摘 要

微芯片之父 Roland Moreno 于 1974 年 3 月申请了智能卡的专利以来，智能卡现已广泛应用到公安、医疗、交通、金融、电信、社保、石油、体育等各个行业中，并且随着电信行业中 3G 通信的发展，对智能卡的需求将进一步扩大。随着智能卡应用的推广，其产业链也得到极大的完善和细分。整个产业链从智能卡芯片设计、硅晶片的生产制造、芯片封装、智能卡操作系统研发、灌卡、个人化、发行、最终到用户使用以及系统升级和维护。本文所描述的内容主要是智能卡产业链中的一个重要环节：智能卡操作系统研发，也就是我们所说的 COS 开发。

本文讲述 COS 开发中 3 个主要问题：智能卡 COS 的系统结构、COS 系统中硬件驱动模块设计和绑定式 COS 研究。文章从描述智能卡系统开发的研究背景、选题意义和项目背景等内容开始；介绍了智能卡的物理特性，包括了：智能卡的分类、外部通信接口和内部逻辑结构；总结出智能卡操作系统的研究现状和今后的发展趋势，智能卡开发所需要遵循的国际标准体系，讲述了智能卡 COS 的基本功能，工作原理及一般系统结构等内容；从这基础上提出本文智能卡 COS 设计目标、原则以及本文所设计的系统结构；并且完成 A 公司芯片硬件驱动模块开发设计，阐述了其中的主控函数设计、ATR 编码含义、PPS 协商、硬件驱动函数和掉电保护设计。并在 A 芯片开发完成基础上把 COS 移植到 B 芯片过程中，提出了绑定式思想，并阐述了统一接口设计，驱动函数库管理和 COS 生成器设计。论文最后阐述了课题涉及创新点和取得的成果。

本文所涉及的课题是广东工业大学与中国通信服务股份有限公司广东公司之间的横向合作项目，该项目于 2007 年 7 月 1 日开始进行分析设计，到目前为止各模块设计已经完成，并能实现电话、短信和菜单应用等功能；并成功完成从 A 芯片到 B 芯片的移植。本文正是在这个项目实践的基础上编写的。

关键字：智能卡；COS；绑定式；硬件驱动；掉电保护。

ABSTRACT

Roland Moreno, the Microchip Father, applied for the smart card patent in March 1974. And the smart card has been widely applied to public security、medical care、transport、finance、telecommunications、social security、oil、sports and so on. As the 3G communications telecom industry expanding, the development of smart card would be developing rapidly in the further. And the industry chain of the smart card which includes many tache: IC chip design, chip manufacture, IC COS R&D, IC individuation, system upgrades and maintenance and so on has been greatly perfect and fractionized. The contents expatiated in this paper is an important tache in the chain of the smart card: IC COS designment, that is , we call COS design in general.

This article focuses on three main issues which in the smart card COS design: COS system architecture, hardware driver module design and Bind-COS research. Firstly, this paper expatiates this problem' s researching background、item background and the selecting significance etc; Next, introduces the smart card physical characteristic, including: the classification of the smart card, external communication interface and internal structure; Summes up the smart card operating system research actuality and trends, the ISO criterion and telecom industry criterion to smart card; Describes the basic functions of the smart card COS, working principle and the general system architecture and so on; On this basic, proposed the design objectives and principles of smart COS, designed the new system structure; And finish the design of hardware driver modules of A company chip, including: main program design、ATR response、PPS consult、hardware driver program and power-down protection design; When the completion of design and transplanting the COS from A chip to B, proposed the new idea——bind-COS design: Describes the designment of common interface layer、driver-base and COS generator; Finally, this paper

summarizes innovation and harvest.

This paper' s project is a transverse item, which is cooperated with China Telecom Service .Ltd xx smart card .ltd by Guangdong University of technology. The project has been design from 2007.7. Up to now, all the modular design has been completed, and can realize telephone, message and application menu functions; And successfully complete the transition from A chip to B, This paper is write on the basis of this item.

Keyword: Smart Card; COS; Bind-COS; Hardware-driven; Power-down protection

独创性声明

秉承学校严谨的学风与优良的科学道德，本人声明所呈交的论文是我个人在导师的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，论文中不包括其他人已经发表或撰写过的研究成果，不包含本人或其他用途使用过得成果。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声明，并表示了谢意。

本学位论文成果是本人在广东工业大学读书期间在导师的指导下取得的，论文成果归广东工业大学所有。

申请学位论文与资料若有不实之处，本人承担一切相关责任，特此声明。

指导教师签字：

论文作者签字：

2009 年 5 月 20 日

第一章 绪论

1.1 研究背景

1974年3月，微芯片之父者罗兰·莫雷诺（Roland Moreno）申请了智能卡的专利；1976年，法国布尔(BULL)公司首先开发出智能卡产品；1987年，国际标准化组织ISO专门为IC卡制订了国际标准，ISO/IEC7816-1，2，3，4，5，6，7，8，9等；1993年，我国启动“金卡工程”，智能卡正式在我国得到广泛应用。

智能卡将微电子技术、计算机技术以及信息安全尤其是密码学技术结合在一起，提高了人们生活和工作的安全程度。由于智能卡具有较强的安全性，是用户身份认证实现的良好载体，因此，自20世纪70年代IC卡问世以来发展迅速。2001年，我国总计发放智能卡3.2亿张；2002年其发行量增长20%；2006年，中国IC卡的出货量与出货总价分别达到16.76亿张和74.26亿元。2007年，中国IC卡的出货量突破20亿张大关[1]。如图1-1所示：

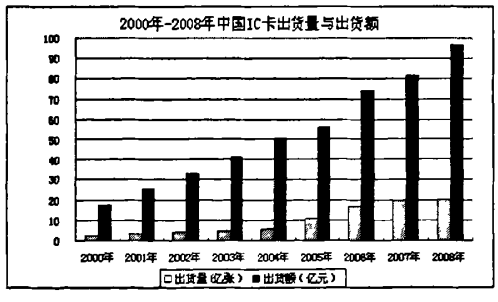


图 1-1 2000－2008 年中国 IC 卡出货量与出货额图

Figure1-1 2000-2008 IC Card Shipment Volume And Amounts In China

2007年全球IC卡出货达到了26亿张，比2006年增长27%，其中支持3G的SIM卡已经占到了14%。全球最大的网络运营商中国移动在2007年前三季度订货超过2.8亿张，比上年同期增长60%。2007年全年中国出货量6.5亿张移动电话卡[1]。如图1-2所示：

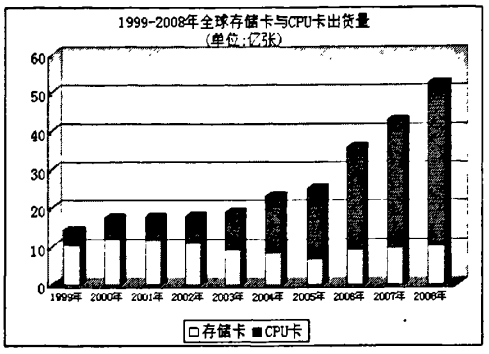


图 1-2 1999－2008 年全球存储卡与 CPU 卡出货量图

Figure1-2 1999-2008 Memory Card And CPU Card Shipments Map In Global

智能卡的迅猛发展离不开电信行业的推动作用。电信行业智能卡的使用包含2G系统的移动SIM卡、联通UIM卡、电信PIM卡，以及3G系统的移动USIM卡，电信R-UIM卡，电信行业智能卡应用是目前最大的智能卡行业应用，随着3G通信的发展，电信行业对智能卡的需求将进一步扩大。2008年4月1日，中国移动将在包括广州、深圳等8个城市启动3G社会化业务测试和试商用，还将同步向公众市场放号销售。2009年1月7日，工业和信息化部确认国内3G牌照发放给三家营运商，中国移动获得TD-SCDMA牌照，中国电信获得CDMA2000牌照，而中国联通获得WCDMA牌照。 2009年我国正式进入第三代移动通信时代。此外，中国移动也将在2011年把3G网络覆盖到100%的国内地市。

1.2 选题意义

智能卡系统应用是一个跨多学科的应用系统，它将大量来自不同专业领域的技术综合在一起，诸如计算机技术、网络技术，数据库处理技术、高频技术、电磁兼容性、半导体技术、数据保护和密码学等。智能卡由硬件资源(智能卡芯片)与芯片操作系统 COS(Chip Operating System)组成，COS 是智能卡的核心。

传统 COS 是针对某一种特定芯片开发的，事实上，这种操作系统与嵌入在芯片的应用程序别无二致。该操作系统通常由芯片生产商自己来开发，因为只有他们才真正了解芯片底层的技术细节(大多数厂商对自己的芯片技术都是保密的)。每个厂家芯片 COS 开发的环境不一样，在 COS 支持的上层应用不变的情况下，当更换不同的硬件时，需要重新了解新硬件 COS 的开发环境，新硬件底层的技术细节，重新将上层逻辑应用移植到新的硬件上，移植的工作量非常巨大，不低于重新编写一次 COS，移植过后 COS 的稳定性也需要重新测试，这样极大制约了智能卡应用的快速

发展。另外智能卡COS目前使用的开放式系统研发环境，如Javacard等，存在较高的版权费用，软件成本偏高；使用自然语言开发的COS，目前大多使用层次结构，效率较低，编写的代码量较大，降低了硬件的效率和增加了存储成本。

针对上述部分问题，在COS设计研发过程中，采用了模型改进、软件构造方法改进的策略。为了提高COS的效率，改进原有的层次模型，采用半层次模型，以提高COS的运行效率；软件构造方法上，我们分析现有各个厂家芯片底层信息，对各芯片形成共享驱动组件，采用动态构造编译方法设计了一种支持多芯片的COS，当更换硬件时，不需要重新移植，只需要重新选择相应硬件的驱动组建，重新编译构成新的COS。这是本课题研究的重要意义。

1.3 项目背景

本文作者在硕士研究生期间，主要从事“嵌入式系统和智能卡技术研究”，期间研究分析了ISO7816系列的智能卡通用标准，及2.5G、3G体系标准与USIM卡、UIM卡相关的标准，本课题研究内容属于广东工业大学与中国通信服务有限公司广东公司的下属公司XX智能卡公司合作的横向项目。本项目以当前智能卡在各行业广泛应用和第三代移动通信系统商用网络即将出炉为背景，以智能卡安全技术和文件系统技术为核心，在第二代移动通信系统UIM卡的基础上，以区别于JavaCard、Window for Sart Card、MULTOS等通用系统方式，采用自然语言方式研究设计了UTKCOS的系统结构。

中国通信服务有限公司广东公司的下属公司XX智能卡公司在智能卡行业具有较强的生产能力，但该公司在COS上的技术研发偏弱，该公司生产的UIM卡和USIM卡均由芯片公司提供付版权费COS，这样导致XX智能卡公司的每张卡的成本增加，企业竞争力下降。本横向项目恰好结合了学校的嵌入式研发技术力量和XX智能卡公司较强的生产力量，使企业和学校之间达到了双赢。

本项目的开展到撰写本文为止已有1年6个月时间，目前完成了UTKCOS的结构设计，智能卡硬件分析、安全管理模块、命令处理模块、文件系统模块设计和代码编写，智能卡底层硬件驱动代码编写，整个操作系统基本完成；并实现发短信和通话功能。本文主要在这项目的设计和实践基础上，进行了分析和总结而编写的。

1.4 论文结构安排

本文从比较通用的IC开始，介绍通用IC卡的现状、COS情况；接着介绍了本文

需要重点讲述的一种特殊的智能卡：UTKUIM卡。具体的细节参见各章中的描述：

第一章：绪论，在本章中本文重点描述了本文所选课题的研究背景、项目背景、选题意义等内容。

第二章：智能卡物理特性主要阐述智能卡的硬件结构，分别介绍了：智能卡分类，智能卡的外部通信触点和智能卡内部结构、并介绍开发芯片的物理性能及其Flash扩展能力。

第三章：智能卡芯片操作系统主要介绍智能卡软件方面，包括了智能卡COS研究现状及发展趋势、智能卡遵循的国际标准体系、智能卡COS的基本功能及一般系统结构、并在这基础提出本文设计COS的设计目标和原则并提出新的系统结构。

第四章：COS硬件驱动设计介绍了COS硬件驱动模块的研发思路，包括主控函数设计、ATR编码含义、PPS协商过程、驱动函数设计和掉电保护设计。

第五章：绑定式设计，它是基于快速COS移植提出的。主要包括了统一接口层的设计、另一公司的硬件驱动函数开发、驱动库设计和COS生成器设计。

第六章：课题涉及创新点和取得的成果。

第二章 智能卡物理特性

智能卡的名称来源于英文名词“Smart Card”，又称集成电路卡(Integrated Circuit Card)，即智能卡。它将一个集成电路芯片镶嵌于塑料基片中，封装成为卡的形式。

2.1 智能卡分类

随着超大规模集成电路技术、计算机技术和信息安全技术等的发展，智能卡技术也更成熟，并获得更为广泛的应用,智能卡的分类因此也变得复杂起来，根据分类标准的不同，智能卡具有不同的分类。智能卡一般分类的标准具有结构体系、数据获取方式、行业应用等方式进行划分。

按智能卡的结构划分,智能卡可分为：存储卡(Memory Card)，CPU 卡(Smart Card/ IC 卡)。CPU 卡是由一个或多个集成电路芯片组成，并封装成便于人们携带的卡片,在集成电路中具有 MCU 和存储器,智能卡具有暂时或永久的数据存储能力，其内容可供外部读取或供内部处理和判断之用，同时还具有逻辑处理功能，用于识别和响应外部提供的信息和芯片本身判定路线和指令执行的逻辑功能，我们目前使用的二代身份证和 SIM 卡都是 CPU 卡的典型代表；存储卡(Memory Card)的定义是由一个或多个集成电路芯片组成，并封装成便于人们携带的卡片，具有记忆存储功能，不带 CPU，存储卡目前在公交、一卡通、消费电子等行业广泛应用。

智能卡按卡片数据的读写方式分为：接触式智能卡、非接触式卡、双界面卡。接触式智能卡由读写设备的触点和卡片上的触点相接触，进行数据读写；国际标准 ISO7816 系列对此类 IC 卡进行了规定；接触式卡片应用的典型是手机中的 SIM 卡。非接触式卡，则与读写设备无电路接触，由非接触式的读写技术进行读写（例如，光或无线电技术）；其芯片除了内嵌存储单元、控制逻辑外，增加了射频收发电路。这类卡一般用在存取频繁，可靠性要求特别高的场合，国际标准 ISO10536 系列阐述了对非接触式 IC 卡的有关规定。非接触式智能卡典型代表如目前广泛使用的公交卡、校园卡、各种 RFID 电子标签等。

双界面卡主要是针对目前的接触式和非接触式智能卡在获取数据方面均有各种缺憾而产生的，双界面结合了两者的优势，更为准确的应该叫双接口卡(Double

Interface Card)，因为实际上它在一张卡片上同时提供了两种与外界接口的方式：接触式和非接触式。双界面卡的外形与接触式 IC 卡相象，表面符合国际标准的金属触点。内部结构则与非接触式 IC 卡相似，有天线和芯片的模块。在双界面卡的具体实现上，有三种结构类型：1)把接触式智能卡芯片和非接触式逻辑加密芯片加上天线封装在一张卡中，构成一张双界面卡。接触式和非接触式系统的运行分别由两个独立的芯片控制，卡内有两个独立的 EEPROM 存储器，两套系统互相独立，这实际上是将两块芯片封装在一张卡上。2)由一个芯片和天线构成，它具有非接触式逻辑加密卡功能、接触式 CPU 卡的功能，两个系统共用 EEPROM 存储器。3)由一个芯片和天线构成，共用芯片内的 EEPROM 存储器、微处理器、ROM、RAM 等资源,控制接触式与非接触式系统的运行。三种双界面智能卡中，只有最后一种才称得上是真正意义的双界面 CPU 卡，也是集成电路设计公司将来开发的重点所在。

智能卡按行业应用分差别比较大，因为不同行业的卡片差别巨大。目前使用智能卡应用比较广泛的行业，首先是电信行业，其次是金融、医疗、社保、石油等行业。在电信行业常见的智能卡有 SIM 卡、UIM 卡、PIM 卡，还有即将发行的 3G USIM 卡、RUIM 卡等。目前我国电信行业总发卡量已超过 20 多亿张，占我国 IC 卡发行总量的 75%左右。2006 年中国联通电信智能卡发行量超过了 1.27 亿张，中国移动发行量约 4 亿张。

智能卡另外按卡中集成电路不同可分为：非加密存储器卡、加密存储器卡；按照数据交换格式分类，智能卡可以分为串行和并行两种。

本项目开发的智能卡 COS 是应用在 CDMA 网络中，带 UTK 应用菜单功能的 UIM 卡操作系统。用到的是带加密逻辑电路的接触式 CPU 卡。

2.2 智能卡的外部通信触点

在实际使用中，有两种功能相同而形式不同的智能卡：卡片式(俗称大卡)和嵌入式(俗称小卡)。

“大卡”上真正起作用的是它上面的那张“小卡”，而“小卡”上起作用的部分则是卡面上的铜制接口及其内部胶封的卡内逻辑电路。目前国内流行样式是“小卡”，其大小只有 25mm×15mm，小卡也可以换成“大卡”(需加装一卡托)。“大卡”和“小卡”分别适用于不同类型的 GSM 移动电话，早期机型如摩托罗拉 GC87C、308C 等手机用的是“大卡”，而目前新出的机型基本上都使用“小卡”。

嵌入式智能卡有八个触点，通常与移动设备连接需要六个触点，具体接口定义如图 2-1 所示：

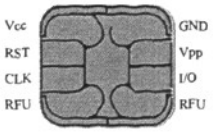


图 2-1 智能卡触点示意图
Figure2-1 Smart Card Touch Interface

VCC：电源电压； RST：复位信号；CLK：时钟触脚；GND：参考地；VPP：编程电压；I/O：I/O 数据；RFU：保留未用。

2.3 智能卡的内部结构

无论是哪个厂商哪个系列的产品，智能卡芯片的逻辑结构都基本类似，如图 2-2 所示是一个典型的芯片硬件逻辑结构示例。通常包括：I/O 通信接口、微处理器 CPU、加密协处理器 CAU、存储单元、储存器管理单元 MMU、定时器、随机数发生器、硬件安全保护等。智能卡一般的内部结构如图 2-2 所示：

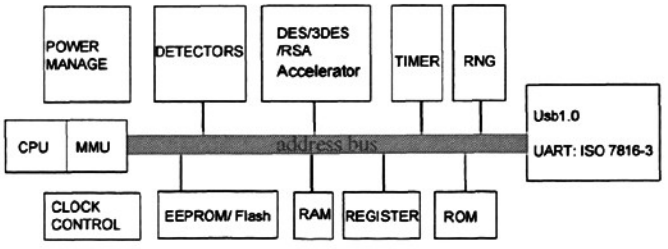


图 2-2 智能卡结构图
Figure2-2 Smart Card Frame

1）I/O 接口：I/O 接口是芯片与外界联系的唯一通道，也是芯片中标准化程度最高的模块，所有的芯片都按 ISO－7816 的串行异步通信标准设计 I/O 接口。在传输协议中定义了两种协议：字符传输协议（T＝0）和块传输协议（T=1）。终端一般都支持者两种协议，但目前智能卡使用基本使用 T=0 协议。

2）智能卡 CPU：是智能卡芯片的核心，在功能上类似于 PC 机上的 CPU。完成基本的指令执行、存储控制和逻辑控制等。目前国产芯片厂商（如华大、华虹公司）提供的智能卡，CPU 主要是 8 位字长，通常其核心电路为 Motorola6805 或者 Intel8051 内核。而国外芯片厂商（如三星）以 16 位 CPU 芯片为主，并且 32 位芯片已经出现并在部分高端产品中应用。

3）加密协处理器 CAU：除了 CPU 提供的基本运算功能外，为了支持安全应用

中的加密运算，一些中高端的芯片提供一个常用算法的协处理器（Cipher Arithmetic Unit，简称 CAU），例如奇偶校验、DES、MAC 计算等。在这些算法中，奇偶校验主要用于 I/O 通信协议，除此之外，DES 算法协处理器最为常见。一般的，如果采用软件实现 DES 算法，一方面可能消耗很宝贵的卡片代码存储空间，另一方面，软件实现比硬件实现大约要慢 100 倍左右，无疑也极大降低了卡片的应用响应速度，DES 算法协处理器目前也基本成为了中高端芯片的标准配置。为了实现高级算法的应用，部分芯片厂商甚至配备 T-DES 协处理器或者 AES 协处理器。而是否包含协处理器一般是影响智能卡的成本和价格主要因素。

4）数据存储单元 ROM：是卡片内的只读存储区，COS 代码和一些基本用于存储在这里。在现在电信卡中，由于成本原因都是应用 FLASH 来存储数据和程序。

RAM（Random Access Memory）：类似于 PC 上的内存，是卡片使用阶段的临时数据空间，在卡片每次复位时自动清零，掉电以后数据也全部丢失，所以只能用来存储一些中间数据。

5）储存器管理单元 MMU：MMU 由一个或一组芯片组成，其功能是把逻辑地址映射为物理地址（即进行地址转换）；还能够扩大寻址空间，辅助 CPU 进行内存读写数据。并不是每个厂商 MMU 模块功能都相同，在一些厂商（如英飞凌）的芯片 MMU 完成代替 CPU 进行存储器管理数据，大大减少 CPU 的负担。

6）定时器：产生外部时钟计算。芯片不一样，定时器的个数和位数也不一样。普遍的是两个 16 位的定时器。

7）随机数发生器：基本上每个芯片都有随机数发生器。主要用于系统加密解密。

8）电压：根据 ISO－7816 标准，SIM 卡的供电分为 5V、5V 与 3V 兼容、3V、1.8V 等，当然这些卡必须与相应的移动电话机配合使用，即终端供电电压与该 SIM 卡所需的电压相匹配。

9）硬件安全保护：硬件安全保护机制一般是芯片提供一些安全扩展功能集。厂商根据不同应用要求提供各种硬件安全包括。包括：硬件抗攻击保护、内存访问控制机制。

硬件抗攻击保护为了防止外界对芯片的物理攻击，芯片需要有自动的电压、频率、光照等检查手段，当出现异常情况时自动对卡片进行复位或者销毁数据等保护措施。

内存访问控制机制是指，对于存储器的权限控制通过软件一般无法实现，在多应用环境中又是必不可少的。硬件的访问控制一般通过存取空间分区、分区属性字来进行。

2.4 开发芯片的物理结构

本文设计的 COS 选用了 A 有限公司的 A112F 型号芯片进行开发(因为与芯片公司签订保密协议，文中开发的芯片公司名称或型号均以 A、B 表示。如 A112F，表示 A 公司，112kFlash 型号芯片)。芯片基于标准的 8051 架构，具有 112 KB FLASH，2KB + 256 B RAM，专有高性能 CPU，DES、CRC 协处理器，UART，真随机数发生器，安全检测单元。结构如图 2-3 所示：

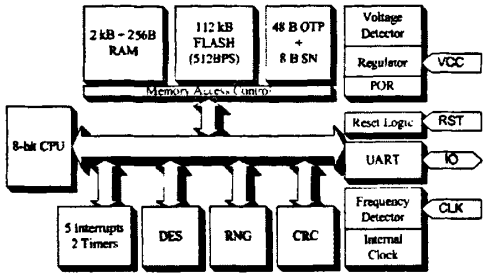


图 2-3 A112F 模块结构
Figure2-3 A112F Module Structure

2.4.1 智能卡性能

- ✧ 高性能 8 位 CPU 内核兼容标准 8051；
- ✧ CPU 工作时钟可以在内外部时钟之间动态切换；
- ✧ 执行 1 条指令需 1 到 3 机器周期，1 个机器周期通常为 4 时钟周期；
- ✧ 存储器： FLASH 112 KB (512 BPS) RAM 2 KB + 256 Bytes OTP 48 Bytes SN 8 Bytes；
- ✧ 支持 CODE Bank 和 XDATA Bank；
- ✧ 2 个 16 位定时器，5 个中断；
- ✧ 数据加密存储；
- ✧ 内嵌 DES、CRC、真随机数发生器；
- ✧ 电气安全探测器，防御高低电压攻击和高低频率攻击；
- ✧ UART 支持 7816-3 T=0 协议，支持 10 种波特率：FD = 11, 12, 13, 18, 91, 92, 93, 94, 95, 96；
- ✧ 支持 GSM 功耗标准，包括 Clock Stop 工况。

2.4.2 寻址能力扩展

1. 8051 标准寻址

智能卡采用标准 8051 内核，所以与 8051 内核兼容，按照寻址方式划分，CPU 可访问 4 个逻辑地址空间。逻辑地址空间如图 2-4 所示：

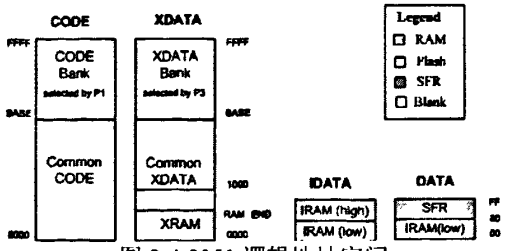


图 2-4 8051 逻辑地址空间

Figure2-4 8051 Logic Address Distributing

2. 对标准8052寻址能力的扩展

标准8051是8位CPU处理器。所以寻址空间最大64KB。为了实现COS代码长度大于64KB，或者数据空间大于64KB，芯片在逻辑空间CODE和XDATA 架构内提供了bank扩展能力。 FLASH物理空间被划分为1个Common Bank（不可切换）和8个bank（可动态切换）。因为bank多而FLASH小，所以部分bank有空白区。Common Bank 是指位于Flash物理空间低端，由0地址起始到可切换bank起始地址之间的Flash物理区域。

8个bank可被软件动态映射到CODE Bank和XDATA Bank。芯片使用寄存器SFR P1配置CODE Bank；寄存器SFR P3配置XDATA Bank。Common CODE是指CODE区的低端，只能固定地看到FLASH Common Bank；Common XDATA是指XDATA区的低端，只能固定地看到RAM和部分FLASH。图2-5是bank大小为48k空间的CODE bank影射图。

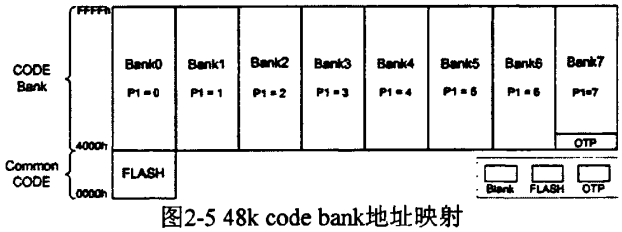


图2-5 48k code bank地址映射

Figure2-5 48k Code Bank Address Mapping

相应的物空间如表3-1所示：

表3-1 48k code bank地址

Table3-1 48k code bank Address

名称	逻辑空间CODE	物理存储器地址	
CODE Bank	CODE[4000h, FFFFh]	P1 = 7	OTP [00h, 6Fh]
		P1 = 6	空白
		P1 = 5	空白
		P1 = 4	空白
		P1 = 3	空白
		P1 = 2	空白
		P1 = 1	FLASH [10000h, 1BFFFh]
		P1 = 0	FLASH [04000h, 0FFFFh]
Common CODE	CODE[0000h, 3FFFh]	FLASH [00000h, 03FFFh]	

图2-6所示为48K XDATA BANK映射图。

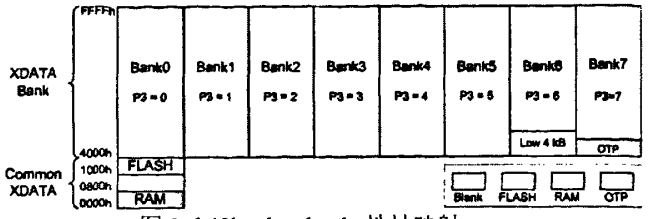


图 2-6 48k xdata bank 地址映射

Figure2-6 48k xdata bank Address Mapping

第三章 智能卡芯片操作系统

智能卡实现某项功能，除了有必要的硬件之外，还需要有与之相关的芯片操作系统 COS(Chip Operating System)。就像计算机一样，没有操作系统的计算机根本无法使用。所以说芯片操作系统 COS 是智能卡的灵魂，智能卡正因为有了 COS 才具有了“智能”。

3.1 智能卡 COS 研究现状及发展趋势

由于智能卡进入我国时间相对比较晚，我国的智能卡操作系统发展与国外比较，还是处于比较落后的情况。

3.1.1 国外智能卡操作系统发展

在智能卡操作系统领域，主要有开放式系统和私有操作系统。

最近几年，开放式操作系统平台取得了重大发展，这大大方便了智能卡的应用开发和一卡多用的实现，并且允许动态地装载、更新或删除卡片应用。现在开放式系统国外较有代表性的包括：

由Mondex公司提出，以MasterCard为首的八家厂商共同推出的MULTOS。MULTOS是高安全、多应用的智能卡操作系统标准，支持 EMV、身份证与其他多种应用。卡片有效生命周期内允许动态地装载、更新或删除卡片应用。另外，它本身固有的安全性和方便性，使其成为目前公认的网络安全用户端解决方案。利用智能卡可以较方便通过数据加密及通过 PKI 进行身份验证，保证在线安全支付。现在 MULTOS 主要应用于金融领域。

SUN 公司基于 Java 的特点提出了一套应用于智能卡的框架，将 Java 虚拟机移植到智能卡芯片中，提出 Java Card 的概念。目前国际上超过 90%的芯片厂家都购买了 Java 版权并推出 Java Card 的产品。Java Card 正在逐步变为未来行业内的一种事实标准。

Java Card 以 Java 虚拟机(JVM)为基础，支持多应用动态下载。Java 智能卡具有平台无关、高安全性、高可靠性、一卡多用的特点，适于开发特定场合、突出个人身份验证、并在用卡交易过程中确保持卡人和证件一致性的应用。

Microsoft 公司基于 Windows 操作系统提出的微软智能卡视窗(Windows

ForSmart Card)的概念，在这个平台基础上进行 COS 的开发，可以简单使用 Microsoft Visual Studio 作为开发工具。微软智能卡视窗(Windows For Smart Card)与微软 Windows 操作系统相结合，将在电子商务、网络安全有广阔前景。[2]

3.1.2 我国智能卡操作系统发展

智能卡应用系统进入我国的时间较晚，1993 年 9 月国家金卡工程正式启动。金卡工程的主要任务是推行电子货币，实现支付手段的革命性变化，为个人和企业提供方便、快捷、安全的支付手段。

由于我国的智能卡产业起步较晚，整体水平不高，对智能卡的需求也处于一个比较低的层次上，技术水平与发达国家相比还有较大的差距。目前我国已有 30 多个厂家从事智能卡的研究和开发生产工作，如北京握奇数据、大唐微电子、华大电子、东信和平等公司。在智能卡操作系统开发领域，较有代表性的是握奇 TimeCOS2.0. 清华同方 ZTCOS 等。作为首批启动的国家信息化重大工程，我国的智能卡系统应用的普及推广还需要广大智能卡科研学者的努力。

开放式系统避免了对底层的繁琐，可以使研发人员集中精力开发上层应用，但一般开放式系统建立在高端芯片(如 16 位或更高的处理器)的基础之上，对芯片的处理器速度、存储空间等都有很高的要求，目前支持这些方案的智能卡芯片都价格高昂，并且开放式系统需要版权费用。这些方面，高昂的软硬件成本制约了智能卡应用的发展。

由于开放式系统 COS 研发的软硬件成本较高，私有操作系统是目前国内厂家普遍采用的方式，指使用自然语言编写的 COS（目前主要是 C 语言），这种方式对硬件的要求较低，不需要版权费用，目前在国内广泛流行。

在整个电信智能卡行业 COS 研发来说，将来的发展趋势应该是开放式系统研发，但就将来一段时间来看，国内 COS 研发，由于芯片目前基本受到国外半导体公司的垄断，开放式系统也受到国外软件公司的垄断，这对国内智能卡厂家来说需要付出极高的软硬件成本。所以，私有系统的研究在国内相当一段时间内还是具有相当高的价值，甚至可形成另一种智能卡的工业标准[2]。

3.2 电信行业智能卡遵循的标准体系

3.2.1 智能卡遵循的标准体系

智能卡与外界交换，必须遵循一定标准。1987 年开始国际标准化组织相继制定

和颁布了CPU卡的国际标准。有关接触式CPU卡本身的标准有[3]:

ISO7816:识别卡一带触点的集成电路卡。

ISO7816-1:规定卡的物理特性。卡的物理特性中描述了卡应达到的防护紫外线的能力 、X光照射的剂量、卡和触点的机械强度、抗电磁干扰能力等等。

ISO7816-2:规定卡的尺寸和触点的位置。

ISO7 816-3:规定卡与接口设备之间的电信号和传输协议。传输协议包括两种:同步传输协议和异步传输协议。

ISO7816-4:规定卡的行业间换用命令。包括:在卡与读写间传送的命令和应答信息内容;在卡中的文件、数据结构及访问方法;定义在卡中的文件和数据访问权限及安全结构。

ISO7816-5:规定应用标识符的编号系统和注册过程。

ISO7816-6:规定行业间数据元素格式。

ISO7816-8:规定有关卡命令的安全属性和机制。

ISO7816-9:卡命令和安全属性的补充规定。

3.2.2 电信行业卡遵循的标准体系

除了7816协议之外，在各个可能应用IC卡的特定领域内还有一些更为具体的协议，比如在中国，金融领域制定PBOC规范，交通管理体系，社会福利体系都有其特定的规范。这些协议规范都是建立在7816协议基础之上，且将7816协议加以具体化形成的。

电信领域的智能卡在ISO7816国际标准之上，针对行业应用特点，由相应国际组织制定电信智能卡标准，包括3GPP、3GPP2和GSM。在项目开发过程中，除了遵循ISO7816之外，还主要遵循了：

GSM11.11 个人身份鉴别模块——移动设备接口规范。

GSM11.14 UIM卡应用工具箱。

中国电信CDMA卡需求规范一UIM卡分册(v1.0)

中国电信CDMA卡测试规范- UIM卡分册(v1.0)

中国电信CDMA卡需求规范-ESN信息上报分册(v1.0)

中国电信CDMA卡需求规范-UTK应用分册(v1.0)

中国电信CDMA卡需求规范-单向全能读分册(v1.0)

中国电信CDMA卡需求规范一超级号簿分册(v1.0)

中国电信 CDMA 卡测试规范－超级号簿分册(v1.0)

3.3 智能卡 COS

和传统的 PC 软件系统相比较,COS 在智能卡系统中的地位类似于 Windows、UNIX 这样的操作系统，所以称之为卡片操作系统。从功能上来说，COS 并不是一个完整意义的操作系统，更类似于一个监控程序。

首先，COS 是一个专用系统而不是通用系统。应用在不同领域的 COS；或者同一领域不同芯片的 COS，他们的实现方式都不一样。因为 COS 一般都是根据某种智能卡的特点及其应用范围而特定设计开发的，尽管它们在所实际完成的功能都遵循着同一个国际标准。

其次，在当前阶段，COS 所需要解决的主要还是对外部的命令如何进行处理、响应的问题，没有涉及共享、并发的管理及处理，而且，就智能卡在目前的应用情况而言，并发和共享的工作也确实是不需要的。甚至从文件系统来看，COS 实现的功能远比通用操作系统简单。

3.3.1 智能卡 COS 的基本功能

COS的主要功能是控制智能卡同外界的信息交换,管理卡内的存储器并在卡内部完成各种命令的执行，管理文件，管理和执行加密算法。总的说来，COS提供的基本功能包括以下几个部分：

芯片底层硬件的驱动：由于COS直接在芯片平台上运行，对芯片各个部件的操作都在COS中完成，包括存储器的读写、加密协处理器的使用、端口的控制、随机数发生器的使用等。

卡片对外接口交互：芯片对外通过I/O端口收发数据，所传递的数据格式必须符合ISO7816-3的标准。

内存空间的维护：卡片的内存空间是程序的数据临时空间，是智能卡内重要的资源之一由于价格等因素，内存空间通常十分有限，COS需要提供相应的保障机制合理分配内存，并且区分保护好系统和应用各自的内存空间，确保安全性。

文件系统的维护：数据在卡内以文件的形式存在，和PC中操作系统的文件系统类似，COS的文件系统必须提供文件的建立、读写、删除、维护等基本操作，此外，还有文件的寻址，文件的访问安全控制等等。

系统运行安全的控制：对于COS的系统安全，主要表现在两个方面，一是COS

对卡内数据的安全保障能力，二是COS对外提供的安全定义。应用应该能够利用COS提供的安全定义自定义出其卡内数据的访问控制。安全控制是COS的核心，也是衡量一个COS性能最重要的部分。

命令的处理：卡片与终端的信息交互采用命令与响应方式进行交互。COS利用I/O接口接收命令，根据命令的内容进行相应的处理，最终通过I/O接口返回响应。命令处理是COS最重要的部分之一，也是最终测试COS正确与否的依据。

3.3.2 智能卡 COS 的一般系统结构

从上文可以看出，COS 是面向应用的针对性开发，针对不同类型的应用，COS 的功能也千差万别。但是，从 COS 的总体结构来分析，不同 COS 所包含的模块以及模块之间的联系关系有很大相似性。所有的 COS 在设计时都需要考虑三个问题，即：文件操作、鉴别与核实、安全机制。鉴别与核实和安全机制都属于智能卡的安全体系的范畴之中。所以，智能卡的 COS 中最重要的两方面就是文件与安全。从指令的流向上看，指令从外部终端开始进入智能卡，一般需要四个过程的处理，通常也可以说是 COS 中的四个功能模块：通信管理模块、安全管理模块、命令处理模块和文件管理模块，整个处理过程如图 3-1 所示：

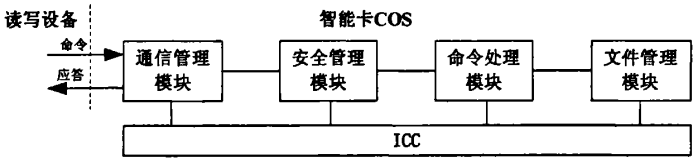


图 3-1 COS 层次模型图

Figure3-1 COS Hiberarchy Model

上图 COS 采用自然语言进行设计时具有的一般模型，大多数 COS 设计采用此模型，此模型具有简单明了的优点。

1．通信管理模块

通信管理有两个主要任务：1、根据所使用的传输协议，发送 ATR（Answer to reset）信息，如通信波特率、通信协议等。2、接收命令和发送响应数据。

ATR 响应：智能卡在终端上，上电后，COS 自动从芯片的某位置开始运行程序，此时 COS 首先向终端发送 ATR 指令，让终端清楚知道卡片支持和目前使用的电源特性、通信的波特率等参数，在终端不支持卡片当前使用的物理特性参数时，双方进行按照 ISO7816-5 标准进行 PPS 协商，使双方达成一致的物理特性接口参数，此后 COS 需要设置好所有的环境，等待应用的执行。

接收命令和发送响应数据：COS 从 I/O 输入缓冲区中一个字节一个字节地接收信息，通信管理模块需要对字节信息组装成完整的 APDU 格式，并采取奇偶校验、累加和及分组长度检验等到手段进行正确性判断，这判断只针对传输过程中的错误，不涉及信息具体内容的判断，判断的标准可参见 ISO 7816-4 APDU 指令结构；通信管理模块接收经过安全管理、命令处理、文件管理处理后的信息，并按照 APDU 指令结构要求打包成完整的数据帧，发送到 I/O 的输出缓冲区。

2. 安全管理模块

智能卡的安全体系是智能卡 COS 中一个极为重要的部分，它涉及到卡的鉴别与核实方式的选择，包括 COS 在对卡中文件进行访问时的权限控制机制，还关系到卡中信息的保密机制。鉴别(Authentication)指的是对智能卡(或者是读写设备)的合法性的验证，即是如何判定一张智能卡(或读写设备)不是伪造的卡(或读写设备)的问题；而核实(verify)是指对智能卡的持有者的合法性的验证，也就是如何判定一个持卡人是经过了合法的授权的问题。

安全体系包含了三方面内容：安全属性、安全机制、安全状态。

安全状态是指智能卡在当前所处的一种状态，这种状态是在智能卡进行完复位应答或者是在它处理完某命令之后得到的。事实上，我们完全可以认为智能卡在整个工作过程中始终都是处在这样或是那样的一种状态之中，安全状态通常可以利用智能卡在当前已经满足条件的集合来表示。

安全属性实际上是定义了执行某个命令所需要的条件，只有智能卡满足了这些条件，该命令才是可以执行的。因此，如果将智能卡当前所处的安全状态与某个操作的安全属性相比较，那么根据比较的结果就可以很容易地判断出一个命令在当前状态下是否是允许执行的，从而达到了安全控制的目的。

和安全状态与安全属性相联系的是安全机制，安全机制可以认为是安全状态实现转移所采用的转移方法和手段，通常包括：通行字鉴别，密码鉴别，数据鉴别及数据加密。一种安全状态经过上述的这些手段就可以转移到另一种状态，把这种状态与某个安全属性相比较，如果一致的话，就表明能够执行该属性对应的命令，这就是 COS 安全体系的基本工作原理。

3. 命令处理模块

命令解释模块：命令解析模块从通信管理模块中接收 APDU 命令，根据检查命令的各项参数是否正确，然后执行相应的操作，并做出应答；从而实现了外界对卡

内数据的操作。

智能卡的功能是通过解释执行所接收的外部指令，并回送相应的应答信息来体现的。根据 ISO/IEC7816-4(2005E)标准的一共规定了 39 条指令，不同的 COS 应用要求不同，可以对这些指令进行裁减。

4. 文件管理模块

文件操作是 COS 必须解决的三个基本问题之一，在设计过程中考虑的核心部分。COS 通过给每种应用建立一个对应文件的方法来实现它对各个应用的存储及管理。因此，COS 的应用文件中存储的都是与应用程序有关的各种数据或记录。此外，对某些智能卡的 COS，可能还包含有对应用文件进行控制的应用控制文件。

文件系统由主文件 MF(Master File)、专用文件 DF(Dedicated File)和基本文件 EF(Element File)组成。一般的文件系统是一个树结构，主文件 MF 是根节点，DF 相当于中间节点、EF 是叶节点，如图 3-2 所示。树结构的层次仅受存储空间大小的限制；每个文件在逻辑上和物理上都是连续的；文件的大小、读写权限等文件属性由用户在用 CREAT 指令创建文件时予以规定。每个 DF 在物理上和逻辑上都保持独立，都有自己的安全机制和应用数据，SELECT 命令用于通过应用选择实现对其逻辑结构的访问。基本文件 EF 用于存放用户数据或密钥，存放用户数据的文件为工作基本文件，在满足文件的读写权限时，用户可对文件进行读、写、修改等操作。存放密钥的文件存为内部基本文件，不可由外界读出，但当获得许可的权限时可在卡内进行相应的密码运算。

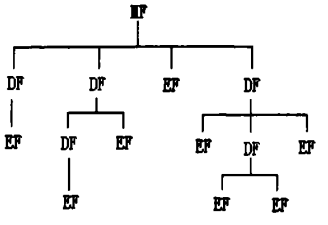


图 3-2 文件结构图

Figure 3-2 File Structure Frame

3.4 设计目标和原则

智能卡COS的研发是一件复杂而精确的事情，其复杂主要是与其相关的各种通信协议较为复杂，与外部的联系特征需要与协议一致，而实现其协议需要的硬件资源有限，要求较为精确。

本文开发的COS是在第二代移动通信系统UIM卡的基础上增加UTK应用。（简称

为UTKCOS）。UTK是“UIM card Tool Kit”的英文缩写，意思是“UIM卡开发工具包”。要使用手机附加增值业务，都要求将普通的手机UIM卡更换为UTK UIM卡。UTK可以理解为用于开发增值业务的小型编程语言，它允许基于智能卡的用户身份识别模块(UIM卡)运行自己的应用软件。

设计目标：一个良好的COS设计方案除需要满足用户的应用功能需求外，还应具有与外部终端的良好通信能力、对内部模块的高效协调管理和可扩展性、对不同硬件的较强适应性，因此COS设计时必须以实现这四方面为系统设计的总体目标。

设计原则：第一，COS严格遵照智能卡的国际规范IOS/IEC 7816-1-2-3-4及相关行业规范（如3GPP和ETSI中对UIM规定）。并且与符合行业规范标准的不同外部终端通信时保持通信一致性。第二，COS各个模块中的接口要标准规范化，有利于系统实现扩展。系统内部各模块之间调度良好，有利于实现系统的高效性，能及时的响应外部命令的输入。第三，在系统内添加一个功能模块或管理模式对系统的整体结构不能产生较大影响，系统仍然能保持系统的高效性和可靠性。第四，COS软件系统平台是建立在硬件智能卡基础上的，目前智能卡行业硬件处于高速发展状态，卡硬件的处理能力、存储容量都有了较大提高，COS设计时要求在应用于不同硬件平台时，对COS的移植代码修改最小，不影响COS的整体上层结构，仍然保持COS的高效可靠运行。

3.5 系统结构提出

根据以上说的设计目标和设计原则，参考传统操作系统的系统结构，提出新的COS系统结构。

传统操作系统系统结构有：1、单块式结构，如图3-3所示。2、层次结构，如图3-4所示。3、客户/服务器结构，如图3-5所示。4、REO模型，图3-6 REO模型结构。

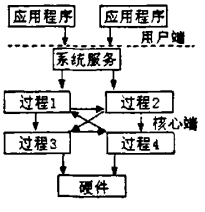


图3-3 单块式结构模型[2]

Figure3-3 Single Block Model

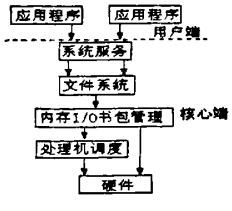


图3-4 分层系统结构[2]

Figure3-4 Hiberarchy Model

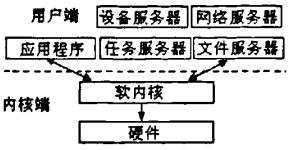


图3-5 客户/服务器系统结构 [2]

Figure3-5 C/S Model

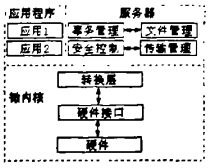


图3-6 REO模型结构[2]

Figure3-6 REO Model

四种结构模型各自有优缺点、使用场景不同，单块模型和层次模型效率较高，但不利于进行维护和扩展，REO模型和客户/服务模型模块封装特性较好，利于系统的扩展和维护，但对硬件的配置要求较高，任务调度复杂，需要耗费较多系统资源。

COS系统首先是一个高效准确的系统，其次要求具有良好的可维护性，和可扩展性。COS系统需要实现与外部终端的高效准确的通信，系统模块化设计使系统具有较好的可扩展性和可维护性，模块之间任务调度效率以实时为目标，系统需要在不同智能卡平台上进行扩展使用，对不同的智能卡硬件平台需要有较好的适应性。

基于本项目COS系统的目标和特点，系统设计方案提出一种适合于本项目的结构模型方案，方案结合了单块模型、层次模型、客户/服务模型、REO模型的优点，主要的思想是：为了系统的可扩展、可维护性、对不同硬件的适应性，系统采用REO模型和客户服务模型中的微内核思想进行模块化设计，为了使系统具有较高的运行效率，系统运行时对智能卡的硬件要求较低，节省部分硬件成本，系统模型简化REO模型的任务调度方式，将系统调度方式设计为单任务调度方式。系统模型结构如图3-7所示：

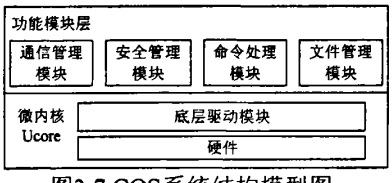


图3-7 COS系统结构模型图

Figure3-7 COS Frame Model

COS结构模型从整体上，分为两个大的层次功能层和微内核层。功能层主要实现COS的应用逻辑处理功能，并调用底层资源管理模块中的底层驱动程序实现对硬件的操作。主要包含通信管理模块、安全管理模块、命令处理模块、文件管理模块。

1．功能层模块层

通信管理模块：按照ISO7816-3，ISO7816-4标准协议规范方式，实现与外部的数据通信。通过奇偶校验、累加和及分组长度检验等手段判断接收数据的正确性，

但不进行信息内容的判断；还负责将安全管理、命令处理、文件管理模块产生的响应信息，经I/O的输出缓冲区发送到终端；

安全管理模块：实现网络与用户相互鉴权，文件的安全访问，机密信息的加密解密，安全报文的处理，相关安全算法的实现等。鉴权的方式是，终端向智能卡发送鉴权命令，智能卡向终端产生响应信息，依靠多次交互实现鉴权。安全模块为每个文件编写了安全访问规则，外部指令对文件的访问必须满足安全规则才能执行。

命令解释模块：对外部指令进行解析，分析指令类别CLA和命令类别INS，并具体执行相应命令的具体功能，执行后产生执行的结果信息。

文件管理模块：对标准中的文件系统的逻辑结构，进行物理实现，包括空间规划，目录文件处理、文件头、文件数据等处理。实现文件系统的初始化、检索、更新、创建、删除等操作。

2. 微内核层

针对该COS能快速移植到其他公司芯片中，把功能层和硬件驱动层分开设计，降低两个模块的耦合性。并且本文COS功能模块之间的调用关系采用与层次结构相似的调度方式， COS中模块的结构关系如图3-8所示。与传统调用方式相比，该系统结构具有更高的效率，主要体现在对安全管理模块的设置上，传统COS中所有从文件系统中进出终端的数据必需进行安全处理，对数据的加密或解密，判断访问文件权限是否符合当前的安全条件等，但在COS系统中并不是所有信息都需要进行安全处理，例如网络的鉴权；或者连续对同一文件的访问时，都不需要重复进行安全处理，而是根据命令类别的需要进行适当的安全处理，比如部分文件数据的加密等。

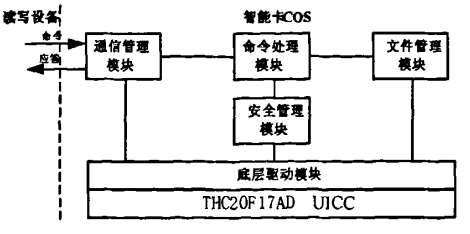


图3-8COS模块关系

Figure3-8 8COS Model Relation Model

功能层各模块之间的关系为程序调度请求和数据返回关系，功能模块调用关系如图 3-9 所示，图中的箭头表示了一种模块之间的一种外部调用关系，箭头表示了函数调用方向和层次，箭头端的一方表示逻辑的下层，另一方表示逻辑的上层。

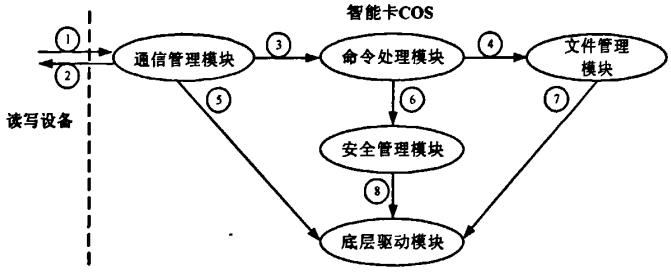


图 3-9 模块状态转换及调度关系图
Figure3-9 Module Relations

与状态图相对应的是如表3-1所示的状态表

表3-1 模块状态转换及调度关系表
Table3-1 Module Relations

状态	通信管理模块	安全管理模块	命令处理模块	文件管理模块	底层驱动模块
通信管理模块			③		⑤
安全管理模块					⑧
命令处理模块		⑥		④	
文件管理模块					⑦
底层驱动模块					

注意：图3-9中的①②表示卡与终端之间的一种通信关系。

按照上图中的模块调用关系描述，需要设计每个模块之间的接口及其数据结构。

1. 接口①

本处特别说明接口①不代表函数的调用关系，只表示终端向智能卡的数据输入。按ISO－7816的串行异步通信标准，终端向智能卡输入的数据有两种协议——字符传输和块传输。是以APDU结构进入智能卡，它包括由一个5Byte组成的命令头。每个命令头由5个连续字节CLA、INS、P1、P2和P3组成。

- CLA（Class Byte of the Command Message）表示命令类别。
- INS（Instruction Byte of the Command Message）表示指令代码。
- P1（Parameter 1）和P2（Parameter 2）表示命令的两个附加参数。
- 根据不同的INS，P3（Parameter 3）指明了发送给IC卡的命令的字节长度或者期待IC卡回送响应的最大数据长度。

在COS中以一个全局的256字节数组进行接收处理。数组的定义和表示如下：

```
#define CLA (gCommBuf[0])
#define INS (gCommBuf[1])
#define P1  (gCommBuf[2])
#define P2  (gCommBuf[3])
#define Len (gCommBuf[4])
```

```
#define Data    (gCommBuf + 5)
#define APDU_BUFFER_SIZE (5 + 256)
unsigned char xdata APDU[APDU_BUFFER_SIZE];
```

2. 接口②

接口②在本处有两种含义表示，一种是COS向终端发送APDU指令响应，另一种是在智能卡上电复位时，COS向终端发出的ATR响应。两种数据的定义如下。

APDU响应定义，为了节省内存资源，COS中定义响应数据与接口①中数据使用相同的内存资源，数据区至少占256字节。

```
#define SW1 (APDU [0])
#define SW2 (APDU [1])
#define Data (APDU + 5)
```

ATR的响应按照ETSI 102.221标准定义了两种结构如下：

```
Struct  ATR_T0{    //T=0 时的基本 ATR 结构
  Unsigned char m_TS;   //指明正向约定，值为 0x3B
  Unsigned char m_T0;   //TB1 和 TC1 存在，值为 0x97 历史字节
  Unsigned char m_TA1;  //FI=1(F=372),DI=1(D=1),值 0x11
  Unsigned char m_TD1;  //只有 TD2 存在，值为 0x80，支持 T=0 协议
  Unsigned char m_TD2;  //只有 TA3 存在，T=15，值为 0x1F
  Unsigned char m_TA3;  //3V 技术的 UTKUIM 卡，值为 C3
  Unsigned char m_T1;   //80
  Unsigned char m_T2;   //31，卡的数据服务
  Unsigned char m_T3;   //C0，EFDIR 的存在,支持 AID 的选择
  Unsigned char m_T4;   //73，卡的性能
  Unsigned char m_T5;   //BE，SFI 支持
  Unsigned char m_T6;   //21，数据译码字符
  Unsigned char m_T7;   //00，没有 Lc 和 Le,没有信道支持
  Unsigned char m_TCK;  //47，校验字符
   };
Struct  ATR_T1{    //T=1 的基本 ATR 结构
  Unsigned char m_TS;   //指明正向或反向约定，值为 0x3B 或 0x3F
  Unsigned char m_T0;   //TB1 和 TC1 存在，值为 0x6X，X 表示历史字节的存在个数
  Unsigned char m_TB1;  //不使用 VPP，
  Unsigned char m_TC1;  //指明所需额外保护时间的长度
  Unsigned char m_TD1;  //TA2 到 TC2 不存在，TD2 存在；使用 T=1 协议，值为 0x81
  Unsigned char m_TD2;   //TA3 到 TB3 存在，TC3 和 TD3 不存在；T=1 协议，值为 0x81
  Unsigned char m_TA3;  //表示 IC 卡的信息域大小的初始值且具有 16-254bye 的 IFSI
  Unsigned char m_TB3;  //BWI=0-4，CWI=0-5
  Unsigned char m_tck;  //较验字节
   };  //ATR_T1
```

3. 接口③

接口③为通信管理模块和命令处理模块之间的接口，此接口中命令处理模块提供如下函数供通信管理模块使用，此部分接口处理的为接口①中的全局 APDU 数组：

表 3-2 接口③函数表
Figure3-2 Interface3 Functions List

函数名称	输入	输出	功能
cmd_translate	iu8 *data, CAPDU *apdu	bool	解释 data，并把分析结果放进 apdu
cmd_setdata( )	Void	STATUS	命令数据解析

4. 接口④

接口④为命令处理模块和文件系统模块之间的接口，此接口中文件系统模块提供如下函数供命令处理模块使用。

表 3-3 接口④函数表
Figure3-3 Interface4 Functions List

函数名称	输入	输出	功
IniFileSystem	Void	Status	初
SelectFile	FileNode*CurrentDirectory,FileNode *CurrentFile,Iu16 fid	Status	查
UpdateFile	Iu8 mode, FileNode *CurrentFile, Iu8 *pdata, Iu8*pDest,	Status	更
DelFile	Iu mode, FileNode *Currentfile	Status	删
CreateFile	FileNode*CurrentDirectory, FileNode*CurrentFile, FileNode	Status	创

5. 接口⑤

接口⑤为通信管理模块和底层驱动模块之间的接口，此接口中底层驱动模块提供了如下函数供命令处理模块使用：

表 3-4 接口⑤函数表
Figure3-4 Interface5 Functions List

函数名称	输入	输出	功能
writeUART	unsigned char *sendBuf, unsigned short intLen	Void	写串口
readUART	unsigned char xdata * receiveBuf, unsigned char data expectLen	Void	读串口
readRNG	Void	Void	读随机数
runCRC	Void	STAT	循环冗余校验

6. 接口⑥

接口⑥为命令处理模块和安全管理模块之间的接口，此接口中安全管理模块提供了如下函数供命令处理模块使用：

表 3-5 接口⑥函数表
Figure3-5 Interface6 Functions List

函数名称	输入	输出	功能
safety_check_pinstate	Void	Void	Pin 状态检测
runDES	Void	void	运行 DES 算法
AES	明文数据或者密文数据，K	void	运行 DES 算法
runMAC	带有 MAC 的数据	void	运行 MAC 算法

7. 接口⑦

接口⑦为文件管理模块和底层驱动模块之间的接口，此接口中底层驱动模块提

供了如下函数供文件管理模块使用：

表 3-6 接口⑦函数表

Figure3-6 Interface7 Functions List

函数名称	输入	输出	功能
WriteFlas	Char mode, intLen, unsigned char *pSrc, unsigned char	Void	写 FLASH
readFlash	unsigned char intLen, unsigned char *pSrc, unsigned char	Void	读 FLASH

8. 接口⑧

接口⑧为安全管理模块和底层驱动模块之间的接口，此接口中底层驱动模块提供了如下函数供安全管理模块使用：

表 3-7 接口⑧函数表

Figure3-7 Interface8 Functions List

函数名称	输入	输出	功能
writeUART	unsigned char *sendBuf, unsigned short intLen	Void	写串口
writeSFR	Void	STATUS	写寄存器
readRNG	Void	STATUS	读随机数
readSFR	Void	STATUS	写寄存器
calDES	unsigned char mode, unsigned char * pDataOut,		调用 DES 加速
	unsigned char * pDataIn, unsigned char * pKey		

第四章 COS 硬件驱动设计

本文重点并不是讲述命令模块、通信模块、安全模块和文件模块的实现，而是讲述硬件驱动模块以及绑定式方式实现芯片移植。

硬件驱动层位于功能层和硬件之间，是沟通功能层和硬件的重要桥梁。硬件驱动层必须保证 COS 与外部终端良好通信能力；对内存读写正确、快速；以及对掉电情况采取相应手段，保证数据准确性和完整性。

4.1 主控函数设计

COS 与我们常说的操作系统具有一定的差别，因为智能卡 COS 并不具备计算机操作系统的并发性、多线程性，文件系统有相对简单。从某种意义上说，COS 只不过是一个在智能卡中运行的监控程序。但在 COS 设计时，本文还是参照了操作系统的部分理论基础（如文件系统、内存管理）以及软件工程的设计方法进了系统整体模块划分。项目设计过程中与智能卡国际规范及 3G 行业规范紧密结合，COS 在外部特性上满足了相关的规范标准要求，在内部实现上，不同的 COS 具有一定的差别。

COS 最重要的功能是命令处理，每次终端发送一条命令到命令结束的过程可以视为一个功能子流程。其中每个功能子流程包括：接收终端发送的命令，启动命令处理流程，返回命令响应数据，等待下一条命令输入。智能卡 COS 在上电复位后首先执行 COS 主守护流程，进行 ATR 应答，运行环境初始化，然后就进入功能子流程。功能子流程的实现通常有两不同的方式：

第一种是循环等待方式。主守护流程进入功能子流程以后，不断地检测是否有新的命令输入，如果接收到新命令就启动命令处理流程，最后返回命令的响应数据，然后继续检测是否有下一条命令输入。

第二种是中断方式，主守护流程进入功能子流程以后，开始进入中断等待状态，如果有外部命令的输入，将触发一个卡片中断，主守护流程被唤醒，开始进行命令接收，然后启动命令处理流程，最后返回命令的响应数据，然后主流程再次进入中断等待状态。

循环等待方式智能卡需要不断地查询是否有新命令输入，CPU 一直处于工作状态；而采用中断方式，功能子流程看作一个中断，当接收到新命令时才进入执行状

态。对于用电池供电的读卡终端（如手机）具有节能用。

针对 UTKCOS 项目，目前基本上所有的芯片都支持中断，从设计开发的角度看使用中断方式的 COS 要对不同的芯片具有较好的适应性，系统复杂度较高；从用户的角度看，其具有节能的功能，从产品服务用户的角度，提高产品在同行业中的竞争力，本项目拟采用中断调度技术。但是采用中断模式与平台的相关性较强，不同芯片的中断模式有一定区别，开发难度较大。项目设计过程中的难点在于各种芯片的中断模式有一定差别，需要从软件上进行解决。

1．中断处理的一般过程

广义上中断处理是由计算机的硬件和软件配合起来完成的，其中硬件部分完成的过程称为中断响应，软件完成的过程则是执行中断处理程序的过程，不同的硬件对中断实现的方式有一定差别，一个典型的处理过程如图 4-1 所示：

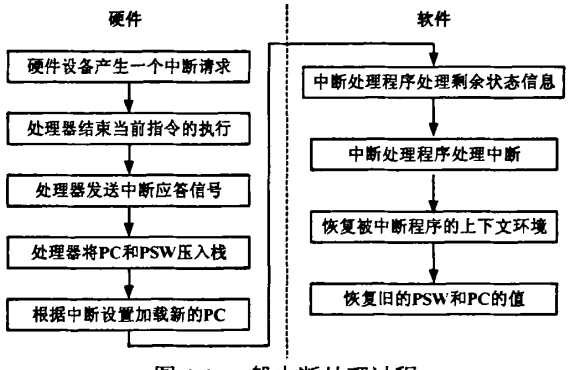


图 4-1 一般中断处理过程

Figure4-1 Common Interrupt Disposal Process

2．智能卡支持的系统中断源

在 A 公司 A112F 芯片中，硬件定义了 5 个中断，分别是 DES、Timer0、UART、Timer1、FLASH。中断编号以及入口地址如下表 4-1 所示：

表 4-1 中断向量表

Table 4-1 Interrupt Vector Table

中断信号	中断源	中断编号	中断向量地址
DES	DES 运算结束	0	0003H
T0	TIMER 溢出	1	000BH
UART	UART 字符接收/发送就绪	2	0013H
T1	TIMER1 溢出	3	001BH
FLASH	FLASH 写操作结束	4	0023H

3. 主控程序设计

COS 系统主控程序设计时主要利用智能卡的 UART 中断和系统休眠模式。当终端发送命令到芯片中时，产生 UART 中断信号。系统检测到该信号时，将关闭其他低级中断源，并对该命令进行处理和响应。命令结束后，系统进入休眠模式并等待下一命令输入。主控程序算法描述如图 4-2：

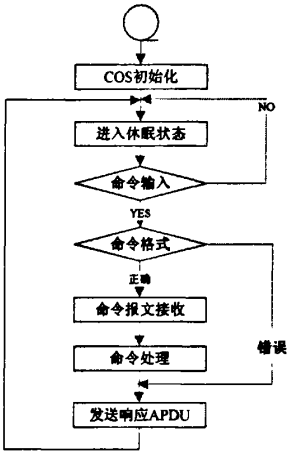


图 4-2 主程序流程图
Figure 4-2 Flow Chart of Main Program

4.2 ATR 响应

ATR 是智能卡上电复位后，由智能卡发送到终端的一串数据，也是智能卡向终端发送的第一串数据。在 ATR 响应前，终端与智能卡不能进行任何 APDU 命令与响应。发送 ATR 的目的有两个，一是告诉终端智能卡存在；二是告诉终端，智能卡当前遵循的通信物理参数、电气特性。

1. ATR 格式

ATR 结构如图 4-3 所示：

TS	T0	TA1	TB1	TC1	TD1	TA2	TB2	TC2	TD2	TA3	……	T1	……	TK	TCK

图 4-3 ATR 结构
Figure4-3 ATR Structure

TS 表示字符串方向，必备的；
T0 格式字符，必备的；
TA(i) TB(i) TC(i) TD(i) 接口字符，任选的，i>2；
T1 T2 TK.....历史字符，任选的；
TCK .. ..校验字符，有条件的；

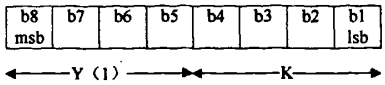
1）初始字符和编码约定 TS

TS 字节用于表示 ATR 字符串的方向。“3B”表示正向，“3F”表示反向。

2）格式字节 T0

如图 4-4，T0 字节由两部分组成：

— 位 b8 到 b5 构成 Y(1)；每个等于 1 的位指示了后续接口字节的存在；
— 位 b4 到 b1 构成 K，K 编码了历史字节的个数，从 0 到 15。



Y（1）……接口字节存在的指示符；
若 b5=1，TA（1）存在；
若 b6=1，TB（1）存在；
若 b7=1，TC（1）存在；
若 b8=1，TD（1）存在；
K……历史字节的个数，从 0 到 15；

图 4-4 T0 编码[4]
Figure4-4 T0 Coding

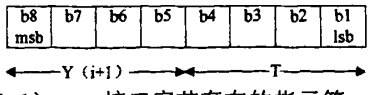
当 T0 为“0X”的时候，说明 TA（i）、TB（i）TC（i）TD（i）均不存在。终端认为该芯片使用默认频率和电压。现在很多芯片都把 T0 设置为“0X”，所有参数均为默认值。

2. 接口字节 TA(i) TB(i) TC(i) TD(i)

1) TD(i)

按照图 4-5，字节 TD(i)含有两部分：

—位 b8 到 b5 构成 Y(i 十 1)；每个等于 1 的位指示后续接口字节的存在；
—位 b4 到 b1 构成参数 T 的值；
参数 T 值如下：
—T= 0 指异步半双工字符传输。
—T= 1 指在异步半双工块传输。
—T= 2 和 T=3 保留用于将来的全双工操作。
—T= 4 保留用于增强的异步半双工字符传输。
—T= 5 到 T=13 保留待将来使用。
—T=14 指未由 ISO/IECJ TCIS C17 进行标准化的传输协议。
—T=15 不指某一种传输协议，仅限定了全局接口字节。



Y（i+1）……接口字节存在的指示符；
若 b5=1，TA（1）存在；
若 b6=1，TB（1）存在；
若 b7=1，TC（1）存在；
若 b8=1，TD（1）存在；
K……协议参考和/或接口字节限定符；

图 4-5 TD（i）的编码[4]
Figure4-5 TD(i) Coding

因此，TO 运送 Y(1)而 TD(i)运送 Y(i+1)。在运送 Y(i)的字节中，位 b8 到 65 表示与 b5 对应的 TA(i)、与 b6 对应的 TB(i)、与 b7 对应的 TC(i)、与 b8 对应的 TD(i)按照这个次序运送 Y(i)的字节之后是否存在(取决于相关的位是否等于 1)，

如果 TD(i)不存在，则接口字节 TA(i 十 1),TB(i+1),TC(i+1)和 TD(i+1)也不存在。

如果两个或更多参数 T 的值存在于 TD(1)TD(2)……中，它们应当按照数字升序存在。如果存在，T=0 应是第一个，T=15 应是最后一个。TD(1)中禁止值 T=15。

“第一次提供的协议”定义如下:

— 如 果 TD(1)存在，则第一次提供的协议是 T,

— 如 果 TD(1)不存在，则提供的唯一的协议是 T=0。

2) TA(i) TB(i) TC(i)

接口字节 TA(i),TB(i)和 TC(i)(i=1,2,3,…)是全局的或特定的。

— 有关卡上集成电路参数的全局接口字节，可参见 ISO7816-3 中关于全局接口参数的定义。

— 有关卡提供的传输协议的参数的特定接口字节。

接口字节 TA(1)T B(1)TC(1)TA(2)TB(2)是全局的。接口字节 TC(2)是特定的;它是为 T=0 定义的，i>2 时对接口字节 TA(i)TB(i)TC(i)的解释依赖于 TD(i-1)中参数 T 的值。

— 如果 T=15，则接口字节是协议 T 特定的。

— 如果 T=15，则接口字节是全局的。

如果为参数 T 的同一个值定义了超过三个的接口字节 TA(i)TB(i)TC(i)，并且这些接口字节在复位应答中出现，则它们应相继出现于指明相同 T 值的 TD(i-1)TD(i)... (i>2)之后；因此，当这些接口字节在 TD(i-1)(i>2 时)中的 T 第一次、第二次或第 n 次出现之后出现，就被无二义性地识别出来。

3）历史字节 T1T2……TK

历史字节指明一般信息，例如卡制造商代号、插人卡内的芯片、芯片的掩膜 ROM、卡的寿命状态。如果 K 不为 0，则复位应答在 K 个历史字节 T1T2.....TK 上继续。

表 4-2 历史字节编码表
Table 4-2 History Byte Encoding Table

名 称		长度（字节）	含 义	备 注
芯片信息	芯片提供商	1	提供芯片的厂家代码	
	芯片型号	3	芯片的具体型号代码	
	芯片注册号	2	在国家机构申请的编号	如果没有则用 0000 代替
COS 信息	COS 功能	1	STK、UTK、PIM 等功能	按电信规范
	COS 版本号	2	COS 版本占 1．5 个字节，调试版本占 0．5 个字节。	COS 版本从 001 递增。表示 V0.01 版 COS，调试字节从 1-F，如果测试通过则变为 0。整个 COS 版本就是 V0．010
	应用空间	2	指留给文件系统和应用的空间总和	举例： 20K：0020，16K：0016，54K：0054，64K：0064，128K：0128
卡商信息	卡商代码	2	卡商在国家机构申请的编号	如果没有则用 0000 代替
卡片状态	卡片状态	1	卡片所处的阶段	00：表示已下载 COS 01：表示已个人化卡片 其他保留

4)校验字节 TCK

TCK 字节是校验位，它与 T0 至 TCK 的所有字节(包括 T0 和 TCK)的异或运算结果应该为 0。如果只有 T=0 存在(可能通过默认来指明)，则字节 TCK 应不存在。如果 T=0 和 T=15 都存在，或者在其他所有情况下时，字节 TCK 应存在。

4.3 PPS 协商

终端接收到智能卡 ATR 响应字符串之后。检测智能卡的电气参数和协议参数是否有终端所要求的一致，如果终端希望智能卡用更高的波特率或者用更高的电压。这时候需要进行 PPS 协商。

1.传输因子 F 和 D

在 I/O 接口上使用的位持续时间被定义为一个基本时间单元(Elementary Time Unit，简称 etu)

初始 etu=372/f，式中 f 是时钟频率，单位是赫兹（Hz）。

复位应答之后，智能卡和终端之间通过协商新的参数 F（时钟速率转换因子）和 D（波特率调整因子），以统一传输的波特率。

letu=F/D×（1/f）

Fi 和 Di 按照表 4-3 和表 4-4 给出：

表 4-3 时钟速率转换因子表

Table4-2 Clock-speed Switch Factor Table

FI	0000	0001	0010	0011	0100	0101	0110	0111
Fi	372	372	558	744	1116	1488	1860	RFU
FI	1000	1001	1010	1011	1100	1101	1110	1111
Fi	RFU	512	768	1024	1536	2048	RFU	RFU

表 4-4 波特率校正参数值

Table4-3 Baud-rate Revising Parameter Table

DI	0000	0001	0010	0011	0100	0101	0110	0111
Di	RFU	1	2	4	8	16	32	RFU
DI	1000	1001	1010	1011	1100	1101	1110	1111
Di	12	20	RFU	RFU	RFU	RFU	RFU	RFU

2. PPS 请求和响应的结构协议

PPS 请求和响应分别包括一个初始字节 PPSS，后面是格式字节 PPS0，三个可选参数字节 PPS1, PPS2, PPS3 和一个检验字节 PCK 作为最后一个字节，如图 4-6 所示。

PPSS	PPS0	PPS1	PPS2	PPS3	PCK

图 4-6 PPS 帧结构

Figure4-6 PPS Frame Structure

PPSS 标识了 PPS 请求或响应，PPSS 的值等于“FF”。

PPS0 根据位 b5, b6, b7 是否等于 1 来指明可选字节 PPS1, PPS2, PPS3 是否存在。位 b4 到 b1 运送参数 T 的值以建议一种协议。位 b8 保留作将来使用并应置为 0。

PPS1 允许终端向卡建议 F 和 D 的值。

按与 TAM 中相同方法进行编码，这些值应分别位于 Fd-Fi 和 Dd- Di 的范围内。如果 IFD 不发送 PPS1，则它建议继续使用 Fd 和 Dd，智能卡或者通过回送 PPS1 来确认两个值(然后这两个值就变为 Fn 和 Dn，或者不发送 PPS1，继续使用 Fd 和 Dd(然后，Fn 置为 372, Dn 置为 1), PPS2 和 PPS3 保留作将来使用。

PCK 的值是校验字节，使涉及所有字节 PPSS 至 PCK(包括 PPSS 和 PCK)的异或运算结果为 0。

3. PPS 协议

PPS 请求和响应应该以与复位应答相同的方式来发送，即以相同的波特率,符合 TS 建立的约定，并且在连续两个字符的起始沿具有最小延迟 12etu。然而如果接口字节 TC(1)出现在复位应答中，且不等于“FF”，则应保证有附加的保护时间。PPS 响应的两个连续字符的起始沿之间的延迟不应超过“初始等待时间”。

终端应发送一个 PPS 请求给智能卡；
智能卡操作：
if（PPS 错误）
　　智能卡不发送任何相应；
else
　　智能卡发送一个 PPS 响应；

终端操作：
if（12etu 时间接收到智能卡响应）
　　if（响应正确）
　　　　按协商值进行通信；
　　else
　　　　复位，或者拒绝智能卡；
else
　　复位，或者拒绝智能卡；

现在的手机都是进行两次 PPS 协商，如果两次协商都失败，手机就按默认值进行传输，不再进行 PPS 协商。

下面是智能卡与终端开机时候交互的数据：

Reset
3B 1C 95 00 00 10 02 03 00 00 10 00 64 00 00 01

//PPS　第一次协商PPS
FF 10 12 FD
FF 9F　//协商失败
//PPS　第二次协商
FF 10 12 FD
FF 9F　//两次协商失败，不再进行协商，运用默认参数通信。

//GET RESPONSE　//读取文件
A0 C0 00 00 17
00 00 00 00 7F 25 02 00 00 00 00 00 0A 01 00 4C 06 00 83 8A 83 8A 00
90 00　//返回成功

4.4 驱动函数设计

1. 串口驱动设计

按A公司芯片，串口传输将涉及到UBUF、USR、UCR寄存器；当UCR.MOD＝‘0’时，表示传输使用UART模式。 此时由芯片UART模块按ISO7816-3协议自动处理，完成字符帧传输（包括起始位、数据位和奇偶校验位、错误检测及处理）。“接收寄存器满”和“发送寄存器空”等事件能触发CPU的UART中断。

（1）发送

复位后UART处于接收状态，在发送任意字符前， COS必须将UART转换到发送模式。COS不必关心错误检测及处理，这步由智能卡硬件完成，等待硬件将USR.TBE置1后，将该字节打入UBUF，再将USR.TBE清零（使能发送）。如果来自读卡机的NAK信号出现在起始位下降沿之后的11 etu处，则硬件认为该字符传输失败。于是UART自动重发该字节，直到没有错误信号出现。

卡片通常被要求实现重发次数可控，COS的操作流程是：

step1.清UCR.PH为0，关闭硬件自动处理偶校验错误；

step2.转换UART到发送模式，将该字节打入UBUF，将USR.TBE清零（使能发送）；

step3.等待硬件将USR.TBE置1后，转换UART到接收准备状态。转换的目的是，令硬件及早进入接收状态，及时处理可能出现的待接收字符；

step4.轮询UCR.TACT和USR.PE，如果USR.PE置位，则重复step2到step4若干次。

函数流程图如图4-7所示：

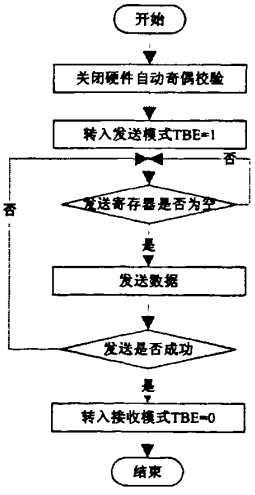


图 4-7 发送函数流程图

Figure4-7 Send function Flow Chart

（2）接收

在接收任何字符前，COS均应将UART转换到接收准备状态。在接收模式下，要接收一个字节，COS只需等待硬件将USR.RBF置1后，读取UBUF，COS不必将RBF清0，因为在COS读取UBUF时，硬件会自动将RBF清零。当UCR.PH=1时，COS不必关心错误信号产生及处理。 当偶校验错误时，UART将发送1etu的错误信号，并期待发送方重发该字节。只有校验位正确的字符才会被硬件存入UBUF。 当字符送入UBUF后，硬件将RBF置1，并发出中断请求。接收函数伪代码如下：

```
开始
    UCR.PH寄存器置1，硬件自动检测奇偶错误；
    转入接收状态TBE=0；
    do{
        等待USR.RBF硬件置1；
        读取UBUF寄存器；
        读取长度减一；
    }while（长度<=0）；
结束
```

2. DES加密解密协处理器

在数据加密和解密运算时，需要大量的运算操作，为了减轻CPU负担，使用DES协处理器很有必要。但是DES加密解密协处理器不同公司的芯片有不同的控制方法。需要参考该公司的芯片资料手册进行编写。一般的操作方法是：

1）把需要加密或解密的数据存到相应的寄存器中。

2）触发控制寄存器相应的控制位。

3）读取结果数据。

3. FLASH读写设计

FLASH技术发展越来越成熟，成本也越来越低。现在电信行业智能卡都使用FLASH做存储介质。但是FLASH有个特点，就是擦除要整块擦除（512个字节）。所以在改写之前，必须把原来的数据保存到另外一个地方。然后再擦写FLASH，最后把以前的数据回写到原来的地方。就是说，要是只改写一个字节，必须把511个字节移到别处，改写好之后，再把这511个字节移动回原来地方。

FLASH读写使用到的寄存器主要有FL_CTL（FLASH 擦写状态）、FL_SDP1，

FL_SDP2（擦写序列保护寄存器）、FL_RETRY（FLASH擦除重试控制）、FL_PATCH（FLASH擦写中断补丁）。

为了保护Flash中数据的安全，对Flash要进行擦除和写操作之前，都要先向Flash中写入一串特定的命令字，正确方可擦写。这项技术叫做SDP（Software Data Protection），可以防止误擦写。就是说对FLASH进行擦除和写操作之前，对FL_SDP1，FL_SDP2（擦写序列保护寄存器）写入特定字节。

(1)改写FLASH时，COS通常应该先擦后写，擦写FLASH的一般流程：

1)先将目标区域切换到XDATA。

2)缓存目标sector中的无关数据。

3)通过SFR控制和写XDATA动作，完成擦操作(erase)：将整个sector中的每个字节擦为FFh，硬件自动擦校验和并在必要时擦除重试若干次。

4)通过SFR控制和写XDATA动作，逐字节完成写操作(program)：改写目标字节，并回写先前缓存的无关数据。

5)COS校验擦写结果。

(2) Sector擦操作步骤：

1)先向SFR SDP1写入55h，再向SFR SDP2写入AAh；

2)将目标bank切换到XDATA（这一步也可以位于第一步之前；如果目标bank已经位于XDATA，则可以省略这一步）；

3)向目标sector中的任意地址写入数据FFh；

4)置FL_PATCH为78h，置IE为仅使能FLASH中断，使CPU进入IDLE；

5)在CPU IDLE状态下，硬件完成擦除目标sector，并根据SFR FL_RETRY的设置自动进行回读校验。如果必要，硬件还会自动做擦除重试。完成后，硬件给出中断申请唤醒CPU。

6)CPU退出IDLE后，应立即FL_CTL.OVER标志，再清除FL_PATCH，并恢复IE为原值。最后，根据需要处理FL_CTL给出的2个错误标志。

(3)单字节写操作步骤：

1)先向SFR SDP1写入**h，再向SFR SDP2写入**h；

2)将目标bank切换到XDATA（这一步也可以位于第一步之前；如果目标bank已经位于XDATA，则可以省略这一步）；

3)向目标字节地址写入数据；

4)置FL_PATCH为78h，置IE为仅使能FLASH中断，使CPU进入IDLE；

5)在CPU IDLE状态下，硬件完成写入目标字节，随后给出中断申请唤醒CPU；

6)CPU退出IDLE后，应立即FL_CTL.OVER标志，再清除FL_PATCH，并恢复IE为原值。最后，根据需要处理FL_CTL给出的2个错误标志。

4.5 掉电保护设计

由于 FLASH 特性，要改写 FLASH 需要擦除一整块的 512 个字节 FLASH 空间,所以在改写过程中需要备份原来不改写的数据，再擦除 FLASH,之后再改写。上面改写 FLASH 程序有个缺点，就是把原来不改写的数据用缓冲区记录下来，如果在擦写完 FLASH，而缓冲区数据还回写时发生掉电，缓冲区的数据就会丢失。造成不可预见的结果。为了防止数据在改写时掉电，在改写时进行处理，即使发生掉电也不至于影响整个系统正常运行。

实现方式：实现掉电保护要把原来不改写的数据写到另一块 FLASH 空间，即使发生掉电，原来改写的数据能保存下来，所以需要在整个 FLASH 空间中开辟两块 512 字节大小的 FLASH 块作为备份区。

改写 FLASH 分两种情况：

1）改写的数据在同一块 FLASH 空间里面。如图 4-8 所示：

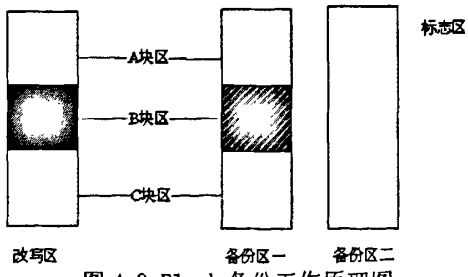


图 4-8 Flash 备份工作原理图

Figure 4-8 Flash Backup Schematic

(1)把备份区数据擦除；

(2)将改写区的 A 块区和 C 块区保存到备份区一上；

(3)标记为旧数据备份完成，保存改写区地址和 B 块区的数据大小；

(4)把新数据写到画斜线的 B 块区；

(5)在标志区，标记为改写备份完成；

(6)然后擦除改写区，把备份区 1 写到改写区；

(7)在标志区标记为改写完成；

(8)擦除备份区一和备份区二，以备下一次改写；

改写过程掉电，重新开机，读取标志位，若为旧数据备份完成，则根据第(3)保存的地址擦除改写区，把原来的数据恢复，标记改写完成，擦除备份区 1 和备份区 2，以备下一次改写。若为改写备份完成，则根据第(3)保存的地址擦除改写区，将整个备份区一写入改写区，擦除备份区 1 和备份区 2，以备下一次改写。

2) 当改写区为两 FLASH 块时，操作比较复杂如图 4-9 所示,要改写的为改写区一和改写区二的阴影部分：

(1)擦除两个备份区；

(2)将改写区一的 A 块区和 C 块区保存到备份区一上；

(3)标记为改写区一旧数据备份完成，保存改写区地址和 B 块区的数据大小；

(4)把新数据写到画斜线的 B 块区；

(5)然后擦除改写区，把备份区 1 写到改写区；

(6)在标志区标记为改写区一改写完成；

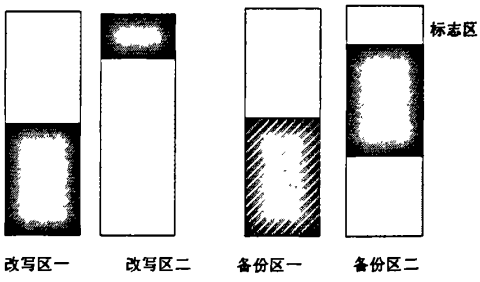


图 4-9 跨块改时 FLASH 备份工作原理图

Figure 4-9 Flash Backup Schematic When Span Area

(7)然后擦除备份区一，再把改写区二不改写的数据备份到备份区一；

(8)把新数据写到备份区一对应画斜线的位置；

(9)在标志区，标记为改写区二新数据备份完成；

(10)擦除改写区二，将备份区一数据写到改写区二；

(11)在标志区标记为改写区二改写完成；

(12)擦除备份区一和备份区二，以备下一次改写；

数据恢复：

开机读标志位，若为改写区一旧数据备份完成或改写区一改写完成，则把改写区原来的数据恢复到改写区一，若为改写区二新数据备份完成，则把备份区一的数据写到改写区二。

标志位设计如图 4-10：

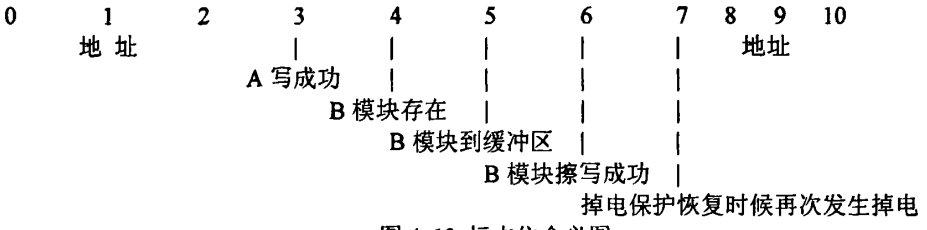


图 4-10 标志位含义图

Figure 4-10 Symbol Meaning Figure

智能卡复位以后，经过 ATR 响应和 PPS 协商，之后就检测该标志位，如果检测到出现掉电情况，应该对 FLASH 进行恢复。表 4-5 说明了不同标志位对应相应的操作。例如：当标志位是 FF00 FFFF FF 表示出现跨块写 Flash，在擦写 A 块后出现掉电，COS 需要把在备份区中 A 的数据回写到 A 区域，并把 A 原来的内容写回到 A 区相应的位置。

表 4-5 掉电保护恢复表

Table 4-5 Power-down Protection To Restore Table

标志位					意思	操作
3	4	5	6	7		
00	FF	FF	FF	FF	擦写成功	
FF	FF	FF	FF	FF	没有跨块时发生掉电	把缓冲区恢复目标块
FF	00	FF	FF	FF	跨块，擦写 A 时掉电	回写 A＋A 原来的内容
00	00	FF	FF	FF	A 成功，擦缓掉电	擦缓,A＋内容到,写标记,缓到 A
00	00	FF	FF	00	回滚 A 时掉电	擦 A，缓到 A
00	00	00	FF	FF	擦 B 时掉电	缓到 B
00	00	00	00	00	恢复成功	

第五章 绑定式设计

硬件驱动函数编写完成，COS 已经能实现一定功能，比如在指定地址空间上改写字符或者读取某个地址上的字节。功能层也是通过硬件层驱动函数完成与外界通信、内存读写和硬件使用等。硬件驱动函数与文件、安全、命令模块再加上菜单应用构成整个 COS 系统。把上述函数通过 keil 编译器编译生产 Hex 文件，再把 Hex 文件内容转换成 APDU 指令，用读卡器把这些 APDU 灌入新卡，最后完成个人化（例如写入电话号码，KI 密钥等）。此时的智能卡真正具有智能了，能实现打电话，发短信功能。

但是，该系统只能运用到 A 公司 A112F 芯片中，无法运用到其他公司的芯片中。若想把 COS 从一个芯片移植到另一个芯片并且作最少修改，首先必须知道：那些部分需要修改，那些函数需要保留。把每个 COS 共同拥有的部分保留，对不同的部分做出相应的改变。

芯片移植引起代码修改的原因有两个：第一，芯片改变。不同公司设计的芯片各有不同，实现控制的方法方式也是各不一样。第二，应用改变。例如原本运用到 CDMA 网络的 COS，改成运用在 CDMA2000 网络，其鉴权算法必须随之改变。本文对功能层的差异实现暂不详述。下面讨论是由于芯片改变引起的移植问题。

举个例子。下面是 A 公司和 B 公司对比：

表 5-1 A 公司与 B 公司物理特性比较表

Table 5-1 A Company and B, Physical Properties Comparison Table

公司		A公司	B公司
型号		A112F	B128F
CPU		8位 8051内核	8位 8051内核
ROM			
RAM	内	256	256
	外	2K	2.5K
EEPROM			
Flash		112K	128K
随机数发生器		8位	
定时器		2×16	
RSA			
CRC		有	

DES	有	有
I/O	T=0(字符传输)	T=0(字符传输) 附带ETU计数器
工作频率	1－5MHz	1－5MHz
工作电压	3V/5V	3V/5V
节电模式	Standy stopclock	Standy stopclock

从上面对比可以看出，两个公司的产品有共同之处：采用 8 位 8051 内核 CPU、存储设备采用 Flash、有 DES 协处理器、I/O 采用相同的通信标准、相同工作频率和工作电压等。但是也有很多不同的地方，比如 Flash 大小不一样、RAM 大小不同、A112F 芯片有他自己的定时器、CRC 循环冗余校验电路、和随机数发生器，而 B128F 就没有这些设备。

而对芯片硬件进行操作和控制，都是通过相应的寄存器进行的。下面是上述两个芯片的特殊寄存器地址对比图。

	00	01	02	03	04	05	06	07
F8H	DESD0	DESD1	DESD2	DESD3	DESD4	DESD5	DESD6	DESD7
F0H	B							
E8H	FL_CTL	FL_SDP1	FL_SDP2	FL_PETRY			BANK_N	BANK_SEL
E0H	ACC	RNGCTL	RNGDAT					FL_PATCH
D8H	USR	UBUF	UBRC	CHIPSC				UCR
D0H	PSW	MP						
C8H								
C0H	DESCTL		CRC					
B8H	IP							
B0H	P3							
A8H	IE							
A0H	P2							
98H	SCON	SBUF						
90H	P1		PMAGE	BSR				IDLE_EN
88H	TCON	TMOD	TL0	TL1	TH0	TH1		CLKSEL
80H	P0	SP	DPL	DPH				PCON

A芯片和B芯片寄存器一样

A芯片有而B芯片没有的寄存器

红字：B芯片有而A芯片没有的寄存器

A芯片和B芯片都有，但是控制方法不一样的寄存器

图 5-1 寄存器地址对比图

Figure 5-1 Comparison Chart Register Address Figure

在控制方面，两个芯片有其相同的部分和相异的地方。也就是说，若想把系统移植到新的芯片上去，必须改变其硬件驱动函数。

5.1 统一接口层的设计

当上层逻辑应用到不同的智能卡时，如果底层接口设计不好将会对上层代码的修改产生较大工作量，因此在进行 COS 设计时必须考虑底层接口对上层应用逻辑的通用性，尽量使用不同智能卡均支持的函数接口。

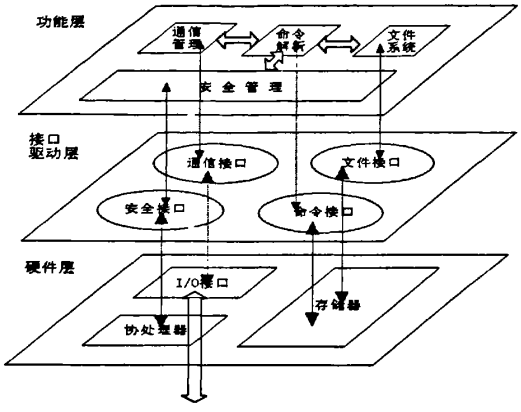


图 5-2 COS 立体结构图

Figure 5-2 COS Solid Structure Figure

在本层的设计上主要是对当前比较主流的智能卡具有的芯片特性进行分类，比如一个智能卡具有 FLASH，DES，RNG 等硬件电路模块，不同的智能卡对 FLASH 的擦除模式、DES 的运算模式具有通用性，因此在设计时，在软件上可以采用比较通用的函数接口，如图 5-2 所示。例如在 FLASH 的通用读写函数接口设计如表 5-2 所示：

表 5-2 FLASH 的通用读写函数接口

Table5-2 Common FLASH Interface Functions

函数名称	输入	输出	功能
writeFlash	char mode, intLen, unsigned char *pSrc, unsigned char *pDest	STATUS	写 FLASH
readFlash	unsigned char intLen, unsigned char *pSrc, unsigned char *pDest	STATUS	读 FLASH

WriteFlash 的输入参数：mode 表示一个 flash 的擦除模式；intLen 写入数据的长度；pSr 表示写入数据在内存中的源地址；pDest 数据写到 FLASH 中目标地址。输出 STATUS 表示写成功与否的状态。

ReadFlash 的输入参数：intLen 读出数据的长度； pSr 表示读出数据在内存中的存储地址；pDest 所读数据在 FLASH 中目标地址。输出 STATUS 表示读成功与否的状态。

上述接口对 flash 来说是一个比较通用的接口，在软件接口上可适用于不同的 flash 硬件，但对于不同的硬件来说，内部的实现过程不一样。其余智能卡电路模

块类似。

5.2 B128F 芯片硬件驱动函数设计

1.串口驱动设计

B128F 芯片设计的串口传输使用到的寄存器有串口数据缓冲寄存器 SBUF、串口控制寄存器 SCON、分频计数器 SCNT。

（1）发送

串口发送过程：首先，串口电路自动产生奇偶校验位，并把 SCON.TB 为置 1（表示串口忙），并在起始位触发后 11etu 处检测 I/O 信号，自动检测数据发送情况，如果发送错误，把 SCON.TX 置 1；否则把 SCON.TX 清零。如果 SCON.TX 为“0”，串口电路在起始位 12etu 处将 SCON.TB 清零；如果 SCON.TX 为“1”，则串口电路在起始位 13etu 处，将 SCON.TB 清零。

串口发送数据流程如下：

a. 等待串口空闲；

b. 把需要发送的字节放进发送寄存器；

c. 检测发送是否成功；

d. 若是不成功，记录失败次数，重复 a、b、c 步流程。

e. 若是发送不成功的次数超过 6 次。产生报错信息，退出发送函数；

f. 若是发送成功，按上述 5 个步骤发送下一个字节，直到所有字节发送完毕。

（2）接收

串口电路在不发送数据时，应该处于接收状态，在 SCON.SE=1 的条件下，串口电路进入接收状态。在接收过程中，串口电路自动完成校验奇偶校验位的正确性。如果正确，同时 SCON.RF=0（串口接收完成），则将数据存入 RSBUF 中，并将 RF 置位。如果奇偶校验错误或 SCON.RF=1（串口接收没完成）。

伪代码描述如下：

```
开始
        串口电路进入接收状态 SCON.SE 置 1；
        for（i=0;i<接收长度;i++）
        {
                等待串口空闲；
```

启动串口，STANDBY 位置 1；

等待接受完成，直到 SCON.RF 为 0；

接收数据；

}

结束

2.DES加速器驱动

DES 运算相关寄存器有两个：DES 数据或密钥缓冲寄存器 DBUF 和 DES 运算控制寄存器 DCON。操作方法：

1）选择操作模式，解密还是加密；

2）把数据放进 DBUF；

3）触发 DES 运算器；

4）判断 DES 运算是否完成；

5）读取结果数据。

3.FLASH读写设计

该款芯片的程序存储器采用128K的FLASH，由于标准的8051对程序存储器最大只支持64K，所以该芯片也是采用Bank的编址方式。程序存储器分成4个32K的区域，分别是Common区、Bank1、Bank2、Bank3.Common区对应的逻辑地址小于32K；当逻辑地址大于32K时，将根据BSR寄存器的值选择相应的Bank。

FLASH读写使用到的寄存器主要有BSR（Bank Switch寄存器）、PMWE（程序存储器编程使能寄存器）。

这芯片Flash擦写也采取了SDP，不同的是这芯片并不是向特殊寄存器写字节。而且向某两个特地地址写入字节。

(1)改写FLASH时，COS通常应该先擦后写,擦写FLASH的一般流程：

1）触发flash编程使能；

2）选择需要操作的Bank；

3）把不需要改写数据放进缓冲区；

4）向flash发送擦除SDP指令串；然后CPU时钟被停，直到擦除完成；

5）向flash发送改写SDP指令串并向指定地址写字；节然后CPU时钟被停，直到字节编程完成；

6）回写原来不需修改数据；

7）取消对flash编程使能。

8）恢复Bank的选择。

5.3 驱动库设计

参考 VC 中静态库的概念：函数和数据被编译进一个二进制文件（通常扩展名为.LIB）。在使用静态库的情况下，在编译链接可执行文件时，链接器从库中复制这些函数和数据并把它们和应用程序的其他模块组合起来创建最终的可执行文件（.EXE 文件）。当发布产品时，只需要发布这个可执行文件，并不需要发布被使用的静态库。

把上两种芯片的硬件驱动函数建立为一个硬件驱动库，要生成某芯片公司的智能卡时，链接器从库中复制相应函数和数据，并把它们和应用程序的其他函数组合起来，然后编译生成所需要文件。应用到该公司智能卡时，只需要把生成文件以 APDU 格式灌进卡里，并不需要对应用层重新开发。如图 5-3 所示：

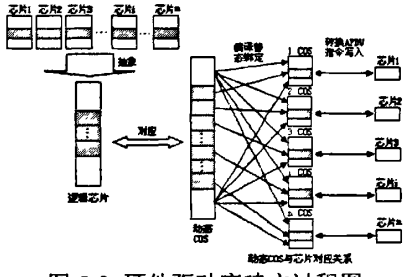


图 5-3 硬件驱动库建立过程图

Figure 5-3 Hardware Drivers Set Up Map

5.4 COS 生成器的设计

COS 生成器需要完成增加新芯片硬件驱动、删除芯片硬件驱动和生成 COS，主要包括两部分：

1）驱动库管理器：主要负责驱动库增加新芯片驱动，删除芯片驱动。其包括应用功能函数库、统一接口层、硬件驱动函数库和驱动管理器。如图

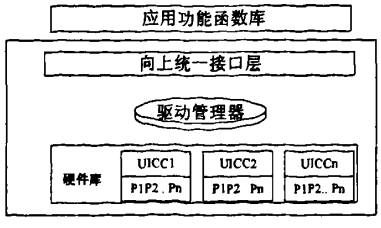


图 5-4 驱动库管理器结构图

Figure 5-4 Driver Library Manager Chart

应用功能函数库：存放安全模块、通信模块、文件模块、命令模块和应用菜单

相关函数，其中文件模块分为 32K、64K 和 128K 文件系统。可根据不同芯片或开发要求选择不同的文件系统。安全模块包括 CDMA、TD-SCDMA、WCDMA 和 CDMA2000 三种鉴权算法。应用菜单分为 UTK 应用和 OTA 应用。可根据卡片应用不同网络，需求不同菜单进行绑定。

硬件驱动函数库：存放各个公司芯片硬件驱动函数。如上述所说的 A 公司和 B 公司。

统一接口层：对每个加入硬件库的硬件驱动函数进行检测，检测其每个驱动函数是否符合接口统一名称。逻辑结构如图 3-15 所示：

实际上，这里所说的统一接口层，只是对硬件驱动函数中的函数名称进行检测。而函数是否能正确、高效地对硬件进行操作，就需要在仿真器中进行实现。

驱动管理器：不论是硬件驱动函数库还是应用功能函数库其存在形式都是用.C 文件形式（汇编程序以.a51 文件形式）存放在库文件夹中。增加硬件驱动函数时通过统一接口检测函数是否匹配。若函数接口与统一接口相匹配，就把该驱动函数增加到硬件驱动库中，并增加相应的描述记录，以便以后维护。删除硬件驱动函数就相对简单，只是把选中的芯片相对应的函数及其相关记录信息删除就行了。如下图 5-5、图 5-6 所示：

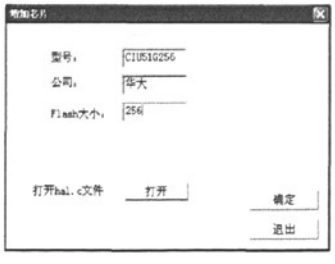


图 5-5 增加芯片界面图
Figure5-5 Add Chip Interface Graph

图 5-6 删除芯片界面图
Figure5-6 Remove Chip Interface Graph

2）COS 生成器：主要实现对指定芯片实现某特定功能进行绑定编译生成 HEX 文件。实现绑定式编译方法有两种：条件编译和批处理。这两种方法各有各的特点，本文采用这两种方法相结合。在宏观方面使用批处理，如两个模块（安全模块、文件模块）进行绑定；在微观方面使用条件编译，如 DES 选择或命令选择。

使用条件编译方法实现绑定：如 DES 选择方面，有些芯片没有配有 DES 协处理器，如果 COS 需要运用到 DES 算法时只能通过软件算法实现。

```
#ifdef _HAL_CALDES_
extern  void calDES(····································);
#endif
```

```
#ifdef _HAL_CALDES_
void calDES(·····)
{
·····//软件实现DES算法
·····
}
#endif
```

如果在没有配置 DES 协处理器芯片中运用到 DES 算法时，只需在配置文件表中定义宏_HAL_CALDES_，编译器会自动绑定软件实现方法。如果该芯片配有 DES 协处理器，不在配置文件定义宏，编译器绑定硬件方式实现 DES 算法。

利用宏还可以对文件系统进行规划。使用宏名字代替一个字符串，可以减少程序中重复书写某些字符串的工作量；用宏名字代替，简单不易出错，因为记住一个宏名要比记住一个无规律的字符串更容易，而且在读程序时能立刻知道它的含义，当需要改变某一个常量时，可以只改变#define 命令行，一改全改。

由于各公司提供的芯片 FLASH 大小会有差别，例如 A 公司和 B 公司芯片 FLASH 空间大小分别是 112K 和 128K，利用宏名字统筹整个 FLASH 空间。当 COS 从一芯片移植到另一芯片时，只改变宏命令行，一改全改。

```
#define FLASH_SIZE 112*1024          //FLASH 存储器大小,单位 B
#define FLASH_BLOCK_SIZE 512         //FLASH 擦除块大小，单位 B

#define FILESYSTEMBEGINADD 0x34000   //文件系统开始的地址
#define FILESYSTEMENDADD   0x3FFFF   //文件系统结束的地址
#define FILESYSTEMSIZE     0x0BFFF   //文件系统大小
```

使用批处理方法实现绑定：批处理就是从芯片硬件驱动库中，选择指定的驱动函数；并且在功能函数库中，选择所需的功能函数，把他 copy 到指定文件夹中，然后调用 keil 编译器对这些函数进行编译，生成 HEX 文件。如图 5-7 所示

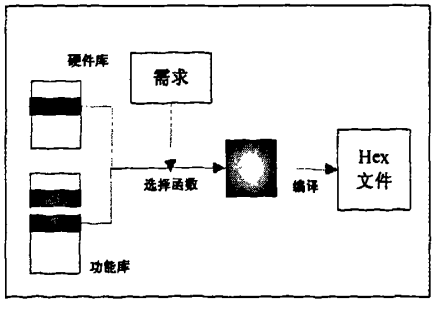


图 5-7 COS 生成图

Figure 5-7 COS Generated Map

硬件驱动库选择相应的芯片驱动：

```
StrHalLoad=".\\cos\\hal\\"+m_ChipName+"\\hal.c";
CopyFile(StrHalLoad,".\\cos\\hal.c",true);
```

功能函数库选择相应的文件系统模块：

```
switch(m_iFileSys)
{
case 0:            //选择32k文件系统
    CopyFile(".\\cos\\filesys\\fat_32k.h",".\\cos\\fat.h",true);
    CopyFile(".\\cos\\filesys\\fat_32k.c",".\\cos\\fat.c",true);
    CopyFile(".\\cos\\filesys\\fs_32k.h",".\\cos\\fs.h",true);
    CopyFile(".\\cos\\filesys\\fs_32k.c",".\\cos\\fs.c",true);
    break;
case 1:          //选择64k文件系统
    CopyFile(".\\cos\\filesys\\fat_64k.h",".\\cos\\fat.h",true);
    ..................
    ..................
case 2:
    ..................
    ..................
default:
    break;
}
```

使用批处理方式对所需文件进行编译：

```
SET C51INC=C:\Keil\C51\INC\
SET C51LIB=C:\Keil\C51\LIB
SET CPU_TYPE=8032AH              //设置硬件配置
SET CPU_VENDOR=Intel
SET UV2_TARGET=Target 1
SET CPU_XTAL=0x00B71B00
C51.EXE .\cos\main.c LARGE OMF2 BROWSE DEBUG PRINT(.\cos\main.lst)
OBJECT(.\cos\main.obj)              //编译.C文件
C51.EXE .\cos\apdu.c LARGE OMF2 BROWSE DEBUG PRINT(.\cos\apdu.lst)
OBJECT(.\cos\apdu.obj)
..............................
..............................
..............................
ax51.exe .\cos\STARTUP.A51 SET (LARGE) DEBUG EP  //编译.a51文件
ax51.exe .\cos\asm.a51 SET (LARGE) DEBUG EP
LX51.EXE @cos_prj.lnp               //链接所编译的文件
OHX51.EXE "cos_prj" HEX             //生成.HEX文件
pause
```

第六章 本文创新与成果

2007年7月到现在，项目组一共开发3个COS，并实现电话通信、短信互发、UTK应用、超级话部等功能。在开发过程中实现了COS系统结构、绑定式设计两方面的创新。

1.创新点

1) COS系统结构的创新

传统COS的命令处理流程一般采用层次系统结构，本文使用间接的层次结构。在传统COS中，所有信息需要安全模块的处理；在UTKCOS中，不是所有的信息均需要安全模块的鉴权、数据加/解密，在COS设计中采用部分数据信息进行处理的方式。并在此结构上增加多一个接口层，使其适用普遍芯片成为可能。

2)绑定式设计

绑定式设计是在COS系统结构创新的基础上提出来的。在传统的COS设计中，大多是针对特定智能卡进行设计，在将COS应用移植到另一智能卡上时，需要经过复杂的移植过程而实现。绑定式设计是针对COS移植设计方案，其方法是在上层逻辑应用不变的情况下，改变硬件驱动函数实现COS移植。本文归纳出COS设计中共同的部分和不同的部分，将不同部分设计成相应的组件形式，形成一个动态的COS，在确定了某种芯片后进行静态绑定编译，形成特定芯片的COS。本文并使用驱动管理器、功能库、硬件库、COS生成器等概念描述了绑定式的系统结构。

2.绑定式研究取得的成果

绑定式COS研究取得的成果：项目从2007年7月开始到现在，项目组一共开发3个COS，如下表所示：

表6-1开发时间表

table6-1 design timetable

名称	应用网络	芯片	开发人数	开发时间	用时
USIM	TD-SCDMA	A112F	4人(08.7后9人)	2007.7-2008.10	15个月
UTK	CDMA	A112F	9人	2008.10-2009.1	4个月
EVDO	CDMA	B256F	9人	2009.3-2009.4	2个月

从USIMCOS开发到EVDOCOS完成，开发时间越来越少，有各个方面的原因：首先是熟练程度，第一个COS从零开始，经历一段长时间的学习和探讨。其次是绑定式设计，无论在硬件驱动移植还是功能应用上的裁减，绑定式设计思想对加快COS开发完成有着重要作用。值得指出的是由A112F型号芯片UTKCOS移植到B256F芯片所用的时间是一个星期。也就是说不增加新应用的情况下，能够从一芯片快速移植到另一芯片。

结 论

目前智能卡已在我国内广泛使用，尤其是电信行业智能卡的发卡量巨大，约占一半以上，在 3G 系统放号后，这一比例将继续增大，电信行业智能卡的商机是无限的。智能卡 COS 研发是智能卡行业产业链中的一个重要环节，由于多种原因，目前国内能进行 COS 研发的企业并不多，这极大的制约了，我国智能卡的应用推广。学校从推动行业发展和产学研结合的角度出发，与 XX 智能卡公司合作共同研发 UTK COS，本文正是建立在此基础上的。在本文设计过程中取得了 COS 研发的一些突破，也有一些需要继续研究。

在本文中，首先描述了智能卡的行业背景、电信智能卡的状况等与智能卡相关的一些基础知识。然后本文在传统的系统结构上提出新的 COS 系统结构，并实现两款不同芯片的硬件层驱动函数。最后针对 COS 移植困难的情况，提出了绑定式 COS 设计。

由于时间精力的限制，在 COS 的设计研发过程中还有很多重要的问题，在本文中没有探究到，在未来也需要继续探究。使用绑定式设计中，可以实现功能层代码重用，但在其中存在一个关键问题：在国产芯片中（如华虹、同方、华大）都是使用 8051 内核，使用的开发工具是 keil C。若是应用在非 Keil C 编译环境（如三星）使用三星公司自己开发的编译工具，此时绑定式设计就无法实现移植，必须重新开发。本文对此问题还未作探究，在未来须继续探讨此问题。

参考文献

[1]中国投资咨询网.2007－2008 年中国智能卡市场研究年度报告[M].中国投资咨询网，2007.

[2]夏志远.智能卡操作系统的研究与实现[D].湖北：华中科技大学图书馆，2006

[3]马奇学.WCDMA 网络的USIM 卡研究与实现[D].北京：北京邮电大学图书馆，2005 年5 月。

[4]ISO. ISO/IEC 7816-3 Smart Card Standard, Part 3 Electronic Signals and Transmission Protocols[S].ISO/IEC, 2006-10-30.

[5]3GPP T3.3GPP TS31.102 3rd Generation Partnership Project; Technical Specification Group Core Network and Terminals; Characteristics of the Universal Subscriber Identity Module (USIM) application, V7.4.1[S].3GPP T3, 2006-04.

[6]GSM 11.11 数字蜂窝电信系统（PHASE 2+）用户识别模块——移动设备（SIM—ME）接口规范[S].ESTI,1996-08.

[7]GSM 11.14 数字蜂窝电信系统（PHASE 2+）用户识别模块——移动设备（SIM—ME）接口的SIM 应用工具集规范[S].ESTI, 1996-08.

[8]黄健.智能卡COS的研究与设计[D].广东：广东工业大学图书馆，2008年5月.

[9]求是科技，李翔.智能卡研发技术与工程实践[M].北京：人民邮电出版社，2003.

[10]ETSI. ETSI TS102 221 Smart Cards Smart Cards UICC -Terminal interface Physical and logical characteristics,V7.8.0[S]. ETSI, 2007.2.

[11]ISO.ISO/IEC 7816-4 Smart Card Standard, Part 4 Interindustry Commands for Interchange[S].ISO/IEC, 2005.

[12]张方舟，3G安全体系结构[M].信息产业部，2004-3-26.

[13]ETSI.ETSI TS131.102 Universal Mobile Telecommunications System (UMTS) Characteristics of the Universal Subscriber Identity Module (USIM) application, V7.8.0[S].ETSI,2007-03.

[14]张超毅，通用javacard平台的研究[D].北京：北京邮电大学图书馆，2006.

[15]刘玉珍,涂航,张焕国.实用智能卡操作系统的设计与实现[J].武汉大学学报(自然科学版)，2000年6月,第46卷,第3 期:309-312.

[16]张鲁国,马自堂.智能卡操作系统中存储管理设计[J].微计算机信息，2005年,第21卷，第8-3期：18-20.

[17]杨瑞霞.智能卡的安全性分析[J].河北省科学院学报，2006年9月，第23卷,第3期:57-60.

[18]范小红，吴今培，张其善.智能卡文件系统的安全访问机制[J].微计算机应用，2004年1月，第25卷，第1期:37-40.

[19]何文,李爽.基于超驰16芯片的COS 设计与测试[J].微计算机信息，2007年,第23 卷,第4-3期:75-78

[20]杨帆，张焕国. 金融智能卡操作系统安全体系研究[J].计算机应用研究，2005年，第9期:95-98.

[21]王玉厚.智能IC卡关键技术研究与应用[D].南京：南京工业大学.2006年。

[22]ETSI.TS102 223 Smart Cards, Card Application Toolkit (CAT),V7.6.0[S]。ETSI,2007-2.

[23]ETSI.TS131 122 Universal Mobile Telecommunications System (UMTS); Universal Subscriber Identity Module (USIM)conformance test specification, V6.4.0[S].ETSI，2007-03.

[24]3GPP.TS34.131 3rd Generation Partnership Project; Technical Specification Group Core Network and Terminals; Test Specification for C-language binding to(Universal) Subscriber Interface Module ((U)SIM)Application Programming Interface (API)，V7.0.0[S].3GPP，2007-06.

[25]中国电信CDMA卡需求规范－UIM卡分册(v1.0)[S].中国电信集团公司，2008-01.

[26]中国电信CDMA卡测试规范－UIM卡分册(v1.0)[S].中国电信集团公司，2008-01.

[27]中国电信 CDMA 卡需求规范－ESN 信息上报分册(v1.0)[S].中国电信集团公司，2008-01.

[28]中国电信 CDMA 卡需求规范－UTK 应用分册(v1.0)[S].中国电信集团公司，2008-01.

[29]中国电信 CDMA 卡需求规范－单向全能读分册(v1.0)[S].中国电信集团公司，2008-01.

[30]中国电信 CDMA 卡需求规范－超级号簿分册(v1.0)[S].中国电信集团公司，2008-01.

[31]中国电信 CDMA 卡测试规范－超级号簿分册(v1.0)[S].中国电信集团公司，2008-01.

[32]曾涌波.CDMA系统接纳控制的研究和WCDMA中NAS和USIM的实现[D].南京：东南大学图书馆，2006年.

[33]刘楠.SIM技术体系和业务研究[D].北京：北京邮电大学，2007年4月.

[34]牛志愿.OTA空中下载技术原理与实现[D].长春：吉林大学，2004年4月.

[35]陈文扬.安全的java SIM卡空中下载机制的研究与实现[D].北京：北京邮电大学，2006年3月.

[36]秦保安.动态多应用智能卡中的卡操作系统研究及实现[D].武汉：华中科技大学,2005年4月.

[37]曹计昌，李纯.UCard中多COS调度问题的研究[J].计算机工程与科学,2006年，第28卷，第2期：128-131.

[38] 张德学，郭立，傅忠谦.Javacard CPU 的设计与实现[J]. 计算机工程,2007年5月, 第33卷，第10期:280-282.

[39]孙海涛,马久和,于春光. USB智能卡应用技术研究USB智能卡应用技术研究[J].科学技术与工程,2007年8月,第7卷第16期:4209-4213.

[40]唐业,张申生.多应用智能卡的空间和时间优化[J].计算机应用与软件,2007年7月,第24卷,第7期:60-63.

[41]翰纪宏，谷大武，任艳丽.机卡分离中基于身份的认证及密钥协商协议[J].计算机工程,2007年3月,第33卷,第5期:126-131.

[42]曹计昌，周宇杰.基于高地址约束和按需存储分配的UCard底层调度模块研究.计算机工程与科学,2007年，第29卷，第6期：123-147.

[43]刘文远,司亚利,刘永由.基于智能卡的多银行离线电子现金方案[J].计算机应用研究,2007年10月,第24卷第10期:116-121.

[44]尉永清,刘培德.双接口智能卡COS软件的设计与实现[J].山东师范大学学报(自然科学版),2006年6月,第21卷,第2期:124-126.

[45]董威，杨义先.一种跨行业多应用智能卡系统模型及实现[J].计算机工程,2007 年4月,第33卷,第8期:230-232.

[46]张伟丽,杨鼎才,元文华.智能卡双向认证协议的改进[J].微计算机信息, 2007年,第23卷,第9-3期:78-80.

攻读学位期间发表的论文

1. 游剑锋，汤荣江，李代平，刘志武，王挺，黄健.智能卡操作系统结构研究，现代计算机，2009.1 总第 299 期，Cn44-115/tp ISSN1007-1423

致谢

本文的COS设计是建立在一个团队项目基础之上的，在设计过程中得到作者所在的实验室的大力支持。在这里首先感谢导师汤荣江副教授、李代平教授在研究生期间的指导，在本文设计中的指导与帮助；感谢黄健和郭广义对底层驱动设计的支持，刘志武同学对安全部分设计的支持，王挺同学对文件系统设计的支持。再次感谢导师汤荣江副教授、李代平教授和各位评审老师在百忙之中抽出宝贵的时间审阅本论文！

附录

部分函数代码实现：

1. 主控函数代码实现：

```
void main(void)
{
   initHardware();
   initSoftware();
   sendATR(); //ATR响应和PPS波特率协商
   PPS();
   while(1)        //main loop
   {
       idleMode();   //包括当前数据，时钟休眠。
       readUART(APDU, 5);   //读取APDU命令
       if (APDU格式错误)
           status标记;
       else
       {
           switch(INS)
               {   命令分析处理;     }
           switch(status)
               {   获取响应数据;     }
        }
           writeUART(SW,0x02);   //发送响应状态
   }
}
```

2. A 公司 idleMode 函数代码实现：

```
void idleMode(void)
{  //记录当前状态
   backupCLKSEL = CLKSEL;  //backup CLKSEL
   backupCHIPSC = CHIPSC;  //backup Chipsc
   backupRNGCTL = RNGCTL;
   backupIDLE_EN= IDLE_EN;

   //时钟休眠
   IE |= 0x84;          //open UART interrupt
   RNGCTL |= 0x01;      //shut off RNG
   CLKSEL &= 0xFE;      //use CLK_EXT, stop CLK_INT
   CHIPSC &= 0x7D;      //in idle state, STDBY_N and FD are disabled; HVD and LVD
                          are not changed.
   IDLE_EN |= 0x0F;     //enable IDLE_EN
   PCON |= 0x01;        //idle CPU
                        //RBF trigs interrupt; TACT trigs NO interrupt;
                        //PE neither if PH=1.

   //产生 UART 中断，恢复当初状态。
   IDLE_EN= backupIDLE_EN;
   CLKSEL = backupCLKSEL;  //restore CLKSEL
   CHIPSC = backupCHIPSC;  //restore CHIPSC
   RNGCTL = backupRNGCTL;
```

```
}
```

3. PPS 算法代码实现：

```
PPS()
{
    readUART (&PPSS,1) ;//读取第一个字节
    if(0xff != PPSS)    //首字节非“FF”，表示不需 PPS 协商，改变标志返回函数。
    {
        gExpect4Chars = 1;
        return;
    }
    readUART (&PPS0,1) ;//读取下一个字节
    expectedCharNum = 1;//判断 PPS1，2，3 和 PCK 是否存在。
    if(PPS0&0x10)   expectedCharNum++;
    if(PPS0&0x20)   expectedCharNum++;
    if(PPS0&0x40)   expectedCharNum++;

    //读取 PPS1,2,3, and PCK
    for(i = 2; i < expectedCharNum + 2; i++)
    {
        while(!RBF);
        gCommBuf[i] = UBUF;
    }
    进行相应波特率设置;
    发送 PPS 响应;
}
```

4. A 公司发送函数代码实现：

```
void writeUART(unsigned char * sendBuf, unsigned short intLen)
{
    unsigned short data intI;
    UCR.TR=1;   //发送模式
    for( intI = 0;  intI < intLen;  intI++ )
    {
        发送多个字节数据;
        while(!USR.TBE);
        UBUF = sendBuf[intI];
        USR.TBE = 0;
    }
    while(!USR.TBE);
    UCR.TR=0;   //返回接收模式
}
```

5. A 公司接收函数代码实现：

```
void readUART(unsigned char xdata * receiveBuf, unsigned char data expectLen)
{
    unsigned char data I;
    for(i = 0;  i < expectLen;  i ++)
    {   //读取expectLen字节长度数据
        while(!USR.RBF);
```

```
            receiveBuf[i] = UBUF;
        }
    }
```

6. A 公司 sector 擦除函数代码实现

```
STATUS eraseFlashSector(unsigned char xdata * pDest)
{
    unsigned char backupIE;
    backupIE = IE;
    IE = 0x90;       //禁止FLASH以外的其余中断
     ..........;
    FL_PATCH = 0x78; //only for the first sector, in case of different ISR4
    FL_CTL = 0;      //disable FLASH standby;  clear ERR, OVER
    PCON |= 0x01;    //idle CPU
    ..........;
    return SW9000;         //擦除成功
}
```

7. A 公司写 flash 函数代码实现：

```
定义写flash缓冲区:unsigned char xdata gFlashBuf[BPS] _at_ (XRAM_TOP - BPS + 1);
STATUS writeFlash(unsigned char mode, unsigned short intLen, unsigned char *pSrc,
                  unsigned char *pDest)
{
    选择XDATA数据区;
    FL_CTL = 0;      //disable FLASH standby; clear ERR, OVER
    FL_PATCH = 0x78;//enable patch
    IE = 0x90;       //ATTENTION: no interrupt other than flashISR should be
                       enabled during flash erasing or programming.

    for(temp_i=0;temp_i<write_Len;temp_i++)
    {
        FL_SDP1 = 0x**;
        FL_SDP2 = 0x**;
        *pDest=pSrc[temp_i];
        pDest++;
        数据检测。若是发生错误，就及时返回，并返回错误信息。
    }

    FL_CTL = 0x80;  //enable FLASH standby; clear ERR, OVER
    IE = 0;         //disable all interrupts
    FL_PATCH = 0;   //disable patch
    return status;
}
```

8. A 公司读 FLASH 函数代码实现：

读 FLASH 比写 FLASH 简单得多，是需要切换到相应的 bank 区进行读取就可以。

```
void read_flash(unsigned char *buf,unsigned char *pDest,char len)
{    ..........;
    地址扩展，BANK 切换;
    for(temp_i=0; temp_i<len; temp_i++)
```

```
    {  *buf=*pDest;
       buf=buf++;
       pDest++;
    }
}
```

9. B 公司串口发送函数代码实现：

```
void writeUART(unsigned char*sBuffer,unsigned char sCnt)
{
    while(i < sCnt)
    {
        for(j = 0; j < 6; j++)
        {
            while(TB);  //等待串口空闲
            SBUF = sBuffer[i];
            if(!TX)  //检测数据是否错误
                break;
            while(TB); //等待串口发送完毕
        }
        if (j >= 6)
            return; //one byte send error
        i++;
    }
}
```

10. B 公司串口接收函数代码实现：

```
void readUART(unsigned char *rBuffer, unsigned char rCnt)
{
    unsigned char i;
    for(i=0x00;i<rCnt;i++)
    {
        if (!TB)//串口是否空闲
            STANDBY = 1;//启动串口
        while(!RF);
        rBuffer[i] = SBUF;
    }
}
```

11. B 公司写 Flash 函数代码实现：

```
STATUS writeFlash(unsigned char mode, unsigned short intLen, unsigned char *pSrc,
unsigned char *pDest)
{
    程序地址变换; //通过运算，确定该地址属于那个区。

    PMWE=0x01; //触发flash编程使能
    BANK=bank_num; //选择Bank区

    把不需要改写数据放进缓冲区;

    SDP1=0x**; //写入SDP指令串
```

```
    SDP2=0x**;
    ..........

    for(temp_i=0;temp_i<write_Len;temp_i++)
    {
        *pDest=pSrc[temp_i];        //向指定地址写入字节
        pDest++;
        数据检测。若是发生错误，就及时返回，并返回错误信息。
    }

    回写原来不需修改数据;
    PMWE=0x00; //取消对flash编程使能。
    BANK=0x01; //选择Bank区恢复Bank的选择。
}
```

12. B 公司读 Flash 函数代码实现：

读 FLASH 比写 FLASH 简单得多，是需要切换到相应的 bank 区进行读取就可以。

```
void read_flash(unsigned char *buf,unsigned char *pDest,char len)
{    ..........;
    地址扩展，BANK 切换;
    for(temp_i=0; temp_i<len; temp_i++)
    {  *buf=*pDest;
       buf=buf++;
       pDest++;
    }
    BANK 区还原;
}
```