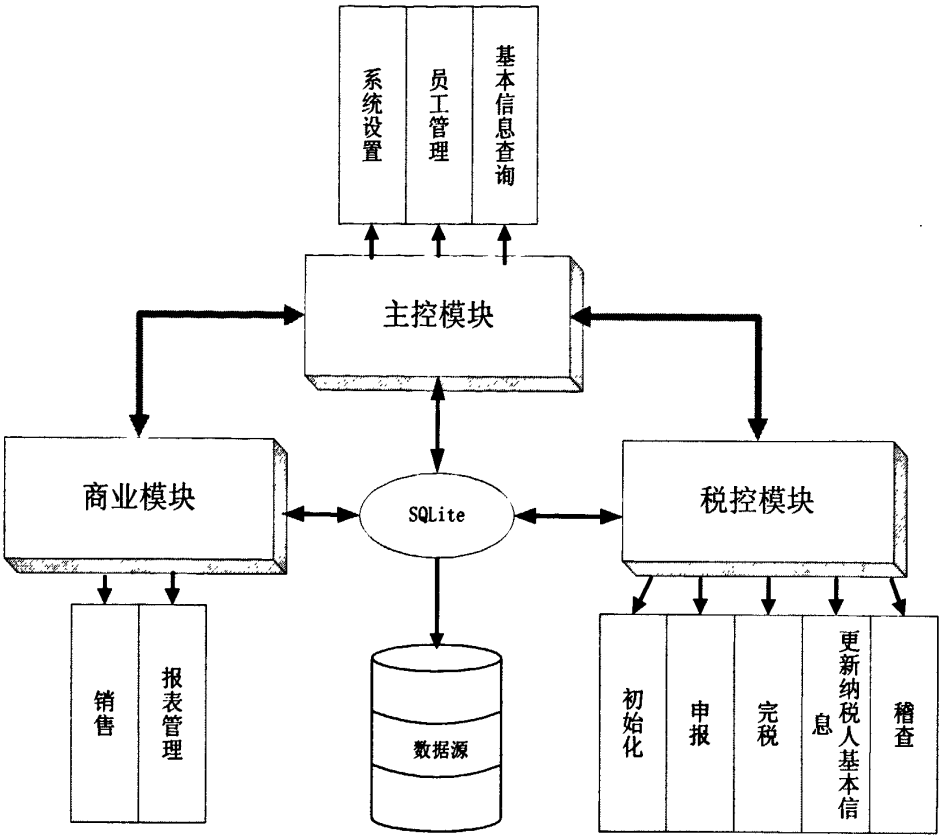

图 2-3 软件整体模块图

主控模块控制着机器的信息，包括系统设置，员工管理和基本信息查询功能，它与商业管理模块和税控管理模块联系比较密切，而商业和税务两个模块相对独立，主要通过机器初始化和开票操作关联起来。在模块化设计的同时，每个模块还采用了分层设计的方式，主要分为界面，数据库和基本流程三层，分别调用不同的 API 接口，结构非常清晰，为后续代码的跨平台移植奠定了基础。

2.4 本章小结

本章从一个全局的角度考虑了税控收款机的总体设计方案。包括图形界面，嵌入式数据库和应用程序三部分的设计原则及设计思想。经过本章的介绍，本课题所要研究的内容和重点其实已经是一目了然。此外，整章内容着眼于系统的架构设计，为三、四、五章的后续介绍做了很好的准备，起到一个提纲挈领的作用。

第 3 章 GUI 图形界面的设计与实现

3.1 税控收款机图形界面的设计

3.1.1 界面的设计思想

对于税控收款机来说，用户第一印象就来自于界面。因此在界面的设计中要严格从需求出发，主要掌握以下原则：

1. 用户原则。税控收款机主要用于员工的操作，要完全从用户的思维方式出发。因此界面按照分层菜单式的设计原则根据总体模块设计菜单；

2. 操作方便。菜单设计要求操作方便快捷，这里采用的方式是每个界面不超过八个菜单项。对于操作过于频繁的销售界面可以通过选择菜单，也可以直接利用快捷键；

3. 权限管理。由于 SQLite 的数据库权限只依赖于文件系统，没有用户帐户的概念。所以在这里用户权限主要由界面部分实现；

4. 显示充分。本次设计中采用 240*128 的大液晶屏幕，对于多条信息显示的界面来说，可以充分显示大屏的优势，有效编排界面信息，减少翻页；

5. 错误提示。由于税控流程操作相对复杂，因此在税务操作的过程中一定要提供完善的错误提示界面，对用户的非法操作进行严格限制，给出详细的出错信息并指导用户进行更改。

3.1.2 界面的菜单设计

税控机界面的设计采用菜单式风格，系统分为两级菜单，每个菜单不超过八个选项，用户可以通过反显按确定键或数字键选择自己需要的操作。菜单整体示意图如图 3-1 所示：

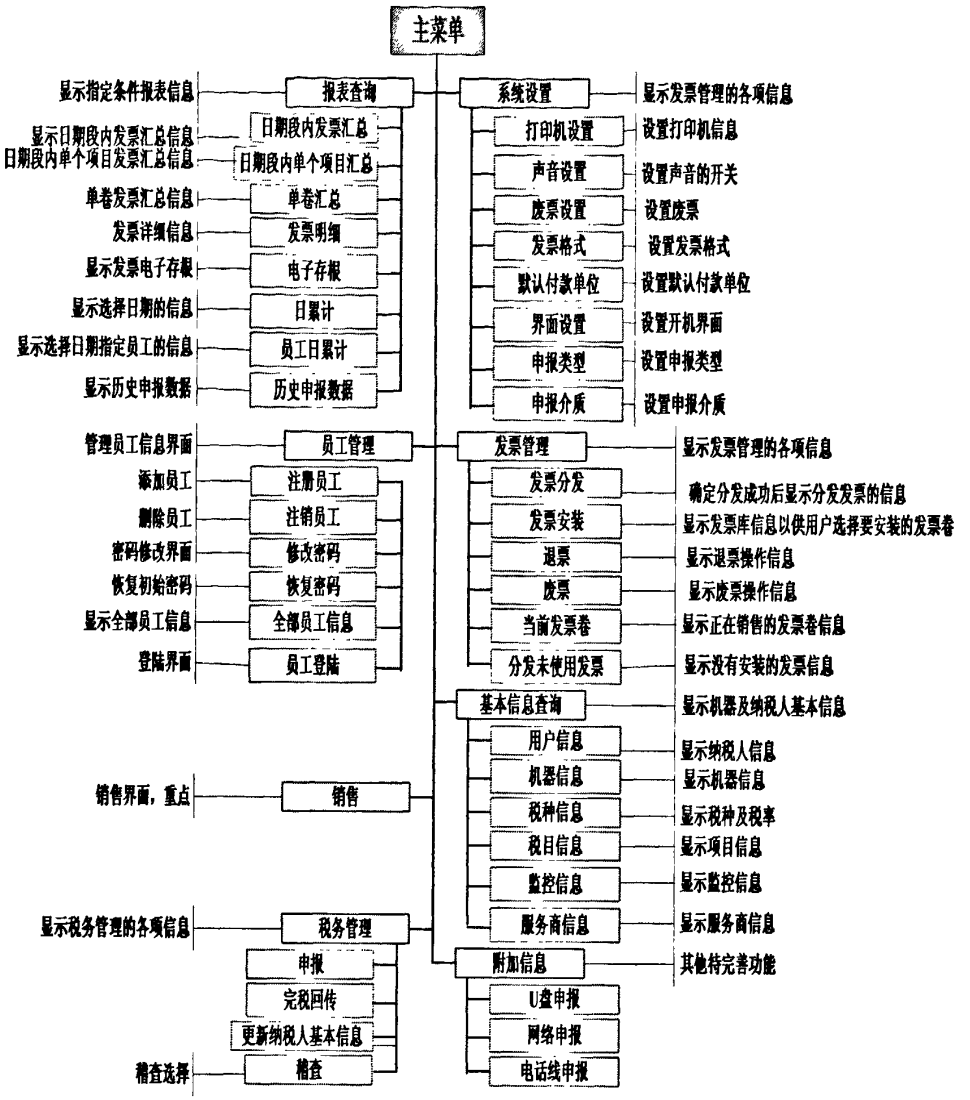

图 3-1 界面菜单整体示意图

机器的用户有两级权限：管理员和收款员。管理员拥有机器的全部操作权限，收款员不能进行系统设置和税务管理，同时不能注册注销员工。员工登陆后，将该员工的权限做为全局变量保存在内存中，当操作到有限制的界面时首先要进行权限判断。

3.2 图形界面的介绍与选用

由于在实时嵌入式操作系统中，硬件环境比较苛刻，因此要求运行其中的图形界面尽可能的精简，而传统的窗口系统尚不能满足实时嵌入式系统的需求。所以，在基于 Linux 的实时嵌入式系统上，设计一个能够充分满足嵌入式系统需求的图形用户界面就成了当务之急[10]。

3.2.1 目前 Linux 下图形界面简介

国内外已有许多专门针对 Linux 的嵌入式 GUI 系统，然而，由于开发人员对实时性嵌入式系统在理解上的不同，使得这些 GUI 系统在接口定义、体系结构、功能特性等方面存在着很大的差别。另外，这些 GUI 系统所使用的授权方式也不同。目前比较流行的就是以下所介绍的几种[13]：

MicroWindows：开源项目，该项目的特色在于提供了类似 X 的客户/服务器体系结构，并提供了相对完善的图形功能，包括一些高级功能，但是 MicroWindows 无任何硬件加速能力，图形引擎中存在许多低效算法，代码质量也比较差，只支持 Linux。

OpenGUI：由于其基于汇编实现的内核并利用 MMX 指令进行了优化，所以 OpenGUI 运行速度非

常快。但是 OpenGUI 库是采用 C++编写的，只提供 C++接口。而且目前只支持 x86 平台，比较适合于基于 x86 平台的实时系统，跨平台的可移植性稍差。这种 GUI 目前发展比较缓慢。支持 DOS、Linux、QNS 几种操作系统。

Qt/Embedded：因为 Qt 是 KDE 等项目使用的 GUI 支持库，因此基于 Qt 的 X Window 程序可以非常方便地移植到 Qt/Embedded 版本上。这是许多嵌入式设备厂商采用 Qt/Embedded 开发嵌入式产品的主要原因。但是 Qt/Embedded 只支持 Linux 操作系统。此外，还有一些问题值得注意：

- ✓ Qt/Embedded 是一个 C++函数库，程序效率低，资源消耗大，对硬件提出了更高的要求；
- ✓ Qt/Embedded 库目前主要针对手持式信息终端，缺乏硬件加速支持，很难应用到对图形速度、功能和效率要求较高的实时性嵌入式系统当中；
- ✓ Qt/Embedded 提供的控件集风格沿用了 PC 风格，并不太适合许多手持设备的操作要求；
- ✓ Qt/Embedded 结构过于复杂，很难进行系统裁剪、扩充、定制和移植；

MiniGUI：为实时嵌入式操作系统提供了非常完善的图形及图形用户界面支持。MiniGUI 本身的可移植性设计，使得不论在哪个平台、哪种操作系统上运行，MiniGUI 均能为上层应用程序提供一致的应用程序编程接口（API），支持 Linux/μCLinux 等操作系统。通过分析，几种嵌入式图形界面的综合比较如表 3.1：

表 3.1 几种嵌入式图形界面的综合比较[13]

	MiniGUI	MicroWindows	OpenGUI	QT/Embedded
API 风格	Win32 风格	X、Win32 子集	私有	QT（C++）
API 是否完备	是	Win32 支持尚不完备	是	是
函数库的典型大小	700KB	600KB	300KB	3MB
内嵌资源方式	有	无	无	无
可移植性	很好	很好	只支持 x86 平台	较好(但函数库本身的跨平台交叉编译很困难)
授权条款	GPL / 商业许可证	MPL/LGPL	LGPL	QPL/GPL/商业许可证
多进程支持	优秀	一般	不好,无多任务支持	优秀
健壮性/稳定性	好	很差	好	好
多语种支持	独特的多字符集支持功能，更加适合嵌入式系统	一般	一般	采用 UNICODE 编码，但效率低
字体支持	多种点阵，矢量字体格式	点阵、矢量字体	点阵	只 QPF 字体
可配置和可定制性	好（提供了大量编译配置选项，可配置能力很强）	一般	差	差
系统资源	小	较大（基于	最小(不支持	最大(用 C++

消耗	（MiniGUI-Threads 和 MiniGUI-Processes 均针对最小系统资源消耗设计）	UNIX 套接字的传统客户/服务器体系，进程间的通讯频繁，系统资源消耗较大）	多进程，资源消耗最小）	实现，系统资源消耗最大）
效率	好	较差	最好（底层用汇编编写）	较差
操作系统支持	Linux/μClinux、eCos、μC/OS-II、VxWorks 等	Linux	DOS、Linux、QNX	Linux
已知能运行的硬件平台	x86、ARM、MIPS、PowerPC、StrongARM、主频最低需 30Hz	x86、ARM、MIPS、StrongARM、主频最低需 70Hz	x86	x86、ARM、StrongARM、主频最低需 100Hz

由表中分析可以得出，对于我们基于 μClinux 的嵌入式税控机来说，MiniGUI 不管在效率、资源消耗以及开发难度方面都更胜一筹，因此，最终我们选择了 MiniGUI 做为开发工具。

3.2.2 MiniGUI 性能描述

一、MiniGUI 的功能特征

MiniGUI 是一个根据嵌入式系统应用特点量身定做的一个完整图形支持系统。作为操作系统和应用程序之间的中间件，MiniGUI 将底层操作系统及硬件平台差别隐藏了起来，并对上层应用程序提供了一致的功能特性，这些功能特性主要包括[15]：

1. 跨操作系统支持。具体包括普通嵌入式 Linux、μClinux、eCos、μC/OS-II、VxWorks、pSOS、ThreadX、Nucleus 等，同时还提供 Win32 平台上的 SDK 开发包，方便嵌入式应用程序的开发和调试；
2. 为了适应不同的操作系统运行环境，我们可将 MIniGUI 配置成三种运行模式：MiniGUI-Threads、MiniGUI-Processes 及 MiniGUI-Standalones；
3. 完整的多窗口机制和消息传递机制；
4. 提供常用的控件类。包括静态文本框、按钮、单行和多行编辑框、列表框、组合框、进度条、属性页、工具栏等；
5. 通过两种不同的内部软件结构支持低端显示设备（比如单色 LCD）和高端显示设备（比如彩色显示器）。前者小巧灵活，而后者在前者的基础上提供了更加强大的图形功能；
6. 对话框和消息框支持以及其它 GUI 元素。包括菜单、加速键、插入符、定时器等；
7. 简体中文（GB2312）输入法支持。包括内码、全拼、智能拼音等。用户还可以从飞漫软件获得五笔、自然码等输入法支持。

其次，在 MiniGUI 近七年的发展过程中，有许多值得一提的技术创新点，包括：

1. 图形和输入抽象层。图形和输入抽象层对顶层 API 基本没有影响，但大大方便了 MiniGUI 自身及应用程序的移植、调试等工作；
2. 多字体和多字符集支持。支持 ISO8859-1～ISO8859-15、GB2312、GBK、GB18030、BIG5、EUCKR、EUCJP、Shift-JIS、UNICODE 等字符集，非 UNICODE 内码实现更加适合嵌入式系统，支持点阵字体(包括 QPF),TrueType 以及 Adobe Type1 等矢量字体；
3. 针对不同操作系统特点的运行模式。和 Linux 这样的类 UNIX 操作系统相比，一般意义上的传统嵌入式操作系统具有一些特殊性，诸如 μClinux、μC/OS-II、eCos、VxWorks 等操作系统，通常没有

MMU（内存管理单元，用于提供虚拟内存支持）的CPU上；这时，往往就没有进程的概念，而只有线程或者任务的概念，这样 GUI 系统的运行环境也就大相径庭。因此，为了适合不同的操作系统，我们可将 MiniGUI 配置成 MiniGUI-Threads、MiniGUI-Processes 及 MiniGUI-Standalone 等三种运行模式。

二、基于 MiniGUI 的嵌入式系统软件架构

MiniGUI 具有良好的软件架构，通过抽象层将 MiniGUI 上层和底层操作系统隔离开来。如图 3-2 所示，基于 MiniGUI 的应用程序一般通过 ANSI C 库以及 MiniGUI 自身提供的 API 来实现自己的功能。MiniGUI 中的“可移植层”可将特定操作系统及底层硬件的细节隐藏起来，而上层应用程序则无需关心底层的硬件平台输出和输入设备[13]。

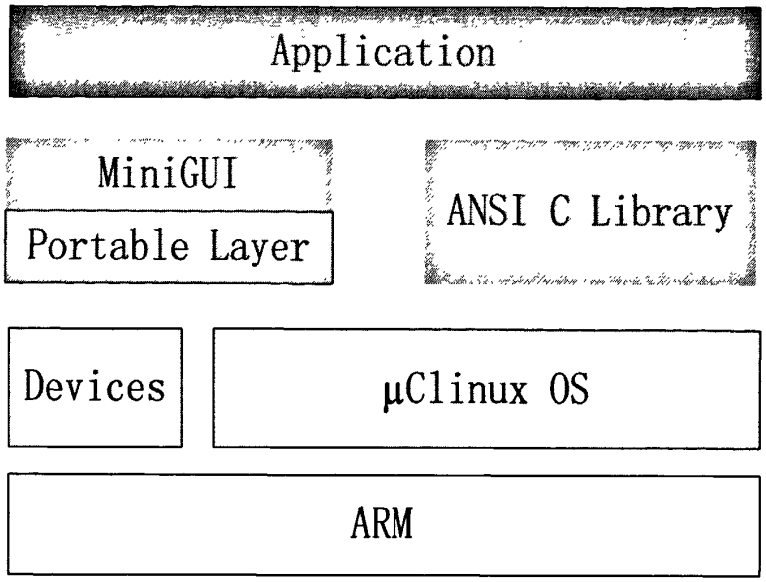

图 3-2 基于 MiniGUI 的嵌入式系统软件结构

三、MiniGUI 的三种运行模式及选择

为了适应不同的操作系统环境，MiniGUI可以配置为三种运行模式，分别为MiniGUI-Threads 模式，MiniGUI-Processes 模式和MiniGUI-Standalone 模式。但无论采用哪种运行方式，MiniGUI都为上层应用软件提供了最大程度的一致性，只有少数几个涉及初始化的接口在不同运行模式上有所不同[16]：

✓ MiniGUI-Threads。运行在 MiniGUI-Threads 上的程序可以在不同的线程中建立多个窗口，但所有的窗口在一个进程或者地址空间中运行。这种运行模式非常适合于大多数传统意义上的嵌入式操作系统，比如 μC/OS-II、eCos、VxWorks、pSOS 等等。当然，在Linux 和 μClinux 上，MiniGUI 也能以 MiniGUI-Threads 的模式运行；

✓ MiniGUI-Processes。和 MiniGUI-Threads 相反，MiniGUI-Processes 上的每个程序是独立的进程，每个进程也可以建立多个窗口。MiniGUI-Processes 适合于具有完整 UNIX特性的嵌入式操作系统，比如嵌入式 Linux 和VxWorks 6；

✓ MiniGUI-Standalone。这种运行模式下，MiniGUI 可以以独立进程的方式运行，既不需要多线程也不需要多进程的支持，这种运行模式适合功能单一的应用场合。比如在一些使用μClinux的嵌入式产品中，因为各种原因而缺少线程库支持，这时，就可以使用MiniGUI-Standalone来开发应用软件。

μClinux运行在没有 MMU（内存管理单元，用于提供虚拟内存支持）的CPU上，这时，往往就没有进程的概念，而只有线程或者任务的概念[17]，这样，GUI 系统的运行环境也就大相径庭。因此，针对税控收款机的需求，并不需要窗口的多线程，可以将MiniGUI配置在MiniGUI-Standalone模式下。

3.3 MiniGUI 的移植

首先要在飞漫官方网站上下载 MiniGUI 的源码包，MiniGUI-STR for μClinux 版本一般有 4 个压缩包，文件名及其说明如下：

- ✓ libminigui-1.3.1.tar.gz：MiniGUI 函数库源代码；
- ✓ minigui-res-1.3.1.tar.gz：MiniGUI 所使用的资源，包括基本字体、图标、位图和鼠标光标；
- ✓ mde-1.3.1.tar.gz：MiniGUI 的综合演示程序；
- ✓ mg-smaples-1.3.1.tar.gz：MiniGUI 的示例程序。

3.3.1 MiniGUI 的整体分析

从整体结构上看，MiniGUI 是分层设计的，层次结构如图 3-3 所示，在最底层，GAL 和 IAL 提供底层图形接口以及鼠标和键盘的驱动；中间层是 MiniGUI 的核心层，包括窗口系统必不可少的各个模块；最顶层是 API，即编程接口。

GAL 和 IAL 为 MiniGUI 提供了底层的 Linux 控制台或者 X Window 上的图形接口以及输入接口，而 Pthread 用于提供内核级线程支持的 C 函数库。对于 MiniGUI 的移植来说，应用程序接口都是通用的，需要设计的主要是 GAL 和 IAL 和核心层的配置。利用 GAL 和 IAL，大大提高了 MiniGUI 的可移植性，并且使程序的开发和调试变得更加容易。可以在 X Window 上开发和调试自己的 MiniGUI 程序，通过重新编译就可以让 MiniGUI 应用程序运行在特殊的嵌入式硬件平台上[18]。

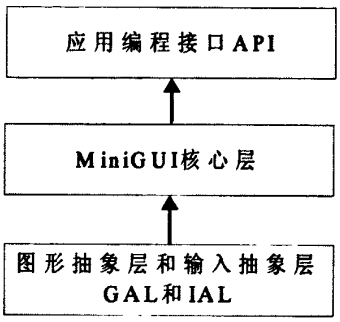

图 3-3 MiniGUI 的整体架构

3.3.2 在 PC 机上安装 MiniGUI

由于基于 MiniGUI 的应用程序都在 ARM 开发板上运行，如果每次修改程序都要下到开发板上看效果的话是非常不方便的，因此在 PC 机上安装 MiniGUI 可以方便随时查看界面的效果，待满意后再下载到开发板中运行，这样节省了大量的时间。在 PC 机上安装 MiniGUI 的步骤如下：

1. 安装 Linux 操作系统；

2. 安装 Red Hat 9 Linux 操作系统，并且配置好 samba 服务器，对 Linux 下的所有操作都可以在服务器上进行。操作系统内核包含了 FrameBuffer 支持，并包含了 VESA FrameBuffer 驱动程序，因此可以进行 FrameBuffer 配置。首先修改/boot/grub/menu.lst 文件，并在 kernel 打头的一行添加 vga = 0x0317（根据分辨率不同此值不同）（见图 3-4），然后重新启动 Linux 操作系统，即已经激活了 VESA FrameBuffer；

```
default=0
timeout=10
splashimage=(hd0,2)/boot/grub/splash.xpm.gz
title Red Hat Linux (2.4.20-8,FrameBuffer)
        root (hd0,2)
        kernel /boot/vmlinuz-2.4.20-8 ro root=LABEL=/ vga=0x0317
        initrd /boot/initrd-2.4.20-8.img
title DOS
        rootnoverify (hd0,0)
        chainloader +1
~
~
~
                                                              1,1           All
```

图 3-4 FrameBuffer 的配置

3. 解压 MiniGUI 源代码及资源包；

4. 下载的源码包都是以.tar.gz 后缀结尾，解压命令为 tar zxvf ****.tar.gz；

5. 安装资源文件；

6. 进入到解压后的 minigui-res-1.3.3 目录，然后以 root 身份运行 make install 命令；

7. 配置和编译 MiniGUI；

8. 进入到解压后的 libminigui-1.3.3.tar.gz，运行.configure 命令配置 minigui，然后以超级用户身份运行 make 及 make install 命令编译并安装 minigui，这样，minigui 的函数库就安装在/usr/local/lib 目录下。修改/etc/ld.so.conf 文件，将/usr/local/lib 目录添加到该文件最后一行；

9. 运行 ldconfig 命令更新共享库。

这样，minigui 就已经安装在 PC 机上了，编译运行 helloworld.c，就可以看到漂亮的 minigui 界面了（见图 3-5）。

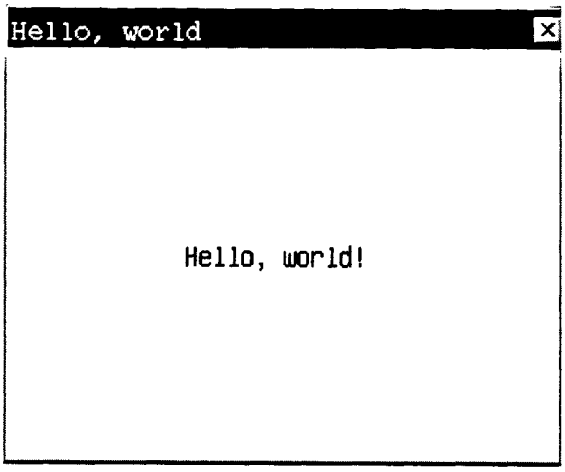

图 3-5 运行 MiniGUI 应用程序界面

3.3.3 GAL 和 IAL 移植

MiniGUI 移植的核心是GAL和IAL的移植。从MiniGUIv0.3.xx版本开始，MiniGUI引入了图形抽象层和输入抽象层（即GAL和IAL）的概念。它定义了一组不依赖于任何特殊硬件的抽象接口，所有顶层的图形和输入处理都建立在抽象接口之上。而用于实现这一抽象接口的底层代码为“图形引擎”和“输入引擎”，类似操作系统中的驱动程序。对目标系统中不相同的显示设备和输入设备，只需要根据抽象层接口实现新的图形引擎和输入引擎即可。GAL和IAL提供底层图形接口以及鼠标和键盘的驱动,它们对顶层API没有任何影响，但大大方便了MiniGUI自身以及应用程序的移植和调试工作[19]。

一、GAL移植

由于我们开发板操作系统内核中已经移植了 FrameBuff 驱动，所以我们不需要编写图形引擎，只需要将内核中的 fbcon.c 文件进行适当的配置即可。配置 FrameBuffer 时主要修改了设备编号，将我们开发板上没有的设备编号初始化为 0。然后将 MiniGUI 选择 Native 的 GAL 引擎，便可以使用 FrameBuff 作为 MiniGUI 的图形发生引擎。其中 MiniGUI 的 GAL 层源码文件如表 3.2：

表 3.2 GAL 层源码说明

文件	说明
src/include/gal.h	根据NEWGAL定制与否选择相应GAL支持的头文件（如下）
src/include/newgal.h	NEWGAL支持的头文件，声明NEWGAL接口
src/include/oldgal.h	一般GAL支持的头文件，声明OLDGAL接口
src/gal/gal.c	初始化GFX，根据GFX数组和配置文件

	所指定的GAL初始化cur_gfx的GAL接口
src/gal/libggi.h&libggi.c	LibGGI实现的GAL
src/gal/svgalib.h&svgalib.c	SVGALib实现的GAL
src/gal/vga16.h&vgal16.c	VGA16色模式下的GAL（需SVGALib支持）
src/gal/native/native.h&native.c	Native GAL的GFX初始化实现等，依赖于具体的设备显示能力，调用src_fb.c中打开fb设备时，调用具体的fb.h中指定的子设备psd驱动模块
src/gal/src_fb.c	打开fb设备，调用fb.h中接口选择子设备
src/gal/fb.h&fb,c	子设备选择操作接口
src/gal/fblin1.c…	具体的子设备psd操作接口

二、IAL的移植

MiniGUI的IAL层将输入设备的输入事件最终映射为GUI系统API层的消息事件。IAL层默认处理两种设备的输入操作：键盘设备和鼠标设备。键盘设备向上层提供不同的按键输入信息，鼠标设备提供点击、抬起和落笔坐标等的信息。在实现MiniGUI与输入设备驱动的接口时，采用select的方式获得输入设备的动作，并转换为消息队列中的消息。消息参数按照Win32接口定义为点击键编号或鼠标当前的坐标(其中触摸屏事件与鼠标事件类似)。通过编写针对硬件开发系统的IAL支持代码，实现了IAL层的移植。其中MiniGUI的IAL层代码说明如表3.3所列：

表3.3 IAL层接口说明

接口函数	说明
static int mouse_update(void)	更新鼠标状态
static void mouse_getxy(int *x,int*y)	获取鼠标当前所在位置
static int mouse_getbutton(void)	获取鼠标当前所按下按钮编号
static int keyboard_update(void)	更新键盘状态
static const char*keyboard_getstate(void)	获取键盘状态
static int wait_enevt(int which,fd_set *in,fd_set*out,fd_set *except,struct timeval *timeout)	事件等待。该函数采用select方式监测键盘和鼠标设备，并从相关事件的设备中读取数据
BOOL InitelPAQInput(INPUT *input,const char *mdev,const char *mtype) void TermIPAQInput(void)	初始化IAL设备 关闭IAL设备

MiniGUI通过INPUT数据结构来表示输入引擎，此数据结构的定义是在libminigui-str-1.3.1/src/include/ial.h文件中，具体代码参考源文件。INPUT数据结构定义了很多函数指针，就是用来指向设定的IAL的对应函数，从而实现硬件的输入。

在此数据结构中，init_input和term_input两个函数指针是IAL的可见接口，用户IAL程序通用接口只需要提供这两个函数。本系统中的输入设备只有一个63键的键盘，为了实现键盘的驱动，内核中必须相应的驱动程序，这是实现MiniGUI的IAL移植的前提。当然这里的驱动程序可以是最底层的实现，更为复杂的通过MiniGUI的IAL驱动扩展实现。由于μClinux内核已经实现了键盘设备的驱动，这里，我们主要做了以下几步完成了IAL移植：

1. 在 libminigui-str-1.3.1的src/ial/目录下建立wbial.c，wbinput.c和wbial.h文件，编写高层IAL驱动程序。这里我参照周立功SmartARM2200开发板的IAL驱动主要做了以下修改：
   - ✓ 按照税控机按键的定义将按键定义数组修改为char EVENT_CHG_TAB[63] ；
   - ✓ 将相应文件中有关鼠标操作的函数屏蔽；
   - ✓ 修改键盘状态缓冲区。

2. 将新的IAL驱动程序编译到MiniGUI的库中，以便为应用程序提供新的IAL支持。为了实现这一步，我修改了libminigui-str-1.3.1/src/ial目录下的Makefile文件，在此文件中加入了一个新的变量WB_SRCS = wbial.c wbinput.c wbial.h， 然后在Make文件中加入了新变量的引用，这样保证了在编译MiniGUI时能够编译移植的IAL源文件。修改部分见黑体：

```
**************************Makefile************************
...
WB_SRCS = comminput.c comminput.h comm_drive.c
libial_la_SOURCES = $(COMMON_SRCS) $(ARM3000_SRCS) $(IPAQ_SRCS) \
                    $(L7200_SRCS)$(DUMMY_SRCS)\
                        $(QVFB_SRCS) \$(PX255B_SRCS) \
                        $(MC68X328_SRCS) $(SMDK2410_SRCS) \
                        $(UCB1X00_SRCS) $(AUTO_SRCS) \
                    $(HH2410R3_SRCS)$(EMBEST2410_SRCS)\
                    $(HH5249KBDIR_SRCS)\
                    $(FFT7202_SRCS)$(FXRM9200_SRCS) \
                        $(SKYEYE_EP7312_SRCS) \
                    $(WB_SRCS)
```

3. 为了使MiniGUI能够识别此输入引擎，需要注册输入引擎。实际上就是初始化一个INPUT的结构单元，即id（引擎名称）、init_input（输入引擎的初始化函数）和term_input（输入引擎的终止清除函数）三个成员。具体就是在libminigui-str-1.3.1/src/ial目录下的ial.c程序中加入输入引擎的注册代码，引擎名为“wbial”，所加的程序为：

```
#ifdef_WB_IAL
  #include “wbial.h”
#ifdef _WB_IAL
    {“wbial”,InitWBInput, TermWBInput }
```

3.3.4 MiniGUI 的交叉编译

另建一个目录同样将四个压缩包解开，然后可以在此目录中进行交叉编译。

1. 搭建交叉编译环境

我们这里使用的交叉编译工具是华邦开发板自带的交叉编译工具链 arm_tools_3.3.tar.gz，将该交叉编译链安装在/usr/local/arm_toos 目录下。

2. MiniGUI 库的编译安装

针对 W90P710 开发板，我们需要专门编写一个配置脚本 buildlib-wb710。在这里，我主要是对源码中自带的 buildlib-coldfire 文件做了修改，整个文件主要分为两部分，清单如下：

```
**********************buildlib-wb710 脚本内容*************************
                    rm config.cache config.status -f
                    CC=arm-elf-gcc \
                CFLAGS="-O2 -g -D__linux__ -D__uClinux__ \
                    -I/usr/local/arm_tools/arm-elf/inc \
            -I/home/sharon/biye/W90P710_PR/uClinux-dist/linux-2.4.x/include \
            -I/home/sharon/biye/W90P710_PR/uClinux-dist/lib/include -fno-builtin  " \
                    LDFLAGS="-Wall -elf2flt -static
                -L/home/sharon/biye/W90P710_PR/uClinux-dist/lib/lib \
                -L/home/sharon/biye/W90P710_PR/uClinux-dist/linux-2.4.x/lib -lc" \
```

表 3.4 配置文件参数说明(a)

参数	意义
CC	指明所使用的交叉编译器为 arm-elf-gcc
CFLAGS	设置编译器的一些参数
LDFLAGS	设置连接器的一些参数
elf2flt	表示指定将生成的 ELF 格式的可执行文件转换成 FLAT 格式的文件
Wall	表示显示编译过程中的所有警告
static	表示使用静态方式连接库

```
./configure \
    --prefix=/home/sharon/biye/minigui \
--host=arm-elf-linux \
--build= i686-linux \
    --target=arm-elf-linux \
    --disable-shared \
    --with-osname=uclinux \
    --disable-thread \
    --enable-lite \
    --enable-standalone \
    --disable-micemoveable \
    --disable-cursor \
    --disable-newgal \
    --enable-nativegal \
    --enable-galfbcon \
    --disable-galqvfb \
    --disable-galecoslcd \
    --enable-fblin11 \
    --disable-fblin8 \
    -- disable-fblin16 \
    --disable-fblin32 \
    --disable-textmode \
    --disable-videoqvfb \
    --disable-videoecoslcd \
    --disable-nativeial \
    --disable-qvfbial \
    --disable-incoreres \
    --enable-gbsupport \
    --enable-imegb2312 \
    --disable-aboutdlg \
    --disable-savescreen
```

第二部分主要是指定 MiniGUI 各项功能的开启和关闭，详细解释可以参考《MiniGUI 用户手册》，这里只对重要的功能做说明：

表 3.5 配置文件参数说明(b)

prefix=/home/sharon/biye/minigui	指定 minigui 的安装路径

target=arm-elf-linux	目标平台类型
build= i686-linux	执行编译的环境
host= arm-elf-linux	主机（编译器）类型
with-osname=uclinux	指定操作系统为 μCLinux
disable-newgal	液晶屏为单色，只能使用旧的 GAL
enable-galfbcon	开启内部 FrameBuffer GAL
enable-fblin1l	液晶屏为单色屏
disable-incoreres	关闭内部资源编译的方式
enable-imegb2312	包含 GB2312 中文简体输入法

注：

✓ 由于我们采用的液晶屏为单色黑白屏，所以选用旧的GAL。MiniGUI提供的GAL有两种，而NEWGAL只能支持至少8BPP的液晶屏；

✓ 使用MiniGUI内嵌资源不能对其库文件进行裁剪，会浪费许多不必要的资源，因此这里不采用内嵌式资源。

这样，MiniGUI 库的交叉编译配置文件就完成了，然后依次进行以下步骤即可完成静态库的交叉编译：

- ✓ ./ buildlib- wb710：生成一个 Makefile 文件；
- ✓ Make：交叉编译；
- ✓ make install：库安装；
- ✓ ldconfig：更新共享库。

3. MiniGUI 应用程序的交叉编译

应用程序的交叉编译有两种方式，第一种是将写好的应用程序添加到 uClinux-dist 目录下，在 uClinux-dist 中直接编译和链接，直至最后生成映象文件；第二种方法是单独编译 MiniGUI 的应用程序，生成可执行程序[20]。显然第一种方式要每次编译内核很浪费时间，我在这里采用了后者。

单独编译应用程序的关键在于编写正确的 Makefile，在本课题中，我设计了一个完整的应用程序的 Makefile 文件，能够实现交叉编译，具体介绍见《5.3.2 MakeFile 文件的设计》。

4. 裁剪配置文件和资源文件

在本设计中我们没有使用内嵌式资源，所以除了可执行的应用程序还要将配置文件 MiniGUI.cfg 和需要的资源文件都手动下载到开发板上。当然这里需要对配置文件和资源文件做一定的修改和裁剪，使得能够做到最优化，最节省资源。这里对配置文件所做的主要修改有以下几点：

✓ 修改system段中的IAL输入引擎，令ial_engine = wbial；

✓ 修改配置文件中指定的资源文件的路径，与自己的资源存放路径相符；

✓ 字体资源是影响MiniGUI存储空间最重要一部分，所以对字体文件中不需要的部分就可以删除掉[21]。首先根据自己目标环境的需要确定需要那些字体，我们这里采用16点阵等宽字体，然后就可以将MiniGUI.cfg中rawbitmapfonts、varbitmapfonts、qpf、truetypefonts、type1fonts段中不需要的项删除，同时将font_number修改成需要的字体数量；然后将资源文件中不需要的字体文件删除，就可以将这两部分下载到开发板中对应的目录下了；

✓ 修改输入法段，将配置文件及对应资源文件中不需要的五笔、双拼等输入法删除。

通过裁剪，得到的 MiniGUI 资源文件大概需要 700K，应用程序大概为 600K 多一点，最终占用的总空间可以控制在 2M 以内。

这样，就可以在开发板上看到程序运行的效果了。

综上所述，MiniGUI 的移植过程可通过图 3-6 来描述：

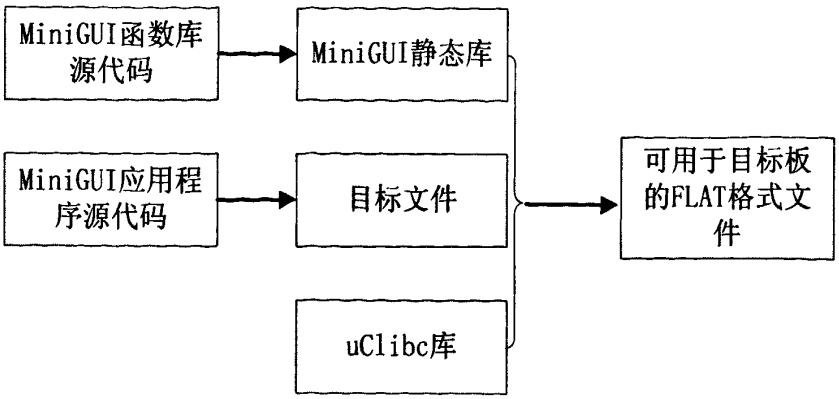

图 3-6 MiniGUI 移植图示

3.3.5 移植中遇到的问题

一、字库的移植

随着税控收款机的不断发展，国标对界面汉字库的要求也越来越高。所以MiniGUI1.3中自带的GB2312已经不能满足要求了。需要移植包含汉字更加丰富的GB18030。移植过程中需要将GB18030的二进制文件拷贝到资源目录下，包括Asi.bin、GB3s1516.hz、 GB3s1516.zf、 Nm、 bin、Py.bin几个文件，因为GB2312和GB18030的查找方式基本相同，只要在MiniGUI中把源代码中的首地址改成一样的即可使用。

二、FrameBuffer配置

在PC上用Red Hat 9编译了一个有FrameBuffer支持的内核，用它启动能够正常运行所有的MiniGUI程序，移植到开发板上不能正常运行，提示无法找到fbcon的出错信息。后来通过仔细分析fbcon.c文件发现文件中的tty0设备并不存在，需要将改处替换为0。

三、MiniGUI.cfg与资源文件的路径和文件名一定要对应

在移植过程中出现了一个比较奇怪的问题，开发板对文件名长度有一定的限制。资源文件中较长的文件名下载到开发板上自动被截取成了短文件名，导致了MiniGUI.cfg文件中的名称和资源文件不对应，出现错误。后来将资源文件的名称手动修改成长度限制以内的，并相应修改了配置文件解决了问题。

3.4 界面的具体实现

利用 MiniGUI 编写界面类似于 MFC 的开发模式，主要的思想就是消息机制和窗口过程。但是在 Linux 下开发 MiniGUI 的界面没有图形化的工具，只能自行定义界面及控件的定义、尺寸、属性等，通过编译运行后才能直观地反映出来。对于调试来说比较麻烦，所以最终采取了在 Windows 下面 VC.net 环境下调试，具体方法见第六章。

通常，我们编写简单的图形用户界面时，可以通过调用 CreateWindow 函数直接创建所有需要的子窗口，即控件。但在图形用户界面比较复杂的情况下，每建立一个控件就调用一次 CreateWindow 函数，并传递许多复杂参数的方法很不可取。主要原因之一，就是程序代码和用来建立控件的数据混在一起，不利于维护。为此，一般的 GUI 系统都会提供一种机制，利用这种机制，通过指定一个模板，GUI 系统就可以根据此模板建立相应的主窗口和控件。MiniGUI 也提供这种方法，通过建立对话框模板，就可以建立模态或者非模态的对话框。

由于我们设计过程中有些界面非常复杂，而且非常关注与用户的交互－向用户提供输出信息及用于用户输入，过程处理相对简单，所以整个界面程序都采用对话框以及相应控件的形式实现。

MiniGUI 提供了众多的系统控件，有静态框、按钮、列表框和编辑框等，如表 3.6 所示。对话框中通常是利用这些控件实现各种显示输出和设置输入等功能。

表 3.6 MiniGUI 的系统控件类[17]

控件类	类名称	宏定义
静态框	static	CTRL_STATIC

按钮	button	CTRL_BUTTON
简单编辑框	edit	CTRL_EDIT
单行编辑框	sledit	CTRL_SLEDIT
多行编辑框	mledit	CTRL_MLEDIT
列表框	listbox	CTRL_LISTBOX
进度条	progressbar	CTRL_PRORESSBAR
滑块	tackbar	CTRL_TRACKBAR
工具条	toolbar	CTRL_TOOLBAR
新工具条	newtoolbar	CTRL_NEWTOOLBAR
菜单按钮	menubutton	CTRL_MENUBUTTON
属性页	propsheet	CTRL_ PROPSHEET
滚动窗口	ScrollWnd	CTRL_SCROLLWND
滚动视	ScrollView	CTRL_SCROLLVIEW

在使用对话框时，主要是编写相应的过程函数，在我们的程序中，只有键盘一个触发，主要处理的消息为 MSG_INITDIALOG 和 MSG_KEYDOWN 两种。一个完整的对话框调用可以采用以下模板来表示：

```
// 定义对话框
static  DLGTEMPLATE  Dlgmy_login =
{
  WS_BORDER | WS_CAPTION,                    //风格
    24, 32, 192, 64,                         //坐标及尺寸
    "    登录",                              //标题名称
    0,                                       //图标文件
    0,                                       //菜单
    1,                                       //控件数
    NULL,                                    //控件指针
    0                                        //附加属性
};
// 定义对话框中的控件
static CTRLDATA Ctrlmy_login[] =
{
  {
    "static",                                //控件类
      WS_VISIBLE | SS_SIMPLE,                //控件属性
      0, 16, 32, 16,                         //空间坐标及尺寸
      IDC_SPASS,                             //空间ID
      "密码：",                              //空间标题
      0,                                     //附加数据
      WS_EX_NONE                             //扩展属性
  }
};
static int  my_loginDlgProc(HWND hDlg,  int message,  WPARAM wParam,  LPARAM
lParam)                                      //回调函数
{
  switch(message)                            //消息处理
  {
```

```
  case MSG_INITDIALOG:
          {
           ...
           }
    break;
  case MSG_KEYDOWN:
         {
          ...
         }
         break
  }
  return(DefaultDialogProc(hDlg, message, wParam, lParam));
}
void my_loginBox(HWND hWnd)
{
 Dlgmy_login.controls = Ctrlmy_login;  //将对话框模版结构和控件结构数
                                       //组关联起来
DialogBoxIndirectParam(&Dlgmy_login, hWnd, my_loginDlgProc, 0L);

}
```

在本次设计中，为了使显示效果更加直观，我们采用了 240*128 的液晶屏，这样一次显示的内容更加丰富直观，例如员工进行报表和销售操作时，每次需要显示的内容较多，这样就可以充分利用大屏的优势，减少翻页，同时也更加美观。具体显示效果如下图：

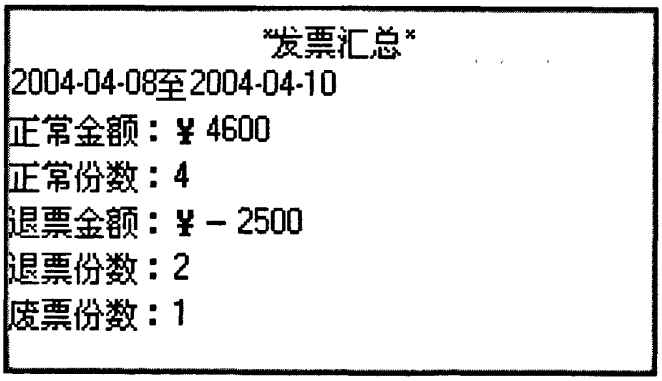

图 3-7 报表界面显示

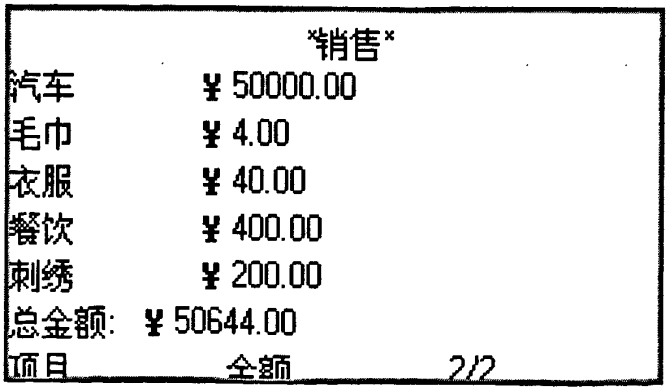

图 3-8 销售界面显示

3.5 本章小结

界面是面向用户的，它的好坏直接影响到用户对税控机的感受。因此本章详细介绍了税控收款机图形界面的完整设计过程。首先按照一定的原则采用菜单方式确定了界面的设计思想。然后比较了目前流行的几种 Linux 下的图形界面，根据系统的硬件平台和功能需求选择了 MiniGUI 做为最佳方案，并详细介绍了 MiniGUI 的移植过程，移植中出现的问题及解决方法。最后结合 MiniGUI 的编程原则说明了界面的具体实现。

第 4 章 嵌入式数据库设计与实现

4.1 数据库的设计

传统的税控收款机对于数据管理都采用人工管理文件的方式，这种管理方式存在着很多缺点，如效率低下、保密性差。另外时间一长，将产生大量的文件和数据，这对于查找、更新和维护都带来了不少困难，因此，设计一个嵌入式数据库应用系统对于税控收款机来说是非常关键的。本章就税控收款机的功能需求完成了一个完整的数据库设计。

4.1.1 数据库概念模型设计

实体主要分为两个模块，一个模块体现在税控机功能方面，主要是发票信息和项目的对应，另外一个主要体现在机器的特有信息方面，这些信息在机器初始化过程中就已经固定了，一般不会改动。

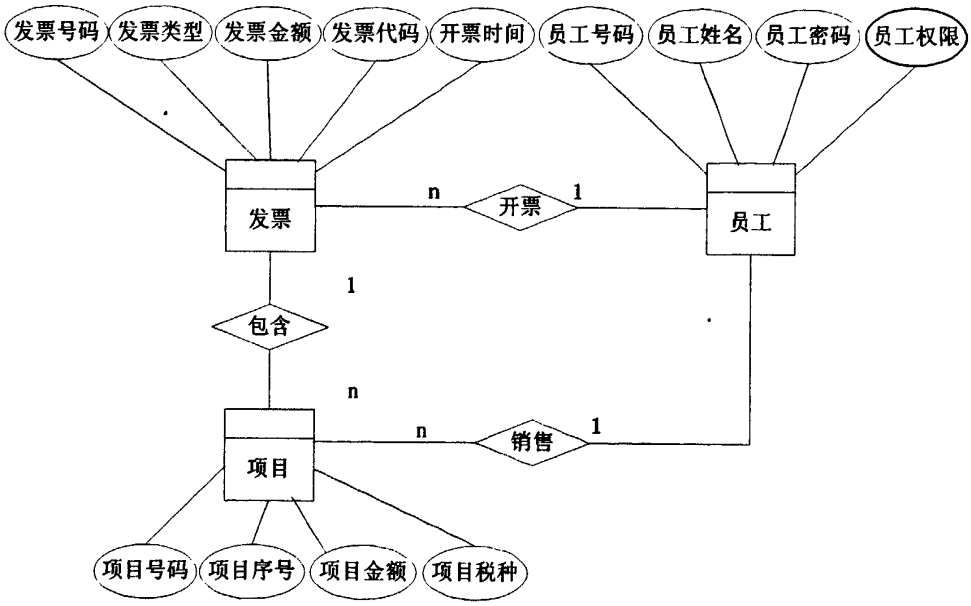

图 4-1 实体关系图(a)

表 4.1 实体关系描述(a)

实体中文名	数据库表名	数据实体描述
员工	Empoyee	指操作机器的员工，可以在员工管理模块进行添加和删除，员工属性是指员工号码，员工姓名，员工密码，员工权限。员工进行销售和开票操作，一个员工可以对应多个销售项目，也可以开多张发票，所以员工与发票和项目实体是一对多的关系。
发票	Invoice_Flow，Invoice_Dsp	发票是税控收款机的重要实

		体，税务流程中主要是传递发票信息。发票属性有发票代码，发票号码，发票类型，发票金额，它还对应有付款单位，原发票号，开票时间。
项目	Item_Info，Sell	项目是发票信息的主要部分这部分可以作为发票实体的弱实体，也可以单独作为实体，由于项目在此牵涉的属性和联系比较多，因此在这里当作一个实体来处理，项目的固定属性主要放在 Item_Info 表中。Sell 表中主要是项目销售的信息。

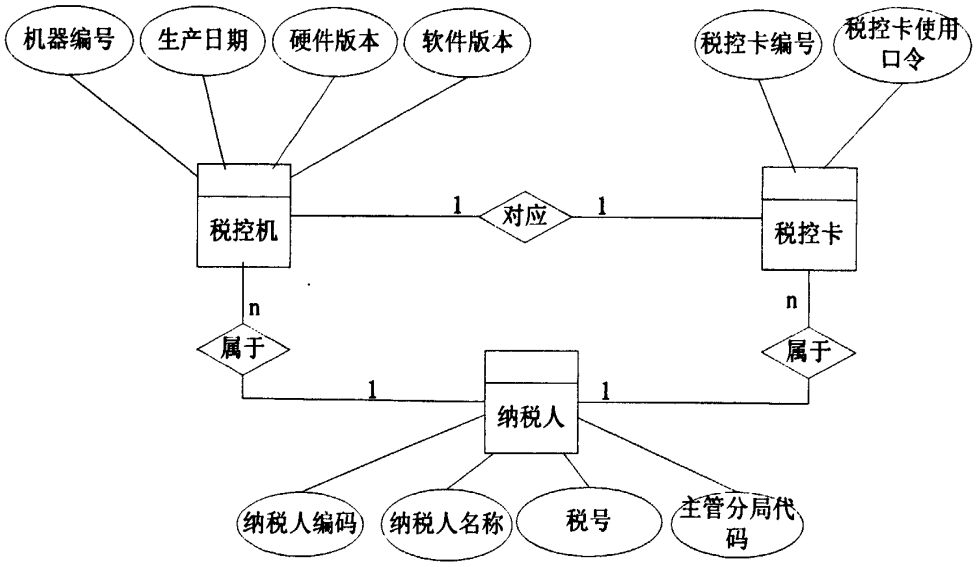

图 4-2 实体关系图(b)

表 4.2 实体关系描述(b)

实体中文名	数据库表名	数据实体描述
税控机	Machine_Info	每台税控机在出厂后都会有一个固定的机器编号和相应信息，通过软件写入。用户购买后会到税务机关发卡，税务后台就根据相应的机器信息及用户需求发一套税控卡和用户卡。在不进行申报和稽查时用户卡可以和机器分离，因此这里不作为必须的实体描述。

税控卡	User_Info，Monitor_Info	税控卡用于控制税控收款机税控数据，鉴别税控收款机身份等，一般情况下一张税控卡对应一台税控机，税控卡中的信息文件必须与机器中相应表的记录相符。
纳税人	User_Info	纳税人是税控机和税控卡的拥有者，但是一个纳税人可以使用多台税控机，因此它和另外两者都是多对一的关系。

4.1.2 数据库逻辑模型设计

从概念结构进行分析、设计，得到数据库全局逻辑结构，包括确定关键字和属性、重新确定记录结构和关系结构、建立各个结构之间的相互关系等。以下是设计系统所有表的结构关系（★代表主键，▲代表外键）。

表 4.3 系统设置表：System_Set

数据属性名	数据类型	数据长度	是否空注解	中文名称
Sound_Set	int	4	否	声音设置
Invoice_Set	int	4	否	发票设置
Invalid_Set	int	4	是	废票设置
Dcl_Type	int	4	否	申报类型
Dcl_Media	int	4	否	申报介质
Dep_Set	char[40]	40	否	付款单位
Graphical_Set	char[40]	40	否	界面设置
Printer_Set	int	4	否	打印设置

表 4.4 用户信息表：User_Info

数据属性名	数据类型	数据长度	是否空注解	中文名称
Taxcard_Num	char[17]	17	否	税控卡编号
Taxcard_Pin	char[17]	17	否	税控卡使用口令
Start_Date	date	9	否	应用起始日期
Valide_Date	date	9	否	应用有效日期
Taxer_Name	char[41]	41	否	纳税人名称
Taxer_Num	char[17]	17	否	纳税人编号
Tax_Id	char[21]	21	否	税号
Dep_Code	long int	4	否	主管分局代码
Dcl_Type	char	1	否	申报方式
Init_Date	date	9	否	初始化日期

表 4.5 监控信息表：Monitor_Info

数据属性名	数据类型	数据长度	是否空注解	中文名称
Up_date	date	9	否	开票截至日期
Single_Limit	long int	4	否	单张发票限额
Sum_Limit	long int	4	否	累计发票限额
Rtn_Limit	long int	4	否	退票累计限额
Tax_Index1	char	1	否	税目税种索引号 1

Tax_Index2	char	1	否	税目税种索引号 2
Tax_Index3	char	1	否	税目税种索引号 3
Tax_Index4	char	1	否	税目税种索引号 4
Tax_Index5	char	1	否	税目税种索引号 5
Tax_Index6	char	1	否	税目税种索引号 6
Dcl_Type	char	1	否	申报方式
Key_Flag	char	1	否	密钥标识

表 4.6 服务商信息表：Service_Info

数据属性名	数据类型	数据长度	是否空注解	中文名称
Service_Name	char[40]	40	是	服务商名称
Service_Tel	char[13]	13	是	电话
Service_Web	char[20]	20	是	网址
Sevice_Reference	char[40]	40	是	联系人

表 4.7 销售表：Sell

数据属性名	数据类型	数据长度	是否空注解	中文名称
Self_Num	char[10]	10	否	员工编号
▲Invoice_Num	long int	4	否	发票编号
Item_Code	char[21]	21	是	项目代码
Item_Seq	int	4	是	项目序号
Item_Money	long int	4	是	项目金额
Item_Tax	char	1	是	项目税种
Depart	char[40]	40	是	付款单位

表 4.8 发票流水表：Invoice_Flow

数据属性名	数据类型	数据长度	是否空注解	中文名称
Invoice_Date	char[13]	13	否	开票时间
Invoice_Type	char	1	否	发票类型
Invoice_Code	char[21]	21	否	发票代码
Invoice_Seq	int	4	否	发票序号
★▲Invoice_Num	long int	4	否	发票序号
invoice_Money	long int	4	否	发票金额
Tax_Ctrl	char[17]	17	否	税控码
Org_Num	long int	4	是	原发票号

表 4.9 机器信息表：Machine_Info

数据属性名	数据类型	数据长度	是否空注解	中文名称
★Machine_Id	char[17]	17	否	机器编号
Mada_date	date	9	否	生产日期
Soft_Edition	char[20]	20	是	软件版本
Hard_Edition	char[20]	20	是	硬件版本

表 4.10 项目信息表：Item_Info

数据属性名	数据类型	数据长度	是否空注解	中文名称
Item_Tax	char	1	否	项目税种索引号
Item_Code	char[21]	21	否	项目代码
★▲item_Seq	int	4	否	项目序号
Tax_Rate	char[6]	6	否	税率
Item_Chname	char[21]	21	是	项目中文名

Item_Enname	char[21]	21	是	项目英文名

表 4.11 发票分发表：Invoice_Dsp

数据属性名	数据类型	数据长度	是否空注解	中文名称
Ivoice_Code	char[21]	21	否	发票代码
Invoice_Sart	long int	4	否	发票起始号
Invoice_End	long int	4	否	发票终止号
MAC	char[9]	9	否	MAC

表 4.12 员工信息表：Employee

数据属性名	数据类型	数据长度	是否空注解	中文名称
★Mem_Num	char[11]	11	否	员工号码
Mem_Name	char[13]	13	是	员工姓名
Mem_Password	char[11]	11	是	员工密码
Mem_Power	int	4	否	员工权限

表 4.13 单卷发票汇总表：Single_Collect

数据属性名	数据类型	数据长度	是否空注解	中文名称
Invoice_Code	char[21]	21	否	发票代码
Invoice_Start	long int	4	否	发票起始号
Invoice_End	long int	4	否	发票终止号
Invoice_Money	long int	4	否	开票总金额
Rtn_Money	long int	4	否	退票总金额
Make_Date	date	9	否	开票起始时间
Up_Date	date	9	否	开票截至时间
Invoice_Count	long int	4	否	开票份数
Rtn_Count	int	4	否	退票份数
Invalid_Count	int	4	否	废票份数

表 4.14 申报表：Declare

数据属性名	数据类型	数据长度	是否空注解	中文名称
Over_Flag	int	4	否	完税标志
State_Word	char	1	否	状态字
Taxcard_Num	char[17]	17	否	税控卡编号
★Dcl_Start	date	9	否	起始日期
Dcl_End	date	9	否	截至日期
Invoice_Count	long int	4	否	开票份数
Rtn_Count	int	4	否	退票份数
Invalid_Count	int	4	否	废票份数
Tax_Index1	char	1	是	税种一索引号
Tax_Well1	long int	4	是	税种一开票金额
Tax_Rtn1	long int	4	是	税种一退票金额
Tax_Index2	char	1	是	税种二索引号
Tax_Well2	long int	4	是	税种二开票金额
Tax_Rtn2	long int	4	是	税种二退票金额
Tax_Index3	char	1	是	税种三索引号
Tax_Well3	long int	4	是	税种三开票金额
Tax_Rtn3	long int	4	是	税种三退票金额
Tax_Index4	char	1	是	税种四索引号

Tax_Well4	long int	4	是	税种四开票金额
Tax_Rtn4	long int	4	是	税种四退票金额
Tax_Index5	char	1	是	税种五索引号
Tax_Well5	long int	4	是	税种五开票金额
Tax_Rtn5	long int	4	是	税种五退票金额
Tax_Index6	char	1	是	税种六索引号
Tax_Well6	long int	4	是	税种六开票金额
Tax_Rtn6	long int	4	是	税种六退票金额
Well_Sum	long int	4	否	开票总金额
Rtn_Sum	long int	4	否	退票总金额
MAC1	char[9]	9	否	MAC1
Elec_Sign	char[257]	257	否	电子签名

表 4. 15 项目汇总表：Item_Collect

数据属性名	数据类型	数据长度	是否空注解	中文名称
★Collect_Date	date	9	否	汇总日期
Item_Seq	int	4	否	项目序号
Item_Name	char[17]	17	否	项目名称
Item_Well	long int	4	否	项目开票金额
Item_Rtn	long int	4	否	项目退票金额

表 4. 16 税务日汇总：Tax_Daily

数据属性名	数据类型	数据长度	是否空注解	中文名称
★Collect_Date	date	9	否	汇总日期
Well_Count	long int	4	否	开票份数
Rtn_Count	int	4	否	退票份数
Invalid_Count	int	4	否	废票份数
Tax_Index1	char	1	是	税种一
Tax_Well1	long int	4	是	税种一开票金额
Tax_Rtn1	long int	4	是	税种一退票金额
Tax_Index2	char	1	是	税种二
Tax_Well2	long int	4	是	税种二开票金额
Tax_Rtn2	long int	4	是	税种二退票金额
Tax_Index3	char	1	是	税种三
Tax_Well3	long int	4	是	税种三开票金额
Tax_Rtn3	long int	4	是	税种三退票金额
Tax_Index4	char	1	是	税种四
Tax_Well4	long int	4	是	税种四开票金额
Tax_Rtn4	long int	4	是	税种四退票金额
Tax_Index5	char	1	是	税种五
Tax_Well5	long int	4	是	税种五开票金额
Tax_Rtn5	long int	4	是	税种五退票金额
Tax_Index6	char	1	是	税种六
Tax_Well6	long int	4	是	税种六开票金额
Tax_Rtn6	long int	4	是	税种六退票金额
Elec_Sign	char[257]	257	否	电子签名

表 4. 17 销售量表：Sell_Amount

数据属性名	数据类型	数据长度	是否空注解	中文名称
Over_Well	long int	4	是	完税开票金额
Over_Rtn	long int	4	是	完税退票金额
Sum_Well	long int	4	是	累计销售金额
Sum_Rtn	long int	4	是	累计退票金额

以上数据表中，根据国标规定税务日汇总是需要保存 5-10 年的。而由于发票流水和销售表数据量很大，所以采用循环表的模式，当发票流水表中发票序号累计到 20000 时两个表就从头循环，这样在保证能够查询较长时间明细数据的同时 64M NandFlash 空间也足够。

4.2 嵌入式数据库的介绍与选用

嵌入式数据库与其他数据库产品的区别是：前者是程序驱动式，而后者是引擎响应式。嵌入式数据库通常与操作系统的具体应用集成在一起，无需独立运行的数据库引擎，由程序直接调用相应的 API 去实现对数据的存取操作，嵌入式数据库是一种具备了基本数据库特征的数据文件，它的一个很重要的特点就是体积非常小，在一些移动设备上具有竞争力[22]。

4.2.1 目前 Linux 下常见嵌入式数据库简介

基于 Linux 平台的数据库非常多，大型的商用数据库有 oracle、Sybase、Informix、IBMDB2 等，中小型的更是不胜枚举，以下是常见的几种[23]：

1. PostgreSQL

PostgreSQL是世界上最优秀的开放源码的数据库之一，是完全免费的数据库，不需要任何版权费用和购买费。因此，它是许多Linux发行版本的首选，例如Redhat、TurboLinux都预装了PostgreSQL。PostgreSQL兼容性很强，如果是SQL92兼容的。移植PostgreSQL非常简单和快捷。

2. MySQL

MySQL是多用户、多进程的SQL database server。MySQL包括一个server daemon（mysqld）和client programs与libraries的client/server实现工具，比较适合小而简单的数据库，对复杂的操作要求支持不是很好。MySQL的使用许可是如果你是普通的最终用户，使用MySQL不需要付费，但如果是直接或间接地出售MySQL的服务程序或相关产品，或是在一些客户端维护MySQL server并收取费用，或是在发行版中包括MySQL就需要获得许可。

3. mSQL（miniSQL）

mSQL是一个单用户数据库管理系统。由于它的短小精悍，使其开发的应用系统特别受到互联网用户青睐。mSQL并非是完全的免费，若是在大学中使用此一软件，或是为了学术研究与慈善等非营利性目的，才能免费得到使用权，否则就得付费注册才能得到正式的版权。

4. BerkeleyDB

BerkeleyDB是一个开放源代码的嵌入式数据库管理系统，能够为应用程序提供高性能的数据管理服务。应用它，程序员只需要调用一些简单的API就可以完成对数据的访问和管理。与常用的数据库管理系统（如MySQL和Oracle等）有所不同，在Berkeley DB中并没有数据库服务器的概念。应用程序不需要事先同数据库服务建立起网络连接，而是通过内嵌在程序中的BerkeleyDB函数库来完成对数据的保存、查询、修改和删除等操作。

5. SQLite

SQLite是2000年开发出来的用小型C库实现的一种中小型嵌入式数据库。可以较为方便地运用于嵌入式系统中。它的源代码完全开放，可以免费运用于任何用途，包括商业目的。SQLite不同于其他大部分的SQL数据库引擎，因为它的首要设计目标就是简单化：

- ✓ 易于管理 ；
- ✓ 易于使用 ；
- ✓ 易于嵌入其他大型程序 ；
- ✓ 易于维护和配置。

SQLite支持绝大多数标准的SQL92语句，采用单文件存放数据库，速度又比MySQL快上1-2倍，存储量2T。在操作语句上更类似关系型数据库的产品使用，非常方便。本系统采用即是SQLite。

4.2.2 SQLite的主要功能特征及优势

SQLite是用C语言编写的开源嵌入式数据库引擎.它是完全独立的，不具有外部依赖性。占用资源非常低，在嵌入式设备中，只需要几百K的内存就够了。它能够支持Windows/Linux等主流操作系统，可与TCL、PHP、Java等程序语言相结合，提供ODBC接口，其处理速度甚至令开源世界著名的数据库管理系统MySQL、PostgreSQL望尘莫及[23]。

一、 SQLite主要功能特征

- ✓ 安装方便；
- ✓ 支持大量数据；
- ✓ 支持大部分SQL命令；
- ✓ 弱数据类型；
- ✓ 速度快，运行速度比MySQL快1～2倍；
- ✓ 零配置—无需安装和管理配置；
- ✓ 存储量大，支持数据库大小为2TB；
- ✓ 体积小，全部代码大概3万行C代码，250KB，代码完全开放；
- ✓ 具备Command窗口，下载一个SQLite.exe文件即可对数据库进行命令行操作；
- ✓ 完全公开的源代码和版权。

二、 SQLite的主要使用场合

仅凭经验来说 SQLite 适用于以下场合：当你更看中简单的管理、使用和维护数据库，而不是那些企业级数据库提供的不计其数的复杂功能的时候，使用 SQLite 是一个比较明智的选择。

- ✓ 网站；
- ✓ 嵌入式设备和应用软件；
- ✓ 应用程序文件格式；
- ✓ 替代某些特别的文件格式；
- ✓ 内部的或临时的数据库；
- ✓ 命令行数据集分析工具；
- ✓ 在Demo或测试版的时候作为企业级数据库的替代品；
- ✓ 数据库教学；
- ✓ 试验SQL语言的扩展。

三、 税控收款机对数据库的需求特点

税控收款机是一款典型的嵌入式设备，它的硬件平台决定了对数据库的需求如下：

✓资源有限。税控收款机存储容量非常有限，为了能够实现要求的功能，必须采用一款体积比较小的数据库；

✓功能完善。税控收款机的关键就在于数据的处理，它的应用需求决定了在其运行中需要有一个大小适中且功能齐备的数据库来实现对数据的管理。对开发人员来说，要求采用的数据库系统提供完备开发的文档，而且易于开发；

✓源码开放。作为一种产品的开发，在功能完备的条件下成本是首先要考虑的问题，选用一款开源的数据库大大减少了生产成本。

综上所述，SQLite简单易用，稳定快速，完全开源，所以最终选择SQLite做为税控收款机的数据库。

4.3 SQLite 的移植

4.3.1 SQLite 的整体分析

SQLite 设计采用一种简单化的思路，这种设计方式不仅大大提高了它的速度，也无形中增加了数据库的稳定性。从整体看来，SQLite 的体系结构如 4-3 图所示[24]：

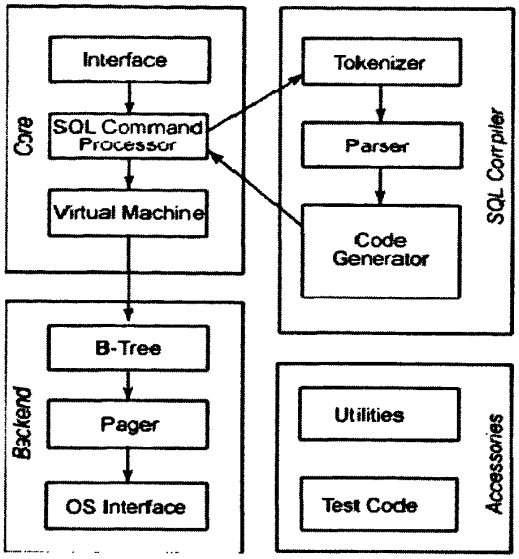

图4-3 SQLite的体系结构

SQLite 链接库的大部分接口的实现函数都能在 main.c、lengacy.c 和 vdbeapi.c 这三个源文件中找到，为了避免和 SQLite 以前的版本发生冲突，接口函数都以 sqlite3 为前缀。

SQLite 的顶层是标记处理器（tokenizer）和分析器（parser）和代码生成器。标记处理器对从接口传入的字符语句进行解析，将解析出的标记符号一个一个传给分析器，分析器根据这些标记符号得到分析树，在分析器完成工作后调用代码生成器，由代码生成器产生相应分析树的虚拟机器代码。SQLite 有自己高度优化的分析器和生成器，可以快速地生产出高效率的代码，而且依靠它新颖的设计对内存泄漏有着特别的抵抗力。

代码生成器生成的虚拟机器程序在虚拟机中执行。为了实现对多个数据库文件进行操作，虚拟机实现了一个虚拟数据库引擎（VDBE），并且使用一个堆栈作为中间存储器。

往下是基于 Knuth 经过优化的 B 树，数据库中的每张表和索引都有自己独立的 B 树，利用 B 树数据库可以运行在可调整的页面高速缓存（Pgae Cache）上，这有助于将对磁盘的查找次数减到最少。

B 树的下面就是页面高速缓存，它作用在 OS 的抽象层之上，这样的安排有助于数据库的移动。

体系结构的核心是虚拟数据库引擎(VDBE)。VDBE 完成与数据操作相关的全部操作，并且是客户和储存之间信息进行交换的中间单元。从各个方面来看，它都是 SQLite 的核心。在 SQL 语句被分析之后，VDBE 开始起作用。代码生成器将分析树翻译成一个袖珍程序，随后这些袖珍程序又被组合成用 VDBE 的虚拟机器语言表示的一系列指令。如此往复，VDBE 执行每条指令，最终完成 SQL 语句指定的查询要求。

VDBE 的机器语言由围绕于数据库管理 128 个操作码（opcode）组成。对于打开表，查询索引、存储和删除记录和很多其他数据库操作都有对应的操作码。在 VDBE 里的每条指令由 1 个操作码(opcode)和 3 个操作数（operand）组成。一些指令使用全部（3 个）操作数；也有些可能一个也未使用。这完全取决于指令的性质。例如 Open 指令，用于打开一个表的指针，使用了全部（3 个）操作数：第 1 个操作数(P1)包含指针的 ID 号；第 2 操作数（P2）指出表的根位置(或者表的首页位置)；而第 3 个操作数则是表的名字。对于 Rollback 指令则不需要操作数。为了进行一次 Rollback，VDBE 仅需知道是否要做 Rollback[25]。

SQLite 支持大小高达 2T 的数据库，每个数据库完全存储在单个磁盘文件中。这些磁盘文件可以在不同字节顺序的计算机之间移动。SQLite 不支持静态数据类型，而是使用列关系。这意味着它的数据类型不具有表列属性，而具有数据本身的属性。当某个值插入数据库时，SQLite 将检查它的类型。如果该类型与关联的列不匹配，则 SQLite 会尝试将该值转换成列类型。如果不能转换，则该值将作为其本身具有的类型存储。SQLite 支持 NULL、INTEGER、REAL、TEXT 和 BLOB 数据类型。

4.3.2 在 PC 机上安装 SQLite

1. 下载 sqlite：你可以到 SQLite 主页 www.sqite.org 上下载 sqlite-3.3.6.tar.gz 软件包，并将这个压缩包解压得到 sqlite-3.3.6 目录；

2. 与 sqlite-3.3.6 同级建一个新的目录 sqlite-pc（mkdir sqlite-pc），进入这个目录执行命令../sqlite-3.3.6/cofigure --disable tcl --prefix = /home/sharon/sqlite-pc/就会在 sqlite-pc 目录下生成 Makefile 及其他相关文件，直接执行 make，make install 即可将 SQLite 安装在 PC 机上了；

3. 查看 sqlite-pc 下的 lib 目录，可以看到 sqlite 的库文件 libsqlite3.a、libsqlite3.la、libsqlite3.so、libsqlite3.so.0、libsqlite3.so.0.8.6 都已经安装好了；

4. 运行/sqlite-pc/bin 目录下的 sqlite3，可以看到下面所示的字样，表明 SQLite 已经在 PC 机上安装成功了；

```
[root@sharon bin]# ./sqlite3
SQLite version 3.3.6
Enter ".help" for instructions
sqlite>
```

5. 这时可以直接输入 SQL 语句进行操作。例如建立用户表 student：

```
sqlite>creat table student(number varchar(10),name varchar(20),age int);
```

向用户表中插入 2 条记录：

```
sqlite>insert into student values ('12345','limeng','20');
sqlite>insert into student values ('23456','zhangxiao','21');
```

进行查询：

```
sqlite>select * from student
```

显示效果如图 4-4：

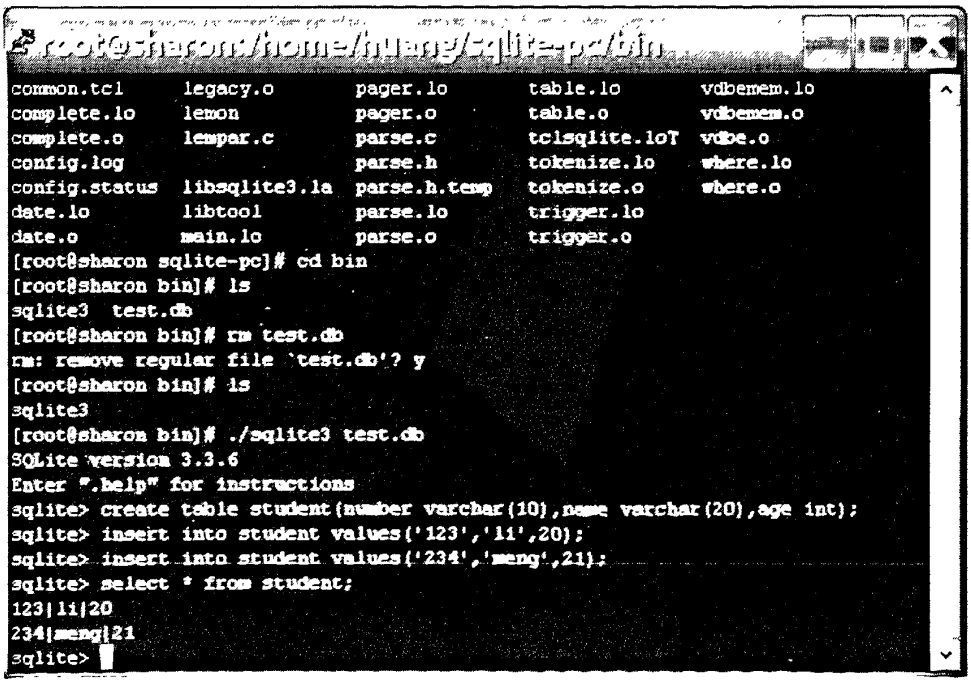

图 4-4 应用 SQLite 实验程序运行效果

4.3.3 SQLite 的交叉编译

SQLite 的交叉编译和 MiniGUI 的交叉编译比较类似，也是先进行 SQLite 库的交叉编译，然后单独编译应用程序。

一、SQLite 库的交叉编译

1. 下载 sqlite-3.3.6.tar.gz from www.sqlite.org。将 sqlite-3.3.6.tar.gz 复制到/home/sharon/biye/W90P710_PR/uClinux-dist 目录下解压缩，生成 sqlite-3.3.6 目录。进入 sqlite-3.3.6 目录并且新建一个 build-cross 目录；

2. 用编辑器打开且编辑 configure 文件 （注意：最好将 configure 文件备份：cp configure configure.old）找到下面语句并屏蔽[26]。保存修改后的配置文件 configure,并进入 build-cross 目录：

```
#if test "$cross_compiling" = "yes"; then

# { { echo "$as_me:$LINENO:: error: unable to find a compiler for building build tools"
>&5

#echo "$as_me: error: unable to find a compiler for building build tools" >&2;}

# { (exit 1); exit 1; }; }

#fi

#else

# test "$cross_compiling" = yes &&

# { { echo "$as_me:$LINENO:: error: cannot check for file existence when cross
compiling" >&5

#echo "$as_me: error: cannot check for file existence when cross compiling" >&2;}

# { (exit 1); exit 1; }; }

#else

# test "$cross_compiling" = yes &&

# { { echo "$as_me:$LINENO:: error: cannot check for file existence when cross
compiling" >&5

#echo "$as_me: error: cannot check for file existence when cross compiling" >&2;}

# { (exit 1); exit 1; }; }
```

3. 执行以下命令：../configure --disable-tcl --host=arm-linux，执行成功后，在 build-cross 目录下，就会产生 Makefile 和 libtool 两个文件；

4. 修改 Makefile 文件：打开 Makefile 文件，因为我们需要编译基于 arm 的 sqlite3 版本的静态库，所以将所有的 libsqlite3.la 替换为 libsqlite3.a；

5. 修改 libtool 文件，清单如表 4.18：

表 4.18 libtool 文件修改清单

原文件	修改
arm-linux	arm-elf
LD="/usr/ /bin/ld"	LD="/usr/ local/arm_tools/arm-elf/bin/ld"
NM="/usr/bin/nm -B"	NM="/usr/local/bin/arm-elf-nm -B"
sys_lib_search_path_spec="/usr/lib/gcc-lib/i386-redhat-linux/3.2.2/ /usr/lib/gcc/i386-redhat-linux/3.2.2/ /usr/lib/gcc-lib/i386-redhat-linux/3.2.2/../../../../i386-redhat-linux/lib/i386-redhat-linux/3.2.2/ /usr/lib/gcc-lib/i386-redhat-linux/3.2.2/../../../../i386-redhat-linux/lib/ /usr/lib/gcc-lib/i386-redhat-linux/3.2.2/../../../i386-redhat-linux/3.2.2/ /usr/lib/gcc-lib/i386-redhat-linux/3.2.2/../../../ /lib/i386-redhat-linux/3.2.2/ /lib/ /usr/lib/i386-redhat-linux/3.2.2/ /usr/lib/"	sys_lib_search_path_spec=" /usr/local/arm-elf/lib "
predep_objects="/usr/lib/gcc-lib/i386-redhat-linux/3.2.2 /../../../crti.o /usr/lib/gcc-lib/i386-redhat-linux/3.2.2/crtbeginS.o"	注释
postdep_objects="/usr/lib/gcc-lib/i386-redhat-linux/3.2.2/crtendS.o /usr/lib/gcc-lib/i386-redhat-linux/3.2.2/../../../crtn.o"	注释
compiler_lib_search_path="-L/usr/lib/gcc-lib/i386-redhat-linux/3.2.2 -L/usr/lib/gcc-lib/i386-redhat-linux/3.2.2/../../.."	compiler_lib_search_path="-L/usr/local /arm_tools/arm-elf/lib"

6. 运行 make 命令，创建 sqlite3 执行文件。编译成功以后，在 build-cross 目录下就有一个隐藏文件夹“.libs”，包含共享目标文件，像：libsqlite.so，或者静态库文件 libsqlite3.a；

7. 将 build/下 sqlite3 下载到开发板上运行./sqlite3 test.db 超级终端显示：

```
SQLite version 3.3.6
Enter ".help" for instructions
sqlite>
```

就完成了 SQLite 库的交叉编译。

二、SQLite 应用程序的交叉编译

首先建立一个 SQLite-Source 目录，将编写的应用程序拷贝到该目录下，然后编写正确的 Makefile 编译安装即可。Makefile 文件具体介绍见《5.2.2 MakeFile 文件的设计》。

4.3.4 移植中遇到的问题

一、有关 TCL 的设置

由于我们应用 SQLite 过程中没有使用 TCL 语言，所以在 PC 机上安装和移植的时候不需要安装 Active TCL 的包，在配制选项的时候也要将有关 TCL 的部分关闭，否则会出现图 4-5 所示的错误：

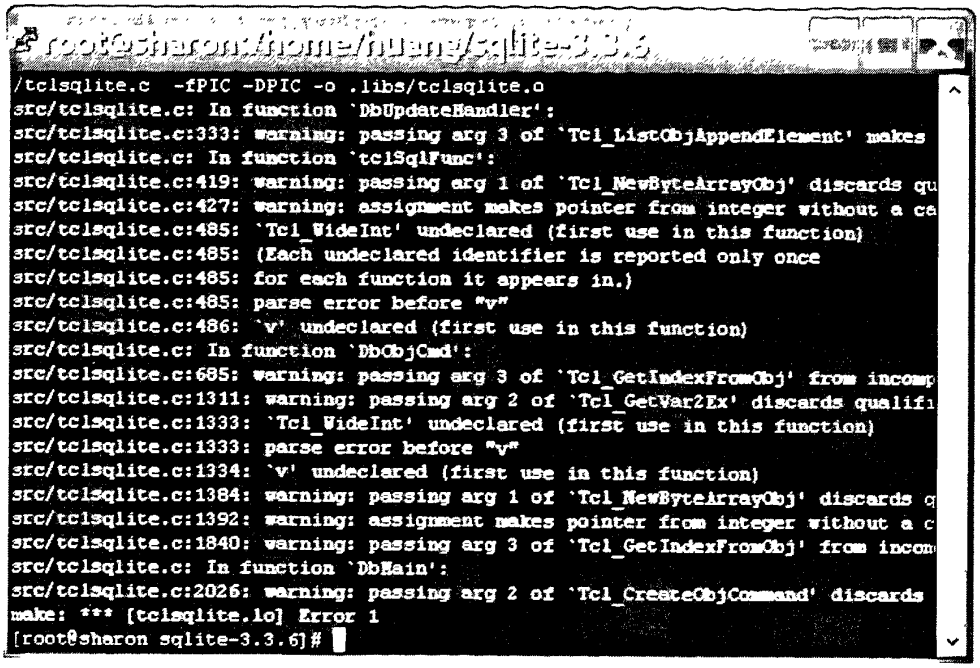

图 4-5 TCL 配置错误显示

二、Makefile 编写

configure 文件的运行可以产生 Makefile 文件，而其中的设置就是 Makefile 的规则。在本次设计中，没有自行编写 Makefile，而是修改了有关 configure 的选项，减少了开发难度。

三、环境变量设置出错问题

我把.libs 目录里面的 sqlite3 下载到板子出现错误提示：

```
error while loading sharad libraries:"libreadline.so.4"
```

后来发现是环境变量的设置问题出错，重新设置环境变量：

```
export LD_LIBRAY_PATH=/***/lib:$LD_LIBRARY_PATH
```

/***/lib 为 SQLite 库文件的路径：

- ✓ libsqlite3.so：symbolic link to libsqlite3.so.0.8.6；
- ✓ libsqlite3.so.0：symbolic link to libsqlite3.so.0.8.6；
- ✓ libsqlite3.so.0.8.6：ELF 32-bit LSB shared objec，ARM，version 1（ARM），stripped；

就是把这三个文件所在的目录的路径加入到 LD_LIBRARY_PATH 中编译后成功。或者是把这三个文件拷贝到开发板的/usr/lib 目录下就 OK 了。

四、设置静态链接库

由于我们在使用 SQLite 的过程中只是将应用程序下载到开发板中运行，所以采用静态库的链接方式，将库函数都编译到应用程序中去，所以在移植的过程中要选择生成静态库，后缀名为.a。

4.4 数据库的具体实现

4.4.1 SQLite 的开发技术

一、SQLite 的基础操作

SQLite 提供的是一些 C 函数接口，你可以用这些函数操作数据库。通过使用这些接口，传递一些标准 sql 语句（以 char*类型）给 SQLite 函数，SQLite 就会为你操作数据库。

SQLite 跟 access 一样是文件型数据库，就是说，一个数据库就是一个文件，此数据库里可以建立很多的表，可以建立索引、触发器等等，但是，它实际上得到的就是一个文件。备份这个文件就备份了整个数据库。

SQLite 不需要任何数据库引擎，这意味着如果你需要 SQLite 来保存一些用户数据，甚至都不需要安装数据库。

二、SQLite 基本 API 函数[27]

1. 关键数据结构：

SQLite 里最常用到的是 sqlite3 * 类型。从数据库打开开始，SQLite 就要为这个类型准备好内存，直到数据库关闭，整个过程都需要用到这个类型。当数据库打开时开始，这个类型的变量就代表了你要操作的数据库。

2. 打开数据库：

int sqlite3_open(文件名, sqlite3 **)，用这个函数开始数据库操作。需要传入两个参数，一是数据库文件名，比如：c:\ Database.db。文件名不需要一定存在，如果此文件不存在，SQLite 会自动建立它。如果它存在，就尝试把它当数据库文件来打开。但是需要注意的是此路径和文件名一定不能包含中文。sqlite3 **参数即前面提到的关键数据结构。

函数返回值表示操作是否正确，如果是 SQLITE_OK 则表示操作正常；相关的返回值 SQLite 定义了一些宏。具体这些宏的含义可以参考 sqlite3.h 文件，里面有详细定义。

3. 关闭数据库：

int sqlite3_open(文件名, sqlite3 **)，前面如果用 sqlite3_open 开启了一个数据库，结尾时不要忘了用这个函数关闭数据库。

4. 执行语句：

int sqlite3_exec(sqlite3*, const char *sql, sqlite3_callback, void *, char **errmsg)，这就是执行一条 sql 语句的函数。第 1 个参数是前面 open 函数得到的指针；第 2 个参数 const char *sql 是一条 sql 语句，以\0 结尾；第 3 个参数 sqlite3_callback 是回调，当这条语句执行之后，sqlite3 会去调用你提供的这个函数；第 4 个参数 void*是你所提供的指针，你可以传递任何一个指针参数到这里，这个参数最终会传到回调函数里面，如果不需要传递指针给回调函数，可以填 NULL；第 5 个参数 char **errmsg 是错误信息，注意是指针的指针。sqlite3 里面有很多固定的错误信息。执行 sqlite3_exec 之后，执行失败时可以查阅这个指针（printf(“%s\n”,errmsg)），得到一串字符串信息，这串信息告诉你错在什么地方。sqlite3_exec 函数通过修改你传入的指针的指针，把你提供的指针指向错误提示信息，这样 sqlite3_exec 函数外面就可以通过这个 char*得到具体错误提示。

说明：通常，sqlite3_callback 和它后面的 void*这两个位置都可以填 NULL。填 NULL 表示你不需要回调。比如你做 insert 操作，做 delete 操作，就没有必要使用回调。而当你做 select 时，就要使用回调，因为 sqlite3 把数据查出来，需要通过回调告诉你查出了什么数据。

5. 不使用回调查询数据库：

上面介绍的 sqlite3_exec 是使用回调来执行 select 操作。还有一个方法可以直接查询而不需要回调，有时候还是需要非回调的 select 查询，这可以通过 sqlite3_get_table 函数做到。回调方法显得代码整齐，效率也相对较高，但是使用比较麻烦，

int sqlite3_get_table(sqlite3*, const char *sql, char ***resultp, int *nrow, int *ncolumn, char**errmsg)，第 1 个参数不再多说；第 2 个参数是 sql 语句，跟 sqlite3_exec 里的 sql 是一样的；第 3 个参数是查询结果，它是一个一维数组（不要以为是二维数组）。它内存布局是：第一行是字段名称，后面是紧接着是每个字段的值；第 4 个参数是查询出多少条记录（即查出多少行）；第 5 个参数是多少个字段（多少列）；第 6 个参数是错误信息。

4.4.2 数据库的实现

建立数据表有两种方式，第一种就是先用 SQLite Database Browser 工具先将所有的表建立好下载到开发板中，第二种方式就是在代码中创建。我在设计中采用后者，这样在文件损坏的情况下程序可以直接修复，而不需要重新手工建表。具体建立过程如下（以建立员工表为例）：

```
int CreateDataBase( char* pszDBPath )              //建立数据库
{
  int iRet;
  char* pszErrMsg;
  iRet = sqlite3_open( pszDBPath, &g_db );
  if( SQLITE_OK == iRet )
  {
    return 0;
  }
  else
  {
    return iRet;
  }
}
// 创建员工表 employee     Mem_Num(员工工号), Mem_Name(员工姓名),
Mem_Password(员工密码), Mem_Power(员工权限)
int CreateEmployeeTable()
{
  int iRet;
  char* pszErrMsg;
  char* pszSql = "create table Employee(Mem_num TEXT, Mem_name TEXT, Mem_password
TEXT, Mem_power int )";
  iRet = sqlite3_exec( g_db, pszSql, 0, 0, &pszErrMsg );
  if ( SQLITE_OK == iRet )
  {
    return 0;
  }
  else
  {
    return iRet;
  }
}
```

数据库创建好之后，我们就可以利用 SQLite 自带的 API 函数完成我们所需要的操作了，由于功能不太复杂，能够完成查询，插入，删除等基本操作就可以了。这里主要调用五个函数[27]：

1. 打开数据库：int sqlite3_open(const char*, sqlite3**)；
2. 关闭数据库：int sqlite3_close(sqlie3*)；
3. 回调函数及运行函数：typedef int (*sqlite3_callback)(void*, int, char**, char**)；
4. int sqlite3_exec(sqlite3*, const char*, sqlite3_callback, void*, char**)；
5. sqlite3_get_table(DB_Handle, SqlStr, &retResult, &nsRow, &nColumn, &retErrMsg)。

第五个函数可以给编程带来方便，但是由于这个函数是在基本 API 函数上面封装起来的，所以运行效率相对较低，所以我在设计中根据实际情况少量调用了这个函数。

4.5 本章小结

本章主要完成了税控收款机嵌入式数据库设计与实现，首先按照数据库的设计方法对税控收款机的数据集做了详细的分析，设计了数据库的概念模型和逻辑模型，然后比较了目前流行的几种 Linux 下的

嵌入式数据库，根据系统的硬件平台和功能需求选择了 SQLite 做为最佳方案，并在此基础上详细介绍了 SQlite 的移植过程，移植中出现的问题及解决方法，最后根据 SQLite 的开发技术讨论了数据库的具体实现。

本章结构与第三章类似，通过这两章的分析为后面界面和数据库的应用设计奠定了基础。

第5章 应用软件设计与实现

应用软件是税控收款机的灵魂，因为税控机最终的功能都体现在应用软件上。而由于 ARM7 处理器没有 MMU（内存管理单元），所以在设计应用程序时一定要考虑到内存的使用，否则就会出错导致程序的崩溃。

应用软件的设计主要有三个重点：

1. 模块化设计，通过分析税控收款机的功能，可以知道整个软件是一个比较复杂的体系，为了使软件层次清晰并且便于今后的升级和维护，必须有一个清晰的模块划分，并且各个模块之间关联尽可能少；

2. 税控流程的设计，关于税控流程国标中已经有了基本的规定，但是随着税控机的迅速发展，仅仅按照国标的基本功能设计是远远不能满足要求的。在本系统的税控设计中，根据实际需求更加细致地考虑了异常，功能也有所丰富；

3. 可靠性设计，可靠性主要涉及数据的正确生成、可靠存储和安全传输。税控收款机的可靠性关系到国家税收的安全，因此可靠性在应用软件设计中也凸显重要。

5.1 软件的功能模块化设计

税控收款机的应用软件采用结构化、模块化设计。模块化是将系统总功能分解为若干个功能单位，各个功能单位被设计成为相对独立的模块，通过相互转移调用的约定（即接口）办法，把它们连接起来。这种设计方法的优点是：系统设计简单、结构紧凑、整体性强、开销小、效率高。在软件设计模块化的同时，每个模块同时使用层次化结构，各层之间是单向依赖关系，不构成循环。这种层次结构的显著优点是：

1. 在设计低层软件时，可不考虑高层软件的实现方法，各层间独立性强；
2. 高层软件的错误不会影响到低层软件中，从而方便操作系统的调试、维护、修改和扩充；
3. 系统不会产生递归调用，避免了死锁的发生，提高了系统的可靠性[28]。

其中设计思想主要体现在税控收款机的三大功能上，即商业功能、税控功能和系统管理功能。商业商业功能包括销售管理和报表管理，税控功能包括初始化操作，开票，申报，完税，税务稽查等操作。同时，各个功能模块之间及系统自检、系统与外部接口之间通过系统的主控模块也即系统管理模块来完成。在模块划分的同时，每个模块又分层次设计，从上到下依次为界面层，数据库应用层以及流程层。

5.1.1 系统管理模块的设计

一、 员工管理子模块设计

员工管理主要有两种，分类管理（权限管理）和单个管理（添加员工、删除员工、修改本人密码、恢复初始密码、查看全部员工信息、员工登录）。当机器初始化之后，数据库默认有一个工号为 “1” 的管理员，操作人在注册员工之前只能以这个管理员的身份登陆，然后注册新员工来操作。

✓ 权限管理在界面层实现，前面已经有所介绍；

✓ 注册员工：只能由管理员来操作，既可以注册管理员也可以注册收款员。注册新员工时，数据库设定了员工号码作为主键，不可有重复工号的员工存在。同时考虑到地税版收款机的需求，对注册员工的数量做了限制，管理员不超过 6 个，收款员不超过 30 个；

✓ 删除员工：由管理员操作，注意这里删除的只能是收款员，所以注册管理员时一定要慎重。考虑到纳税人购买收款机后管理员一般比较少，且比较稳定，故增加了此限制。管理员如有更换，可修改密码工号不变；

✓ 修改密码：只能修改本人密码，密码只能输入数字，不得超过 10 位；

✓ 恢复密码：员工若注册时不设定密码，系统会默认将密码设为“12345678”，若员工忘记设定密码，可通过管理员来将自己的密码恢复为初始密码，再通过修改密码重新设定；

✓ 全部员工信息：用于管理员查看系统内所有员工的信息；

✓ 员工登录：员工开机时会进行一次登陆，如若不进行重新开机换人登陆的话可在此进行。

二、系统设置子模块的设计

系统设置模块主要是对系统的功能进行一系列设定，此模块的功能主要是对系统设置表的操作。机器初始化时表为空，此时员工需要对机器进行一系列设置。主要包含以下几项功能：

✓ 打印机设置：用户需要打印前首先要按打印键查看不同时序下不同行的打印效果，之后选择最佳的效果行输入即可得到最清晰的打印效果。注意输入范围在 1-12 之间；

✓ 声音设置：键盘声音的开关，即系统蜂鸣器的开关；

✓ 发票格式设置：系统内共有省内税务局规定的固定几种格式发票，用户自行选择所要使用的发票类型；

✓ 废票设置：有可以连续废票和废票后进一张发票两个选项，具有复选功能，用户进行废票操作时先要读取此设置；

✓ 申报类型选择：单选功能，有自然月申报和输入截至日期两种选择。当后台发卡时会对申报最大期限做一个限制，选择输入截至日期不可超过此期限；若过期不去申报，机器会自动锁定，纳税人要到税务局解锁后才能使用；

✓ 申报介质选择：单选功能，有用户卡申报、电话线申报、以太网申报和 U 盘申报四种选择；目前由于除用户卡以外的方式还没有一个明确的国标和支持后台，所以只能以用户卡申报，此项功能作为软件升级备用；

✓ 开机问候语：开机界面问候语可以通过该项设置来修改；

✓ 默认付款单位：开发票时需要输入付款单位，若不输入，则默认为此单位。

三、 基本信息查询子模块的设计

此项设计主要是对数据库的操作，用户可以在这个子模块内查询机器的信息。包括以下几项功能：

✓ 用户信息：包括税号、纳税人编码、纳税人名称、主管分局代码、应用启用日期、应用有效日期、税控卡编号、申报方式；

✓ 机器信息：机器编号，生产时间，软件版本，硬件版本；

✓ 税种信息：税种及对应税率信息；

✓ 税目信息：（不同于超市版税控机，项目信息发卡时具体指定）项目代码，项目序号，项目税种，项目中文名称，项目英文名称；

✓ 监控信息：开票截至时间，单张发票限额，单张退票限额，累计开票限额，完税开票金额，完税退票金额，累计开票金额，累计退票金额；

✓ 服务商信息：服务商名称，电话，网址，联系人。

5.1.2 商业管理模块的设计

一、销售子模块的设计

销售模块是收款机运用最频繁的一部分，单独设为一个模块。可以选择子菜单，也可以直接通过【销售】键直接进入：

✓ 销售：销售过程中，员工输入项目序号和项目金额，程序会根据项目序号查询数据库，得出项目代码、项目税种、项目名称，然后进行开票操作。在连续输入项目的过程中，VFD 屏会累计显示金额，用户付款后，会显示应找零数；

✓ 取消：在消费过程中可取消本次消费中的任一项出错项目或顾客取消的项目。员工可以按【C/价格】键进入取消界面，此时会将输入的全部项目及金额显示出来，员工可以通过按上下键找到要删除的项目，确定取消；

✓ 单卷汇总：当一卷发票用完时，系统会自动进行单卷汇总，汇总后可显示该发票卷的信息。

二、报表子模块的设计

报表子模块主要是对一个阶段内销售及发票信息进行一个总结，以便用户了解一段时间内的销售量，单个项目的销售量以及员工业绩等等。查看报表后，用户可以打印报表信息。主要由以下几项功能：

✓ 日期段内发票汇总：用户输入日期段后可以查询该日期内的发票使用情况；

✓ 日期段内单个项目汇总：了解某个时期内某个项目的汇总情况。用户需指定项目序号和日期段；

✓ 单卷汇总：了解某一阶段内已经使用完的发票卷信息，有两种查询方式，发票段和日期，用户输入后，只要在用户指定阶段范围内的发票卷都可以显示出来；

✓ 发票明细：了解某一阶段内的发票详细情况。也可以按发票段和日期段查询。可以查询汇总明细或发票明细；

✓ 电子存根：了解用户指定发票段内每张发票的具体内容，此功能没有显示，查看即需要打印；

✓ 日累计：了解某天的交易情况，包括税种和项目的累计信息；

✓ 员工日累计：了解某天某个员工或全部员工的交易情况，包括税种和项目的累计信息；

✓ 历史申报数据：了解某一时间段内的申报数据情况，用户可指定一个日期，包含该日期的申报数据都可查询出来。

5.1.3 税务管理模块的设计

一、发票管理子模块的设计

发票管理模块主要是对税控机的发票进行管理。按常规来说，退票和废票是销售的另外两种方式，这里放入了发票管理子模块，主要是因为这两个模块使用比较少，没有必要和销售放在一次每次做选择。该模块主要包括以下几项功能：

✓ 分发发票：纳税人通过用户卡在税务局购买了发票，就可以将发票分发到机器的数据库中，税控收款机只能打印纳税人从税务机关领购的发票号码段内的发票；

✓ 安装发票：发票分发到机器中以后，用户需要将一卷发票安装到税控卡内才可正常开票。进入安装发票界面后，当前发票库的信息都会显示出来，用户以单选模式选择一卷安装；

✓ 退票[29]：退票主要在两种情况下进行，一是退货，二是发票打印数据错误。例如项目、数量或金额错误，发票打印号和印刷号不通等。用户首先在退票界面输入要退的发票号，若程序检索到该发票确实用过就会进行退票操作。当发票上有多个交易项而仅对其中部分交易做退货操作时，应先将该张发票上的全部交易项做退货操作，打印出退票后再对未退货的交易项打印一张正常发票；

✓ 废票：发票信息打印不完整的发票作废票处理。目前机器只能废除当前票；

✓ 当前发票卷：用于用户查看当前发票卷的信息，当一卷发票快要用完时避免同时打印多张发票；

✓ 分发未使用发票：查看发票库中还未安装的发票信息。

二、税务管理子模块的设计

税务管理模块是税控收款机最主要的功能，此模块要做到安全可靠，考虑到所有的异常情况发生，才能有利于税控机的推广使用。主要包含以下几项功能：

✓ 税务初始化就是用户卡和税控卡相互进行认证，认证成功后读取税控卡的基本信息文件中“应用起始日期”、“应用截止日期”、“纳税人单位名称”，并将上述信息写入税控收款机的存储单元；读取用户卡“税种税目索引文件”中的税率信息，并写入税控收款机的存储单元；

✓ 开票：和销售部分衔接的操作。开票有三种类型，正常开票，退票，废票，通过发票类型区别；

✓ 申报：包括汇总申报，日交易申报，明细申报和单卷申报四种类型，其中汇总和单卷是必须要进行的。而另外两种方式可以通过发卡选择，申报后申报标志写“0”。注意当天的交易信息是不能申报的，如要申报必须通过稽查卡将当前时钟修改为明天；

✓ 完税：用户在税务局后台完税后通过用户卡在机器上进行完税操作，对机器的信息进行更新，完税后申报标志会自动更新为“1”；

✓ 更新纳税人基本信息：当纳税人信息发生变化时，需要进行纳税人基本信息的更新。更新时用户需要拿税控卡和用户卡到税务后台同时更新，否则两者信息不一致在机器上会更新失败；

✓ 稽查：稽查主要用于税务管理人员对机器中的信息进行读取并检查，分为明细稽查，日交易稽查，申报稽查，修改时钟四种类型。机器通过判断稽查卡的类型进行相应操作。

5.2 软件主要流程的实现

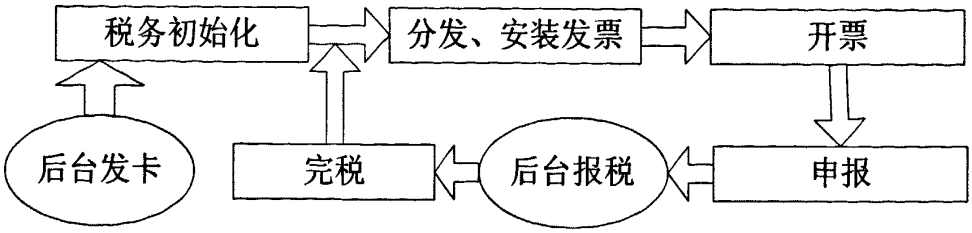

图 5-1 税控机使用流程[30]

如上图所示，当纳税人购买了一台税控机后首先要到税务局发卡和领取发票， 然后用户会用税控卡和用户卡相互认证对机器进行初始化，将机器及用户信息写入机器的数据库中。初始化成功后，用户会将用户卡里的发票分发到机器中并安装单卷。这时的税控机就可以正常的开票（包括退票，废票）使用了。到了申报时间，用户可以通过用户卡（或者 U 盘，以太网等其他方式）将指定时间内的发票数据读取出来，并拿到后台去报税。后台会给用户卡写入完税信息后，用户就可以通过卡给机器完税，更新相应信息后就可以开始新一轮的税务监督工作。这就是整个税控收款机的工作流程（如图 5-1 所示）。

5.2.1 开机流程

入口函数为 main()函数。税控机的开机有两种操作模式，第一种就是未初始化状态，此时进行机器的初始化操作，此状态只在用户第一次使用之前进行一次。第二种就是初始化过后的机器正常状态。具体流程如图 5-2：

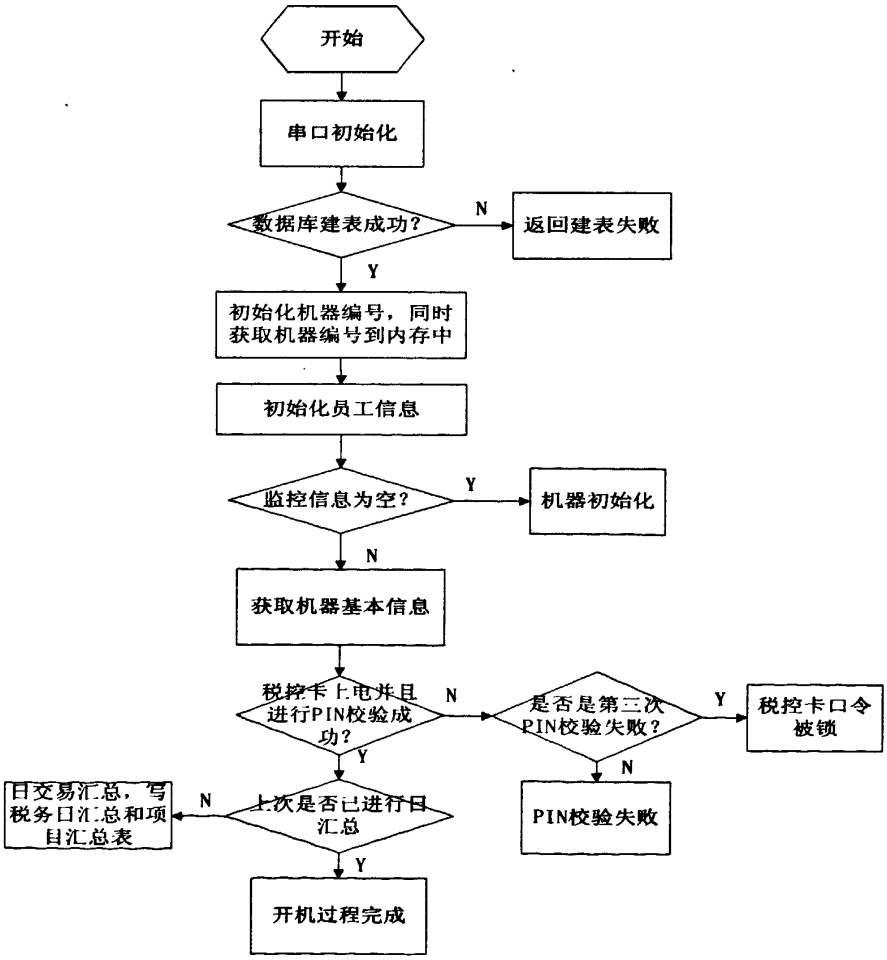

图 5-2 税控机开机流程

5.2.2 税务初始化流程

int Init(void)；

功能描述：初始化界面入口；

返回值：0->成功，非 0->失败；

unsigned short int TaxAffairInit()；

功能描述：税务初始化执行函数；

返回值：0->成功，非 0->失败(具体错误类型看代码注释)；

具体操作流程如图 5-3：

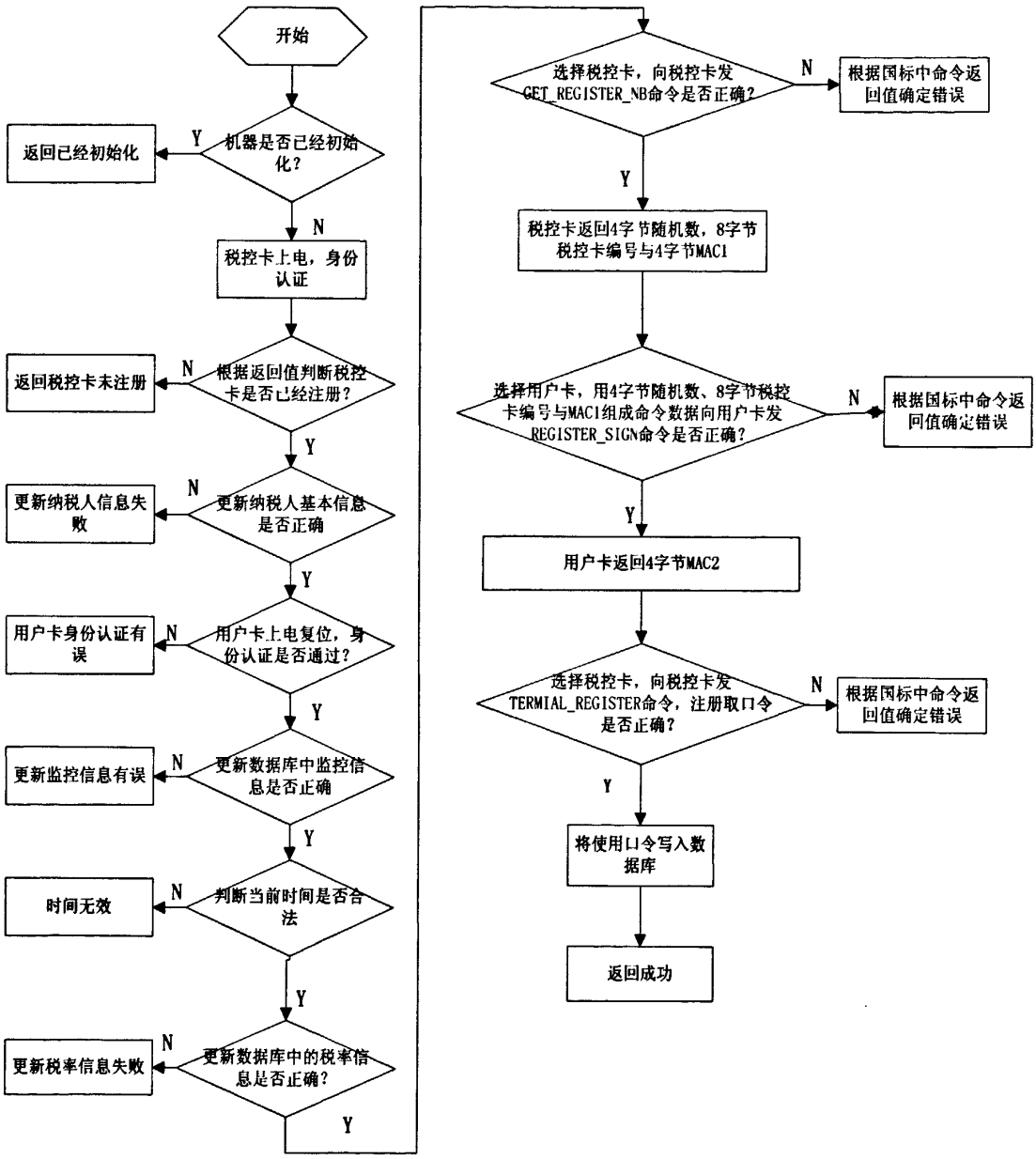

图 5-3 税务初始化流程

5.2.3 发票分发和安装

一、分发发票

int invoice_dsp(void)；

功能描述：分发发票入口；

返回值：0->成功，非 0->失败；

unsigned short int InvoiceDistribute(unsigned int count)；

功能描述：分发发票执行函数；

入口参数：分发卷数；

返回值：0->成功，非 0->失败(具体错误类型看代码注释)；

void invoice_dsp_result(void)；

功能描述：分发结果显示；

具体操作流程如图 5-4：

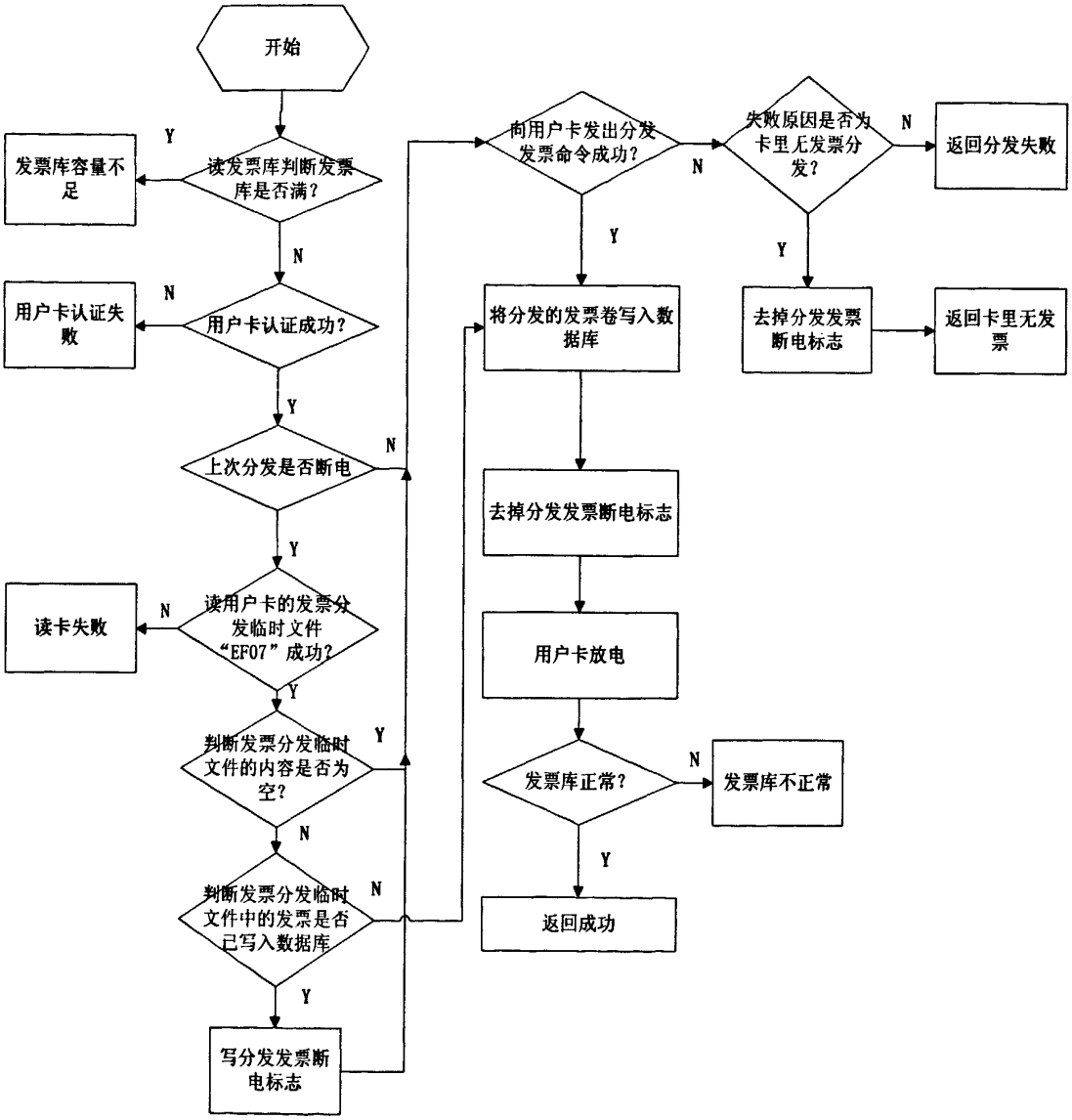

图 5-4 分发发票流程

二、安装发票

int invoice_enter(void)；

功能描述：安装发票入口；

返回值：0->成功，非 0->失败；

void QueryInvoiceInfo()；

功能描述：查询发票库信息；

unsigned short int InvoiceInstall(unsigned long int STARTNum,unsigned long int ENDNum)；

功能描述：安装发票执行函数；

入口参数：发票起始号和终止号；

返回值：0->成功，非 0->失败(具体错误类型看代码注释)；

具体流程如图 5-5：

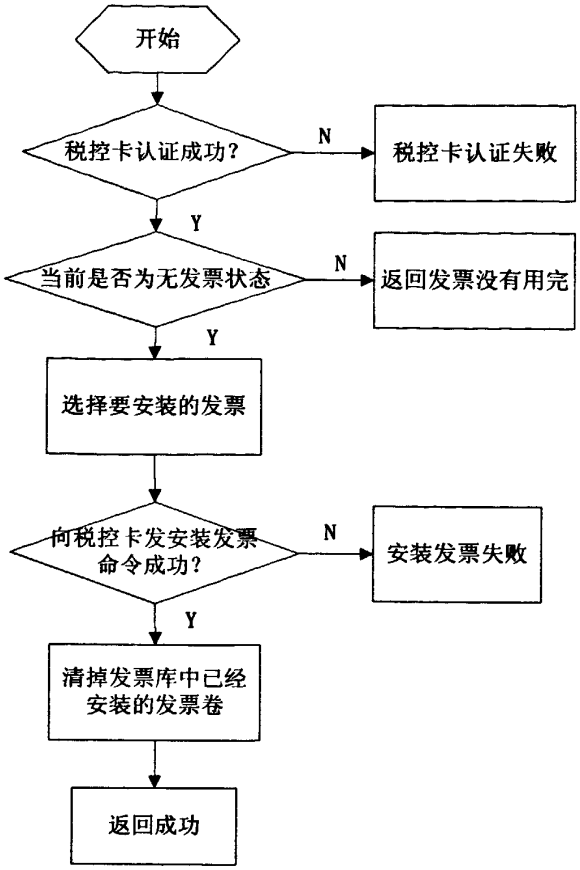

图 5-5 安装发票流程

5.2.4 开票

一、获取当前发票号

unsigned long int Get_CurrentNum_Single();

功能描述：一次获取当前发票号；

返回值：0->成功，非 0->失败；

具体流程如图 5-6：

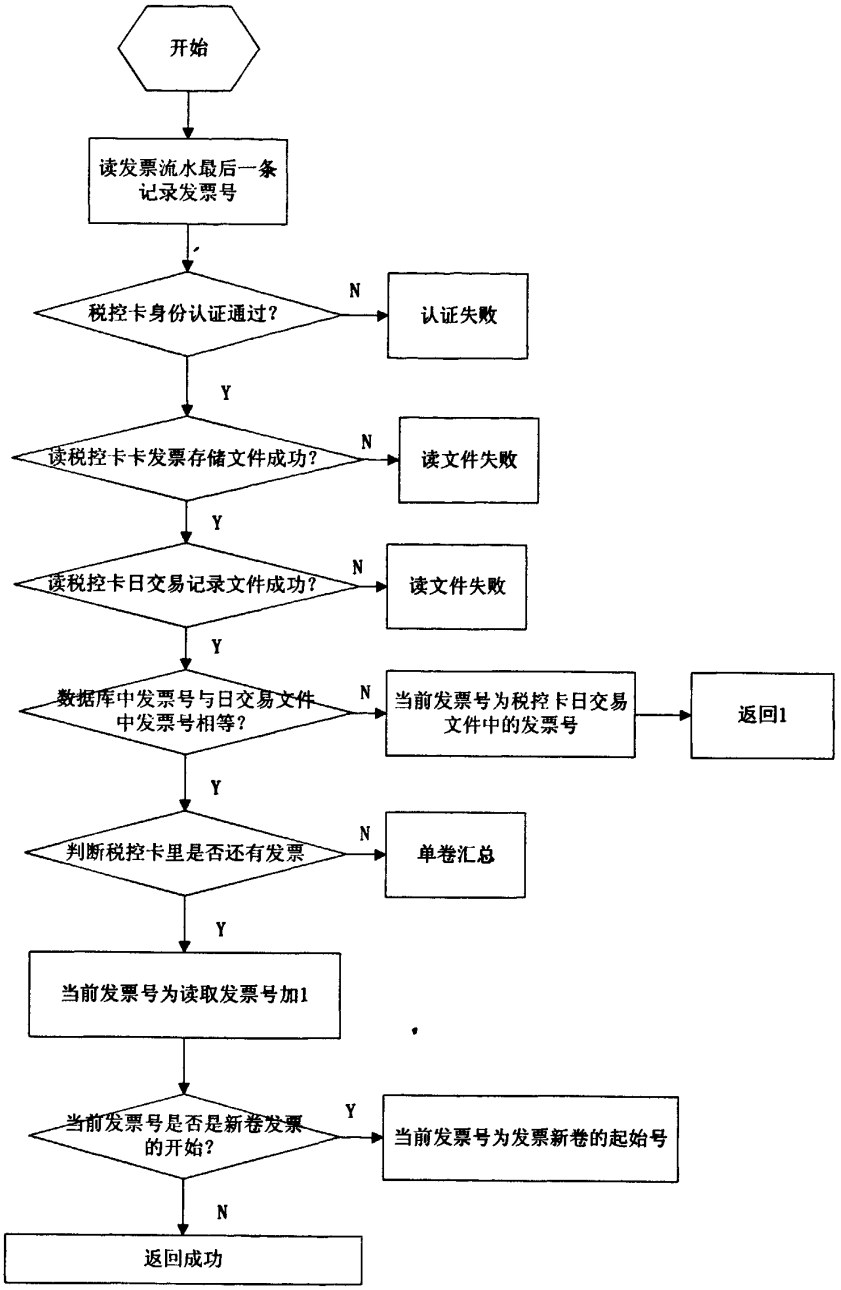

图 5-6 获取当前发票号

二、向税控卡发开票命令

void SellStatistics();

功能描述：销售项目分类统计；

unsigned short int InvoiceMakeout(unsigned char MAKEOUTType, unsigned long int ORIGINALInvoiceNum, unsigned char *TRADEAssoetedIndex, unsigned long int *TRADEAssoetedMoney, struct sttSell *sttSellInfo, unsigned int ITEMSCnt);

功能描述：向税控卡发开票命令；

返回值：0->成功，非 0->失败(具体错误类型看代码注释)；

入口参数如表 5.1：

表 5.1 开票函数入口参数表

参数	意义
MAKEOUTType	开票类型

ORIGINALInvoiceNum	原发票号
TRADEAssoetedIndex	税种索引号
TRADEAssoetedMoney	税种索引分类金额
sttSellInfo	销售结构体
ITEMSCnt	项目数量

具体流程如图 5-7

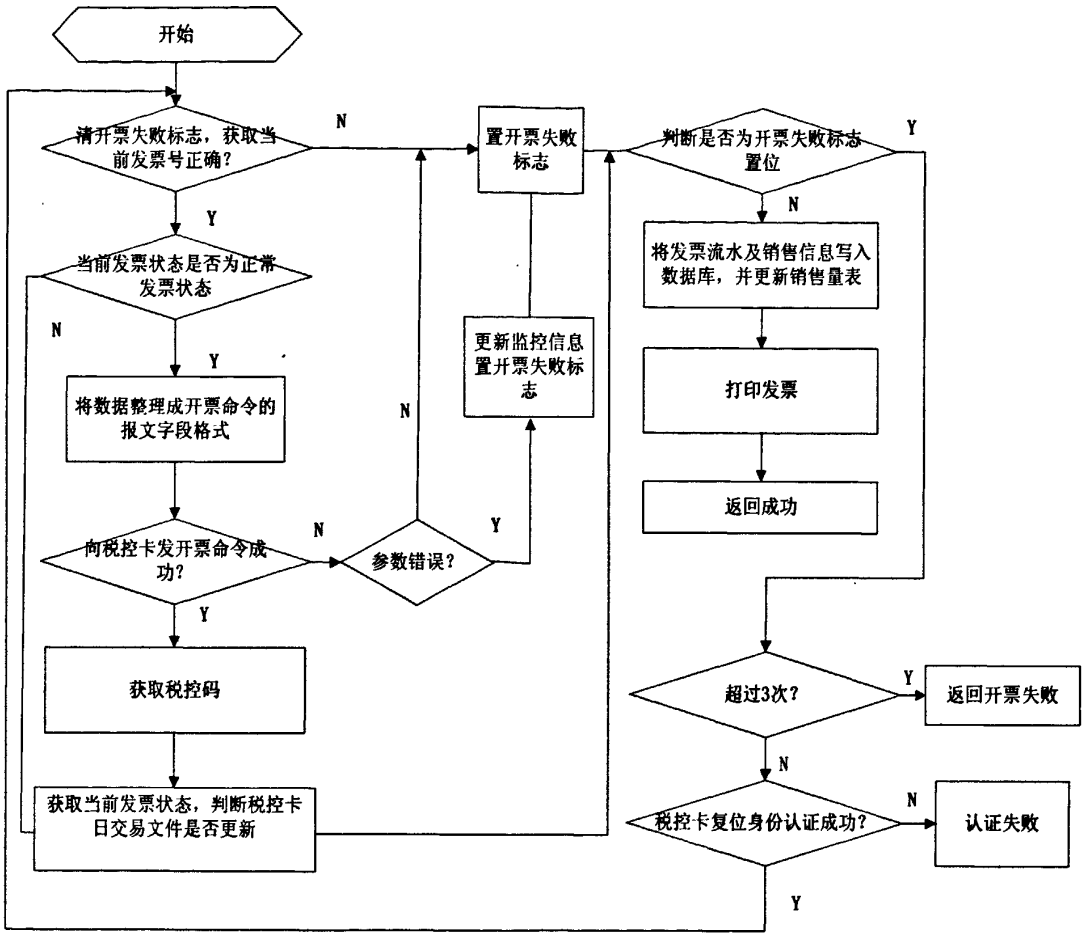

图 5-7 向税控卡发开票命令

5.2.5 汇总申报

unsigned short int DeclareJudge(unsigned char *TAXStartDate);

功能描述：判断抄税的用户卡的合法性；

入口参数：本次抄税的起始日期；

返回值：0->成功，非 0->失败(具体错误类型看代码注释)；

unsigned short int TaxpayDeclare(unsigned char *STARTDate,unsigned char * ENDDate);

功能描述：汇总抄税；

入口参数：STARTDate 起始日期，ENDDate 截至日期；

返回值：0->成功，非 0->失败(具体错误类型看代码注释)；

具体流程如图5-8：

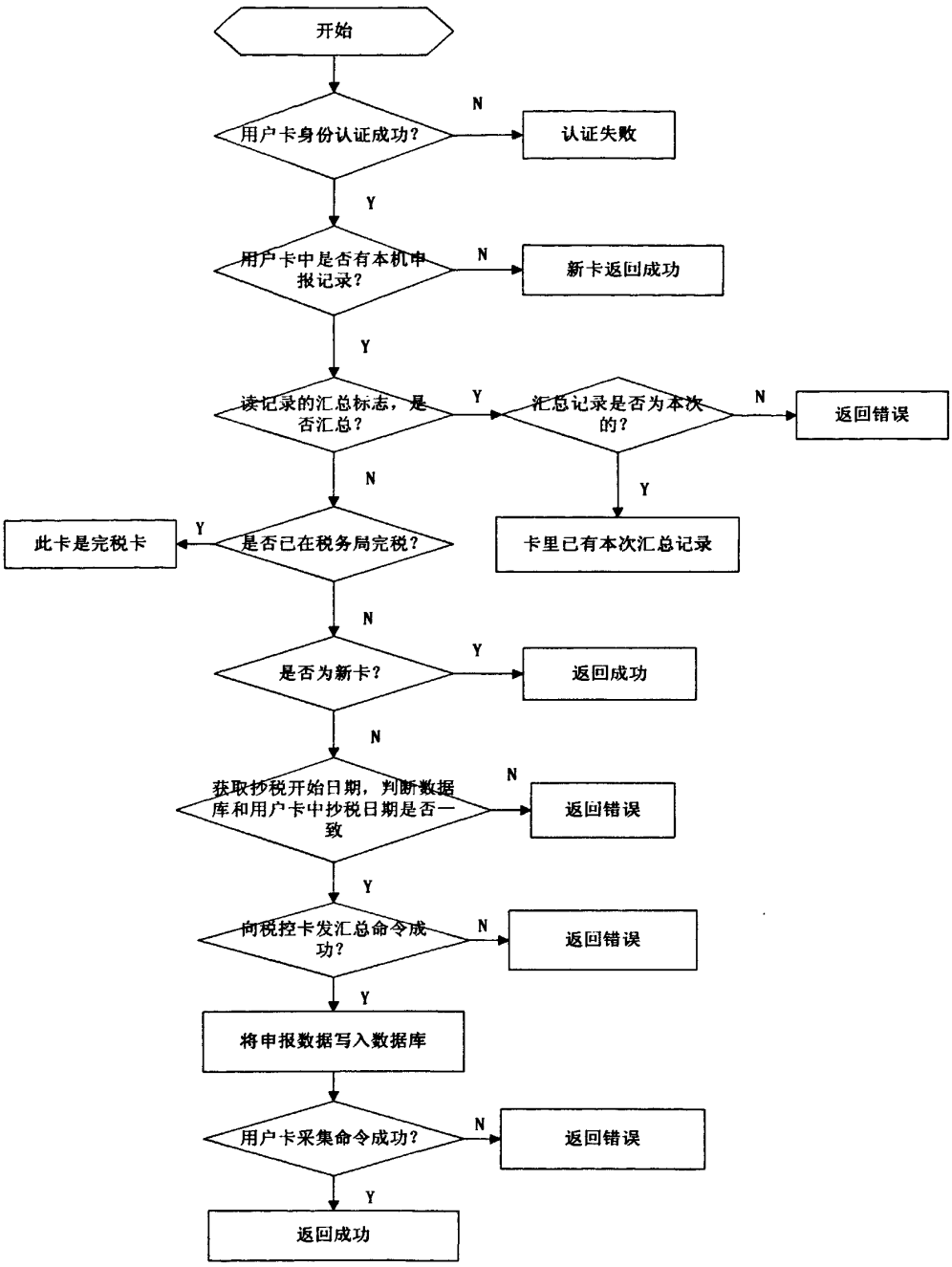

图5-8 汇总抄税

5.2.6 完税

unsigned short int OverTaxJudge();

功能描述：判断此次用户完税的合法性；

返回值：0->成功，非0->失败(具体错误类型看代码注释)；

unsigned short int OverTax();

功能描述：用户卡完税；

返回值：0->成功，非0->失败(具体错误类型看代码注释)；

具体流程如图5-9：

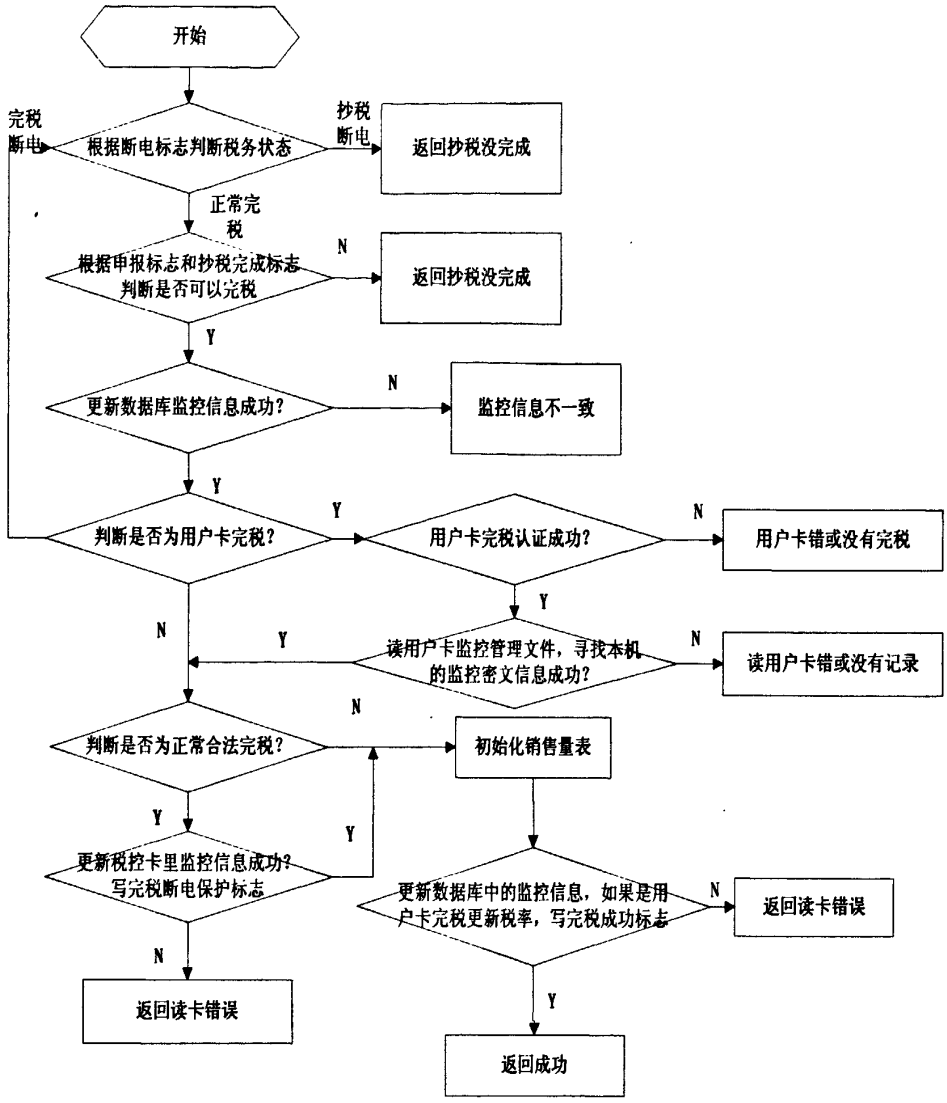

图 5-9 完税

5.2.7 明细稽查

unsigned short int CheckcardAuthentication();

功能描述：稽查卡认证；

返回值：0->成功，非 0->失败(具体错误类型看代码注释)；

unsigned int CheckTakeInit();

功能描述:将稽查卡的记录的关键字读到款机的外部缓存；

返回值：0->成功，非 0->失败(具体错误类型看代码注释)；

static unsigned int InvoiceCheckNumber(long int invoice_num1, long int invoice_num2);

static unsigned int InvoiceCheckDate(char *date1, char *date2)

功能描述：明细稽查发送到卡；

入口参数：稽查的起始号（起始日期）或终止号（终止日期）；

返回值：0->成功，非 0->失败(具体错误类型看代码注释)；

具体流程见图5-10：

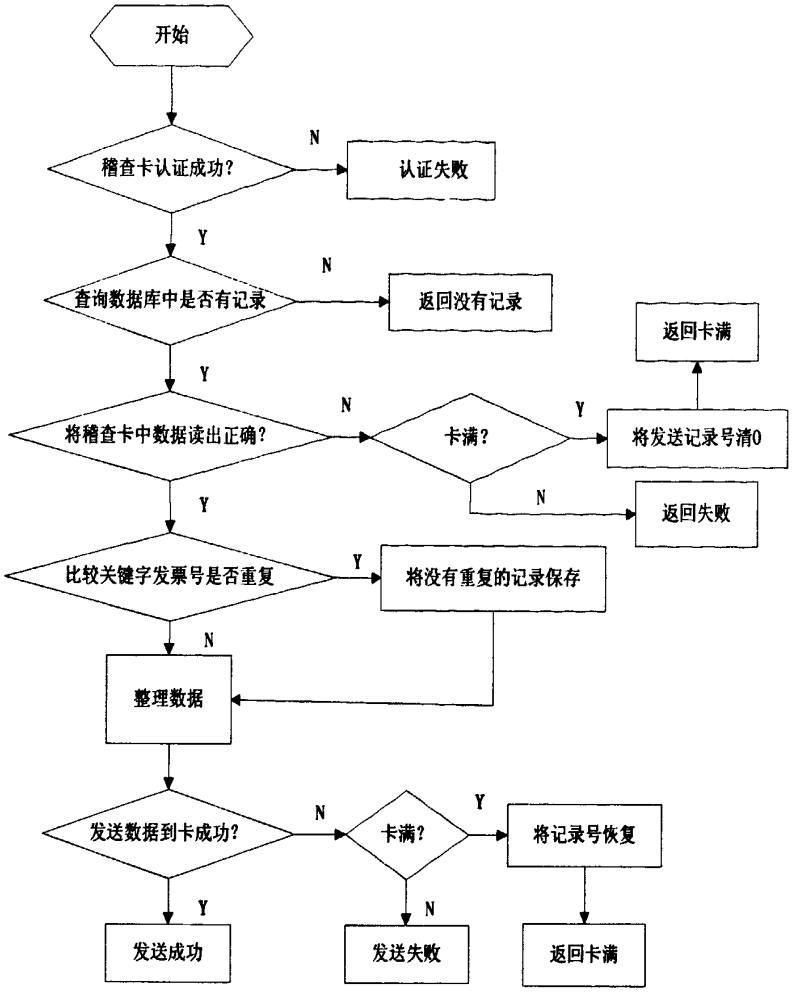

图5-10 明细稽查

鉴于日交易稽查和申报稽查与明细稽查流程类似。修改时钟直接调用系统函数即可实现，流程比较简单，这里不多赘述。

5.3 应用程序的交叉编译

为了能让应用程序运行在不同的目标平台上，需要有针对相应平台的交叉编译工具。目前，常用的交叉开发环境主要有开放和商业两种类型[31]。交叉开发环境是指编译、链接和调试嵌入式应用软件的环境。它与运行嵌入式应用软件的环境有所不同，通常采用宿主机/目标机模式，在宿主机上编写好源代码及相关的配置文件，交叉编译和链接。在应用程序编写完成后，就可以进行编译和链接，以生成可执行代码，下载到开发板上运行。前面已经对 MiniGUI 和 SQLite 的交叉编译做了一些介绍，这里我们只是简单介绍一下。

交叉编译一般要安装一个交叉编译工具，也就是搭建一个交叉编译环境，然后自己编写一个 Makefile 文件来编译自己的应用程序，编译成功的情况下会生成可执行程序。

5.3.1 交叉编译环境的建立

要编译某个特定 CPU 的内核，首先需要建立编译这个体系处理器的编译环境。这个编译环境就是指编译内核所需的工具 ld、as、ar、ranlib、gcc，以及相关的库文件。我建立交叉编译环境的步骤如下：

1. 首先要在 PC 机上安装好 Linux 操作系统，这里选择 Red Hat 9 版本。为了实验室同学使用方便我们搭建了一个 samba 服务器，大家可以以不同身份登陆同时使用一个操作系统。

2. 安装交叉编译工具。这里安装华邦自带的压缩包 arm_tools_3.3.tar.gz，解压后运行 install.sh，交叉编译工具的文件就会安装在用户环境变量默认的路径 usr/local/arm_tools 下。具体文件如表 5.2：

表 5.2 arm_tools 目录下文件列表

/usr/local/arm-tools/bin	编译器和链接器的二进制文件，install.sh 文件会把这个目录添加到所有的用户环境变量中。
/usr/local/arm-tools/arm-elf/inc	头文件
/usr/local/arm-tools/arm-elf/lib	链接库（C 和 pthread 库）

5.3.2 MakeFile 文件的设计

GNU make(简称 make)是一种代码维护工具，在大中型项目中，他将根据程序各个模块的更新情况，自动维护和生成目标代码。Make 的主要任务是读入一个文本文件（缺省文件名是 makefile 或 Makefile），并根据这个文本文件定义的规则和步骤，完成整个软件项目的维护和代码生成等工作。这个文本文件里主要定义了依赖关系（即有关哪些文件的最新版本是依赖于哪些其他文件产生或组成的）和需要用什么命令来产生文件的最新版本或管理各种文件等工作[32]。在 Windows 下，IDE 已经为编程者写好了 Makefile，不需要程序员手动编写。而我们的程序如果想运行在 Linux 操作系统下就需要自己编写一个合适的 Makefile。

在我们的设计中，编译应用程序可以采用两种方式，第一种就是将应用程序直接放到内核中，修改内核的 Makefile 文件，最终产生一个映像文件到目标板上。第二种就是给应用程序编写一个 Makefile 文件，单独编译应用程序，这种方式的 Makefile 相对简单，编译过程也会简化。所以我采用了后者。

Makefile 规则比较多，针对不同的应用程序可以编写一个比较简洁的文件。在本次设计中，我设计的 Makefile 主要完成以下功能：

- ✓ 指定编译头文件，链接库及交叉编译器的路径的路径；
- ✓ 指定交叉编译器；
- ✓ 指定源文件及目标文件。

具体内容如下（解释见表 5.3）：

```
.SUFFIXES : .x .o .c .s
ROOT = /usr/local/arm_tools
LIB = $(ROOT)/lib/gcc-lib/arm-elf/3.0
LIB1 =$(ROOT)/arm-elf/lib
LIB2=/home/sharon/biye/minigui/lib
LIB3=/usr/local/lib
INC :=$(ROOT)/arm-elf/inc
INC1:=/home/sharon/biye/minigui/include
INC2:=/usr/local/include

CC=arm-elf-gcc -I$(INC) -I$(INC1) -I$(INC2)   -Wl,-elf2flt

WEC_LDFLAGS=-L$(LIB) -L$(LIB1) -L$(LIB2) -L$(LIB3)
TARGET = Tax
SRCS :=   *.c

LIBS= -lc -lgcc -lc -lminigui -lmgext -lm -lvcongui -lsqlite3
all:
        $(CC) $(WEC_LDFLAGS) $(SRCS) -o $(TARGET) $(LIBS)
clean:
        rm -f *.o
        rm -f *.x
```

```
rm -f *.flat
rm -f *.map
rm -f temp
rm -f *.img
rm -f $(TARGET)
rm -f *.gdb
```

表 5.3Makefile 内容详解

Makefile 参数	意义
LIB-LIB3	所有库文件的路径
INC-INC2	为所有头文件的路径
LIBS	指定具体的链接库（这样就可以将 MiniGUI 和 SQLite 连接起来了）
CC	交义编译其为 arm-elf-gcc
-elf2flt	生成可执行文件格式为 FLAT
TARGET	目标文件的名称
SRCS := *.c	表示编译当前目标下的所有.c 格式文件
clean	运行 make clean 时需要将上面后缀的文件删除

5.4 本章小结

本章是论文的核心内容，主要介绍了应用软件的详细设计过程和实现。文章首先通过税控机的功能需求及相关国标确定了应用软件的设计原则。然后介绍了整个程序的详细模块划分及相应功能；税务流程是税控机最重要也是安全性要求最高的功能，第二部分详细介绍了重要税务流程的实现；最后讨论了应用程序在 W90P710 开发板上的移植过程。

第 6 章 系统开发与调试

软件调试就是要通过调试软件实现对被调试软件的执行流程地控制、执行状态的获取、代码以及变量的更改，用来辅助软件开发人员发现程序中所存在的问题。软件调试是软件开发过程中无法避免的一个阶段，嵌入式软件开发更是如此。

6.1 调试方法的选择

6.1.1 PC 机 Linux 下调试

一般情况下，在 Linux 操作系统下调试嵌入式软件的方法有两种：

第一种就是搭建 GDB 调试平台，GDB 是一个交互式工具，工作在字符模式。GDB 功能非常强大，可完成如下的调试任务[33]：

- ✓ 设置断点；
- ✓ 监视程序变量的值；
- ✓ 程序的单步执行；
- ✓ 修改变量的值。

由于嵌入式系统资源有限性，一般不可能直接在目标系统上进行调试，通常采用 gdb+gdbserver 的方式进行调试。gdbserver 在目标系统中运行，gdb 则在宿主机上运行，具体操作方法见图 6-1。这种方式虽然使用非常方便，但是环境的搭建非常复杂。

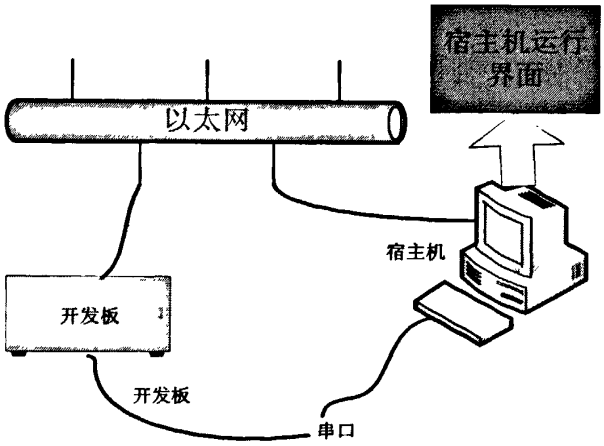

图 6-1 GDB 调试示意图

第二种方法就是直接利用超级终端在可能出现问题的地方打印调试信息，这样虽然不需要做任何准备，但是使用起来却非常麻烦，每做一次修改都需要重新交叉编译，查看修改效果，对于比较复杂的程序一般不予采用。

6.1.2 Windows 下 VC.Net 下编译调试

飞漫软件为习惯于 Windows 编程的开发人员提供了一个可选组件--MiniGUI SDK for Windows。通过这个 SDK，再加上一个 Windows Virtual FrameBuffer（WVFB），开发人员就可以在 Windows 环境下编译和调试 MiniGUI 应用程序了[15]。同样，SQLite 也可以利用相应的工具在 Windows 下编译。这样，我就在本次调试界面和数据库的部分采用了 VC.net 开发环境下的调试工具，既完成了 GDB 的应有功能，又避免了复杂的平台建立过程。在 Windows 下建立一个 MiniGUI 和 SQLite 的调试平台步骤如下：

1. 首先下载一个 MiniGUI SDK for Windows 源码包；
2. 新建一个 VisualC++项目下的 Win32 控制台项目；

3. 将源代码中的.c 文件和.h 文件拷贝到工程目录下，分别保存在 src 和 include 目录下，包括 MiniGUI 的头文件。注意链接 SQLite 除了可以加动态链接库以外也可以直接将 Linux 下交叉编译产生的 sqlite3.c 和 sqlite3.h 也加到相应目录中去；

4. 将 minigui.dll 和 pthreadVC1.dll 两个库文件拷贝到工程的 lib 目录下；

5. 设置工程属性，将编译要依赖的库文件，库文件路径，头文件路径设置在相应的属性中设置好，最好选择相对路径，这样工程可以整体移动；

6. 编译；

7. 运行 wvfb.exe 程序；

8. 运行应用程序，就可以完成 Windows 平台下的运行了。

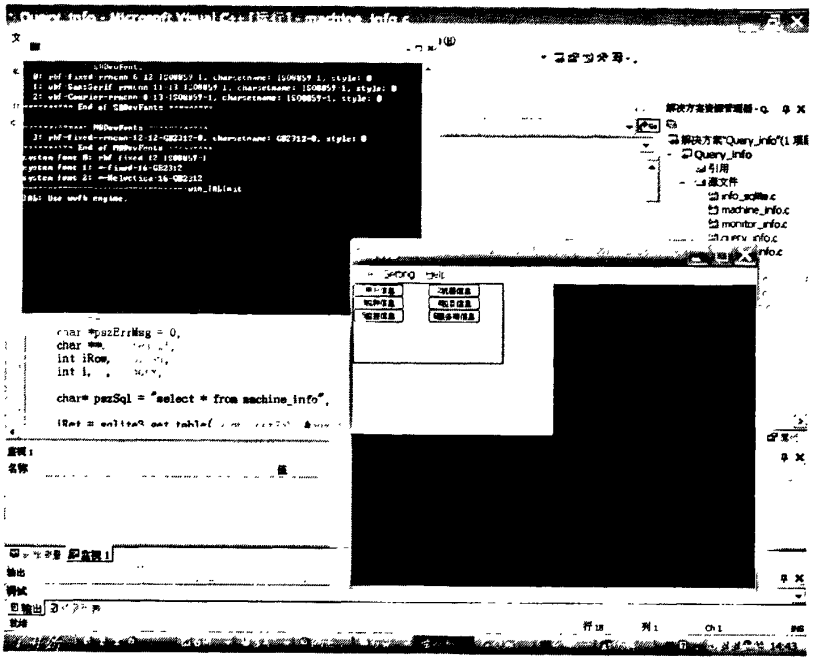

图 6-2 在 Windows 下调试 MiniGUI 的界面

SQlite 的查看可以使用 SQLite Database Browser.exe，通过这个工具，不仅可以查看数据库的运行情况，还可以直接执行 SQL 语句查看语句的正误。运行界面如图 6-3 所示：

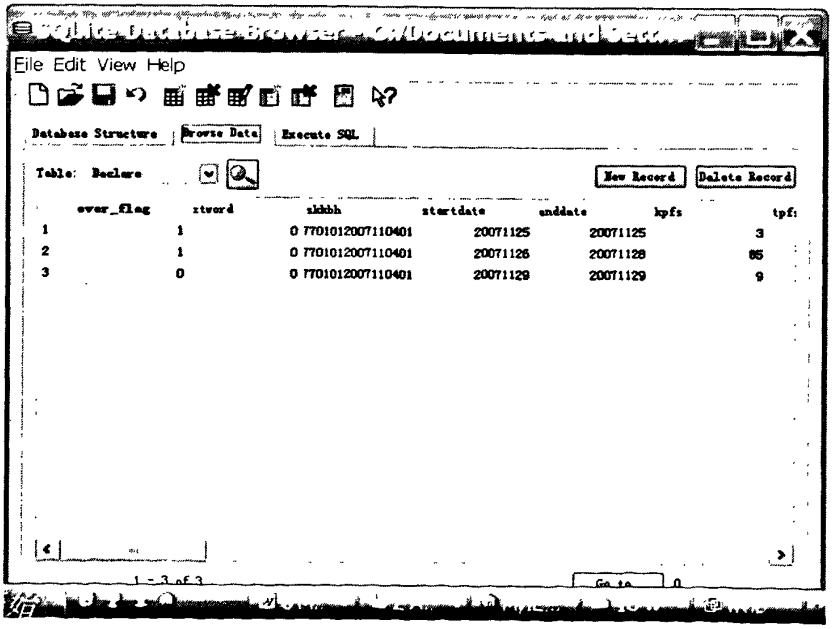

图 6-3 SQLite 调试工具运行界面

利用上述的 Windows 的调试平台，可以进行单步、断点和跟踪等方法查看程序的运行结果，同时可以同步查看数据库的更新情况，大大提高了调试效率。

6.2 调试中遇到的问题及解决方法

一、数据库的存储格式问题

SQlite 数据库中的任何列都可以存储任何类型的数据，sqlite3_get_table 调用了 strlen 函数，导致遇到'\0'有可能切断二进制数据。而税控数据中却存在一些 ASC 码为 0x00 的真实数据，这样就导致了 0x00 后面的数据被自动切断，数据出现错误。对于这个问题我们采用的措施主要是转换。因为这种特殊数据出现的频率非常有限，一般是在税控卡中固定的数据，因此，在保存到数据库时遇到这种数据都将其 asc 码自动加 48 转换，读取时减 48，保证了数据存储的正确性。当然，还是尽量避免用 sqlite3_get_table 这个函数。

SQlite 内部采用 UTF8 存储，但是为转成 GB2312 就必须调用编码函数，非常复杂。后来通过查询网上有关资料[34]，可以将字段谁为 BLOB，然后保存汉字的 GB2312 编码的字节数组。但是这种方式无法用等号查询，可以用“like”替换“=”即可查询成功。

除上面所述的格式之外，数据库存储字符串以'\0'结尾，所以定义缓冲区的时候不能忘了比实际字符多定义一位。

二、内存分配问题

SQlite 中几个函数必须是成对出现的，我们使用的主要是下面两对：

1. sqlite3_open 和 sqlite3_close，sqlite3_open 是打开一个数据库，如果数据库不存在，则创建一个新的；sqlite3_close 关闭数据；

2. sqlite3_get_table 和 sqlite3_free_table，sqlite3_get_table 可以得到行数、列数和数据集，sqlite3_free_table 释放数据集的内存空间；

如果调用了前面的函数，后面必须成对出现相应函数，在程序编写过程中，有地方忘记了释放数据集空间，所以这些空间就一直占用，导致内存泄漏。相比 PC 机而言，嵌入式内存资源非常有限，最终会导致程序的崩溃。同样使用 malloc 函数忘记 free 也会出现这种问题，所以在内存分配问题上一定要谨慎。

三、动态链接库与静态链接库的区别

由于在 Windows 环境下调试调用的是动态链接库，所以一个工程下不同文件之间的函数调用需要用__declspec(dllexport)导出，而在 linux 下交叉编译使用的是静态链接库，可以直接调用，需要将导出函数屏蔽。

四、向 IC 卡发送数据问题

在向 IC 发送命令的过程中，数据正确的命令发送会失败，不停重发。后来经过检查发现，由于串行通信协议的规定[35]，向 IC 卡发送数据一次不能超过 255 个字节，所以对超过 255 字节的命令就需要应用程序来处理。对此，我采取了多次发送的方式，每次最多发送 255 个字节。

五、销售过程中的异常

销售是税控机重要功能之一，稳定性要求非常高，所以必须考虑到各种情况的异常。根据国标规定，在开票之前要判断限额，判断项目个数，当单张发票超过单张限额或单张发票项目超过 6 个都需要开多张票处理。这时情况比较复杂，需要考虑到掉电保护，发票用完，以及打印机缺纸等各种情况，通过不停地调试完善，此部分功能已经比较完善。

6.3 本章小结

软件调试也是软件开发过程中的一个重要环节，本章讲述了应用软件的调试过程，首先介绍了嵌入式软件的几种调试方法，重点说明了Windows平台下的调试。然后回忆了调试中遇到的问题以及解决方法。

结束语

一、工作总结

本文在国家税控机有关标准的基础上，仔细研究了目前税控收款机的不足与缺陷，详细分析了新一代税控收款机的系统需求。根据需求对新一代税控收款机的系统软件和应用软件进行了详细设计，经过精心设计和反复调试，目前样机已完成，在文中本人主要完成的工作如下：

1. 通过分析和比较确定了嵌入式图形界面和数据库的设计方案并完成了MiniGUI和SQLite的移植工作；
2. 利用菜单的思想和MiniGUI开源工具设计了人性化的图形界面，操作简单，界面友好；
3. 利用SQLite完成了一个嵌入式数据库管理系统的设计与实现；
4. 在税控收款机功能需求的基础上采用模块化的方式设计了应用软件，能够完全按照国标的要求完成税控机应有的销售、开票、申报、完税、稽查等所有功能；
5. 完成整个应用程序的编写，并能成功移植到μCLinux操作系统上。

二、展望

针对目前此系统的研究现状，提出如下几点展望：

1. 由于课题的设计需求和研发时间的有关限制，本次设计中仅仅完成了相比超市版功能相对简单的地税版税控收款机，为了对整个系统的开发流程更加熟悉，可在后续时间里继续完成超市版税控机；
2. 作为产品来说，成本是税控收款机推行的一个关键。在本次设计中采用的MiniGUI图形界面遵循GPL条款，商业化需要付费。所以在后续的研发过程中，尽可能自行开发图形界面；
3. 到现在为止，国家的网上报税系统的标准还没有出台，但是为了满足市场需要，网络报税系统的实现已经势在必行，所以，可以结合有关功能需求，在留有开放空间的基础上实现网络报税[36]；
4. 软件方面还需要继续进行优化，降低程序复杂度，提高运行效率，同时安全性考虑方面还有一些做得不足之处；
5. 提高税控收款机的可靠性是税控机发展一直要强调的关键，在本次设计中对能够考虑到的异常常情况都做了处理，例如断电保护，非法操作等等，但是作为以数据为核心的税控机来讲，还需要进一步提高可靠性才能有效防止偷税漏税。

致 谢

本文是在章国宝教授的精心指导下完成的，从论文的选题到方案的设计，以及论文撰写中，每一细节都离不开导师的精心指导。在攻读硕士学位期间，章老师为我创造了优良的研究环境，给予了我悉心的指导和亲切的关怀。章老师学识渊博、治学严谨、思维敏捷，他丰富的科研经验和灵活求实的学术研究方法使我受益非浅，同时章老师幽默诙谐、积极乐观的人生态度及和蔼亲切、平易近人的待人处事态度令我终生难忘。在此，谨向章老师致以最崇高的敬意和衷心的感谢！

在我攻读硕士学位期间，也得到了叶桦教授不断的鼓励和指导，在学习和工作上获得了很多启发性的意见。在此，我也要表示衷心的感谢。

同时，我的师兄师姐以及本实验室的同学，仰燕兰、兰天、陈磊、孟维峰、黄永明、徐英欣、李会娟、宋晓辉、丁昊、孙晓洁、何勇、陈萌、方媛、迟程、荀超、赵辰、宋清华、严芝芳等，在学习、生活和工作中都给予了我很大支持和帮助，在此表示衷心的感谢！

另外，深深感谢敬爱的父母和我的男友对我的支持和鼓励，他们给了我无微不至的关怀，使我能全力以赴地投入研究生的学习与工作，我将继续努力，不辜负他们的殷切期望！

最后，衷心感谢各位专家、老师百忙之中耐心地审阅我的论文，并提出宝贵意见。

参考文献

[1] 周培源,童敏,穆安臻.大型百货、超市业税控方案研究[J].金卡工程,2006:38-40.

[2] 李晓萌,章国宝.基于 S3C2410 控制器的金融税控机的设计与实现[J].南昌大学学报(理科版),2007,31:84.

[3] 赵玮,龚建军,刘仁.税控收款机的应用现状和发展[J].甘肃科技纵横,2004:56-63.

[4] 亿利,金瑞卿.国内税控收款机市场回顾与展望[J],电脑应用,2001:12-13.

[5] 佚名.航信AAH-1000税控收款机[EB/OL], http://www.chinesetax.net/index/SingleInfoShow.asp?InfoID=21274,2006.

[6] 佚名.税控收款机的总体构成[EB/OL], http://www.jmcn.net/web/fcr/skjj/06.asp,2005.

[7] 魏建苗、李莹.基于ARM微处理器和μCLinux的税控收款机的设计与实现[J].工业控制计算机,2005:56-59.

[8] 陆枫.基于嵌入式Linux的税控收款机设计及实现[D]：硕士学位论文.武汉：武汉科技大学,2006.

[9] 王培青.AT91RM9200 在嵌入式税控 POS 系统中的应用[J].维普资讯,2006:5-8.

[10] 李善平,刘文峰,王焕龙等.Linux 与嵌入式系统[M].第 2 版.北京：清华大学出版社,2006:90-113.

[11] 贺安坤,陈明,郝红旗,朱庆峰.税控收款机嵌入式系统的设计与实现[J].计算机工程与应用,2006:89-91.

[12] 胡德鹏.一种税控收款机体系结构及若干技术研究[D]：硕士学位论文.长沙：湖南大学,2007.

[13] MiniGUI 技术白皮书[S].北京飞漫技术有限公司,2005:50-88.

[14] 王震川,短信统计分析系统软件设计与实现[D]：硕士学位论文.北京：北京邮电大学,2006.

[15] MiniGUI 用户手册[S].北京飞漫技术有限公司,2005:15-21.

[16] 周立功等编著.ARM 嵌入式 MiniGUI 初步与应用开发范例[M].北京：北京航空航天大学出版社,2006:144-203.

[17] MiniGUI 编程指南[S].北京飞漫技术有限公司,2005:50-85.

[18] 闫玉忠,石理,嵌入式 Linux 的 MiniGUI 研究和移植[J],西安理工大学学报,2003:5-23.

[19] 周立功.ARM 嵌入式系统实验教程（二）[M].北京：北京航空航天大学出版社,2004:20-50.

[20] wang8012,MiniGUI V1.3.3 在 uCLinux 中的移植经验总结[EB/OL], http://www.minigui.org/cgi-bin/lb5000/topic.cgi?forum=6&topic=3965&show=0,2006.

[21] 魏永明.实时嵌入式 Linux 系统上 GUI 的发展与展望[J].微电脑世界,2000:11-13.

[22] Michele Petrovsky,Stephen Wysham 等著.Linux 数据库宝典[M].耿岳,赵友兵等译.北京：电子工业出版社,2002:314-323.

[23] 张媛媛.嵌入式数据库管理系统的研究与实现[D]：硕士学位论文.上海：华东师范大学,2007.

[24] Michael Owens.The Definitive Guide to SQLite[M].Apress,2006:20-85.

[25] 万俊.智能手机中基于 SQLite 的媒体查询功能的研究与实现[D]：硕士学位论文.南京：南京理工大学,2006.

[26] 天极 yesky.uCLinux 下 SQLite 嵌入式数据库移植全攻略 [EB/OL], http://tech.sina.com.cn/other/2004-12-17/0840478375.shtml.

[27] 林轩.SQLite3 C/C++开发接口简介（API函数）[EB/OL], http://www.sqlite.com.cn/MySqlite/5/251.Html,2006.

[28] 吴澄.基于嵌入式系统的税控收款机设计与实现[D]：硕士学位论文.成都：电子科技大学,2007.

[29] 中华人民共和国国家质量监督检验检疫总局.GB18240.1-2003税控收款机第1部分：机器规范[S].

[30] 陈秋梅,基于 Arm9+Linux 的税控收款机的研究和设计[D]：硕士学位论文.成都：电子科技大学,2007.

[31] 郑灵翔等编著.嵌入式系统设计与应用开发[M].北京：北京航空航天大学出版社,2006:50-64.

[32] 周立功,陈明计,陈渝.ARM 嵌入式 Linux 系统构建与驱动开发范例[M].北京：北京航空航天大学出

版社,2006：42-52.

[33] 贾明,严世贤.Linux 下的 C 编程[M].北京：人民邮电出版社,2001:55-88.

[34] tamsyn.让SQlite以GB2312编码存储汉字[EB/OL],
http://www.sqlite.com.cn/MySqlite/6/406.Html,2007.

[35] 中华人民共和国国家质量监督检验检疫总局.GB18240.1-2003 税控收款机第 2 部分：税控 IC 卡规范[S].

[36] 方立成.嵌入式网上报税系统[D]：硕士学位论文.济南：山东大学,2005.

作者在攻读硕士学位期间发表的论文

第一作者．基于 S3C2410 控制器的金融税控机的设计与实现．南昌大学学报・理科版，2007 专辑，第 31 卷：84-88.